計測自動制御学会東北支部 第 213 回研究集会 (2003.12.12)
資料番号 213-11

Octaveを用いた並列信号処理アルゴリズム開発環境の構築

Implementation of a Prototyping Environment for Parallel Signal Processing Algorithms Using Octave

○小原武†, 青木孝文†, 樋口龍雄‡

○ Takeshi Obara†, Takafumi Aoki† and Tatsuo Higuchi‡

†東北大学大学院情報科学研究科, ‡東北工業大学工学部電子工学科

†Graduate School of Information Sciences, Tohoku University,
‡Department of Electronics, Tohoku Institute of Technology

キーワード： 並列処理 (parallel processing), 信号処理 (signal processing), Octave (Octave), MPI (MPI)

連絡先： 〒980-8579 仙台市青葉区荒巻字青葉 05 東北大学大学院情報科学研究科 青木研究室
小原武, Tel.: (022)217-7169, Fax.: (022)263-9308, E-mail: obara@aoki.ecei.tohoku.ac.jp

1. はじめに

近年，科学技術計算の分野において要求される演算能力は増加の一途をたどっている．ディジタル信号処理の分野においても，膨大な計算量を必要とする信号処理アルゴリズムの実装や大規模な信号処理システムの迅速なプロトタイピングなどを行うことが求められており，これらを高速に短時間で処理する手段として，並列処理技術の有効性が注目されている．一方，近年，PC などの汎用品を利用した安価な並列処理技術が注目され，COTS (Commercial Off-the-Shelf) クラスタなどの名称で普及が進んでいる．ディジタル信号処理の分野においても，これらの安価な並列処理技術を積極的に活用するための高水準なシステム開発環境が求められている．

これに対し，筆者らは，並列計算機上で手軽に利用できる高水準並列ディジタル信号処理環境として並列Octaveを構築してきた[2)]．GNU Octave[1)]は，ディジタル信号処理の分野で広く用いられるMatlabと互換性を持つフリーソフトウェアである．オリジナルの Octave には並列計算を行う機能が存在しないので，並列信号処理アルゴリズム開発環境を構築するために，並列計算ライブラリなどを用いた拡張が必要である．今回作成した並列 Octave には，最もよく使われている並列計算ライブラリの 1 つである MPI (Message Passing Interface) [3)] ライブラリを利用して並列処理用の関数群を実装した．並列 Octave は，PC クラスタなどに手軽に導入できるため，並列信号処理アルゴリズムのプロトタイピングや実証研究などに有用である．

本稿では，Octave を用いた並列信号処理アルゴリズム開発環境の作成と，分散メモリ型並列計算機と共有メモリ型並列計算機上で作成した並列 Octave を評価した結果について述べる．

2. 並列 Octave

2.1 GNU Octave

GNU Octave は，もともと， J. B. Rawlings (The Univ. of Wisconsin-Madison) と J. G. Ekerdt (The Univ. of Texas) らによる化学反応器の設計に関する教科書の付属ツールとして作成された．現在は J. W. Eaton (The Univ. of Wisconsin-Madison) らによって開発が継続されている．Octave はディジタル信号処理の分野で広く利用される MATLAB と互換性を持つ，いわゆる MATLAB クローンの一種である．MATLAB に慣れたユーザならば，各種信号処理アルゴリズムの実装やシステムのプロトタイピングを手軽に行うことができる．また，Octave は General Public License (GPL) に基づくフリーソフトウェアであるため，ソースコードの改変及び再配布を自由に行うことができる．

Octave は，C++を基本言語として，Fortran および各種の数値計算ライブラリ，さらには flex や bison といった字句・構文解析系生成ツールなどを用いて構築されている．そのソースコードは，オブジェクト指向言語の抽象化能力を活用した現代的な設計であり，注意深く検討されたクラス構造に基づいて改変・拡張を系統的に行うことができるように配慮されている．また，約 340 ページのマニュアルをはじめとして，インターネットから得られるさまざまなドキュメント類が存在する．

Octave は，組み込み関数および実行時に動的にリンクされる関数を定義するためのマクロ (DEFUN_DLD) が用意されている．このマクロを記述した C++ソースコードは，oct ファイルと呼ばれるオブジェクトコードにコンパイルされる．作成した oct ファイルを Octave が読み込むことで C++ソースコードで記述した機能を実行することができる．このようにマクロを利用して，新たな関数を定義し，システムへ導入することが可能である．並列計算用関数の追加に際しては，oct ファイルを利用した．

2.2 並列計算環境

一般に並列計算を実行するプラットフォーム (並列コンピュータ) は，大きく共有メモリ型と分散メモリ型に分けられる．共有メモリ型には，主に 1 台のマシンに複数の CPU を持つ SMP (Symmetric Multi-Processor) マシンの形で提供される．各 CPU は，全てのメモリに対して同じようにアクセスすることができるが，CPU の数が多くなると高価になる．分散メモリ型は，主にネットワークで結合された複数のマシンという形で構成される．各マシンは，一般に安価に作ることができるが，他のマシンのメモリ内のデータを参照する際に，通信してデータを送受信しなくてはいけないためオーバーヘッドが生じる．

並列プログラミングをする際に，MPI (Message Passing Interface) ，PVM (Parallel Virtual Machine)[4]，HPF (High Performance Fortran)[5]，OpenMP[6]といった並列計算用のライブラリやプログラミング言語がよく使われている．MPI は，あるプロセスから他のプロセスへデータを明示的に送信する通信方式であるメッセージパッシングに基づいた並列計算を行うための規格である．PVM は，メッセージパッシングを利用して並列計算を行うためのソフトウェアである．プロセス間の通信以外にも，フォールトトレラント機能などを備えている．HPF は，Fortran を元とした拡張言語である．データ分散を行う指示文などを基にして，コンパイラが並列システムに応じた計算処理のアロケーションを行う．OpenMP は，主に共有メモリ型並列計算機で用いられ，C や Fortran などで書かれた逐次プログラムに対して指示文などを用いて並列プログラミングを行うための規格である．

今回は，MPI ライブラリを利用して Octave に並列インタフェースを実装した．MPI は，さまざまな要求に応えるために，単一ホスト間の同期・

非同期や複数ホスト間の集団命令など多数の関数を定義している．これらの関数群により，統一的なインタフェースと柔軟な並列プログラミングを実現しており，複数のプラットフォーム上でプログラムを作成することができる．MPI が提供している関数群を以下にまとめる．

- 一対一通信
- 集団通信
- プロセスグループ通信コンテキスト
- プロセストポロジー
- Fortran，C からの呼び出し形式
- 環境問い合わせ
- プロファイリングインタフェース

本稿では，MPI で定義されている一対一通信関数・集団通信関数・環境問い合わせ関数のうち，利用頻度の高い関数群について Octave の並列計算用関数に追加した．

Octave では，整数型・複素数型・文字・2 次元配列などのデータ形式が抽象化されている．データ形式の抽象化により，ユーザは変数の型を意識する必要がないため，迅速なアルゴリズムのプロトタイピングが可能である．Octave に並列計算用関数を追加するにあたって，このデータ形式の抽象化を失わないようにする必要がある．また，並列 Octave から利用できる並列計算用関数群の定義を MPI 関数群の定義となるべく同じようにすることで，ユーザが新たに並列プログラミング技術を習得する労力を軽減することができる．

Octave 上で MPI 関数群を利用するために，並列 Octave から利用できる並列計算用関数群には以下のような変更を加えた．

- MPI 関数群のデータ型である datatype (c.f. MPI_CHAR，MPI_DOUBLE など) に関する引数の入力を省略した
- データ型に関する取り扱いをユーザから隠蔽するため，通信時に Octave 上のデータ形式を char 型のバイトストリームに変換した
- Octave では引数の参照が行えないため戻り値として取り扱った

追加した Octave の並列計算用関数と，元の MPI 関数の一例を示す．

Octave
[retval status] = MPI_RECV (size, source, tag, comm)

出力	retval	受信結果の保存する変数
入力	size	受信するバイトストリームのサイズ
入力	source	送信元のランク
入力	tag	メッセージタグ
入力	comm	コミュニケータ
出力	status	ステータスオブジェクト

MPI
MPI_RECV(buf, count, datatype, source, tag, comm, status)

出力	buf	受信バッファの先頭アドレス
入力	count	受信バッファ内の要素数
入力	datatype	受信バッファの各要素のデータ型
入力	source	送信元のランク
入力	tag	メッセージタグ
入力	comm	コミュニケータ
出力	status	ステータスオブジェクト

Octave で並列処理を行うために追加した並列計算用関数を Table 1に示す．これらの関数群により，Octave 上で同期・非同期の通信処理と集団通信処理を行うことができる．

2.3 並列 Octave の構成

Octave は，バッチファイルによる非インタラクティブな処理に加え，インタラクティブな処理も可能である．このインタラクティブな処理の実行によって，スクリプトのデバッグなどの効率が大幅に向上する．このため，並列処理を行う場合もインタラクティブ性を残すことは有用である．インタラクティブ性を保存するには，ユーザの操作している Octave プロセスから他の Octave プロセスを操作する必要がある．そこで，並列 Octave でのプロセス制御には，socket をベースとしたシン

Table 1　並列 Octave で利用することができる並列計算用関数一覧

関数名	説明
mpi_init	並列実行環境の展開
mpi_finalize	並列実行環境の終了
mpi_abort	並列実行環境の停止
mpi_comm_rank	グループ内のランクを返す
mpi_comm_size	グループのサイズを返す
mpi_send	メッセージの同期送信
mpi_recv	メッセージの同期受信
mpi_irecv	メッセージの非同期受信
mpi_test	非同期受信の確認
mpi_wait	非同期受信の完了
mpi_bcast	グループ内のブロードキャスト
mpi_barrier	グループ内のバリア同期

グルクライアント・マルチプルサーバーモデルを採用した (Fig. 1)．socket によって確立されたセッションは，サーバ側の Octave を制御するために用いられる．このモデルを採用することにより，クライアントからの指示でサーバに命令を実行させることができ，インタラクティブ性と非インタラクティブ性を実現することができる．

作成した並列 Octave は，並列計算用関数の呼び出しを行う oct ファイル群とその呼び出しに対して通信機能を提供するための実行ファイルである MPI モジュールによって構成される．並列実行環境を展開すると，Octave と MPI モジュールは別々のプロセスとして動作する．これによって，実行環境展開の柔軟性を確保できる．Octave のデータを送受信するために，socket より高速なプロセス間通信 (InterProcess Communication: IPC) によるセッションを利用する．Octave と MPI モジュールとの IPC には名前付きパイプ (Named-Pipe) を用いた．

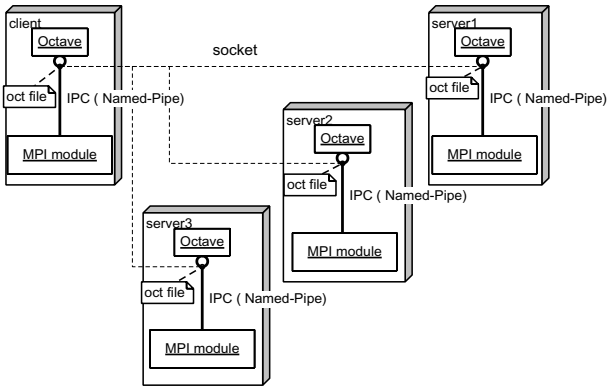

Fig. 1　並列 Octave の構成

3.　性能評価

以上のような並列拡張を行った Octave について，PC クラスタシステム (Fig. 2) と PrimePower 600 (Fig. 3) 上で性能評価を行った．評価には，非線形ダイナミクスの計算と位相限定相関法を用いた指紋照合を使用した．

3.1　実験環境

性能評価を行った実験環境について述べる．実験環境には PC クラスタと PrimePower 600 の 2 種類の並列計算機を用いた．PC クラスタは分散メモリ並列計算機であり，サーバー 1 台，ノード 14 台によって構成される．PrimePower 600 は共有メモリ並列計算機であり，8 CPU を搭載している．PC クラスタシステムの構成と外観をそれぞれ Table 2 と Fig. 2に，PrimePower 600 の構成と外観をそれぞれ Table 3と Fig. 3に示す．

3.2　非線形ダイナミクスの計算の並列化

並列処理時に通信を要する例として非線形ダイナミクスの計算を取り上げた．本稿では，興奮波を発生する反応拡散モデルである FitzHugh-南雲モデル[7]を用いた．FitzHugh-南雲モデルは次式

Table 2　PC クラスタの構成

	Server	Node
CPU	Intel PentiumIII 1GHz Dual	Intel PentiumIII 1GHz
Memory	PC800 DRDRAM 1GB	PC133 SDRAM 512MB
Network	1000BASE–SX	100BASE–TX
OS	Linux 2.4.20	
Number of Nodes	1	14

Fig. 2　分散メモリ型並列計算機 (PC クラスタ)

Fig. 3　共有メモリ型並列計算機 (PrimePower 600)

Table 3　PrimePower600 の構成

CPU	SPARC64 GP 400MHz
Memory	ECC SDRAM 2GB
OS	Solaris 5.8
Number of CPUs	8

で表される．

$$\begin{cases} \tau(\partial u/\partial t) &= \epsilon^2\nabla^2 u + u(u-\alpha)(1-u) - v \\ \partial v/\partial t &= u - \gamma v \end{cases} \tag{1}$$

ここで，u, vは物質の濃度，$\alpha, \epsilon, \gamma, \tau$はパラメータである．FitzHugh-南雲モデルは，平衡点濃度に保たれている空間上のある地点に閾値以上の濃度を与えると，その点から興奮波を発生する (Fig. 4)．本稿では，興奮波を発生させるためにパラメータとして$\alpha = 10^{-6}$, $\gamma = 10^{-1}$, $\epsilon = 10^{-1}$, $\tau = 10^{-3}$を用いた．また，連続系のの微分方程式をそのまま Octave で計算することはできないため．時間と空間に関して離散化したモデルを使って計算した [8]．

式 (1) を解くためには，毎回，隣接する地点の濃度を参照しなければならない．並列 Octave で計算する場合は，それぞれのステップごとに周囲の濃度を更新してしまうと計算時間全体に占める通信処理の比率が高くなってしまう．そこで，あらかじめ，冗長に計算区間を区切って通信回数を減少させた．

FitzHugh-南雲モデルによる興奮波伝搬を並列化したときの処理手順を Fig. 5に示す．まず，計

Fig. 4　FitzHugh-南雲モデルによる興奮波伝搬の様子 ((a) から (e) に進む)

```
1)  procedure FitzHugh-南雲モデルによる興奮波の伝搬
2)    begin
3)      for ステップ数 := 0 to 1000 do
4)      begin
5)        短冊状に分割した 2 次元の計算区間を
6)        各プロセスに対して送信する
7)        if 次の計算に濃度情報の更新が必要 then
8)        begin
9)          隣接する区間を持つプロセス間で
10)         必要な濃度情報を更新する
11)       end
12)       割り当てられた計算区間を計算する
13)     end
14)     分割した計算区間を統合する
15)   end.
```

Fig. 5　FitzHugh-南雲モデルによる興奮波伝搬を並列化したときのの処理手順

算区間をプロセス数だけ短冊状に分割し，さらに冗長に計算する濃度情報を付け加えて各プロセスに渡す．冗長に割り当てた計算区間の計算が終了すると，各プロセス間で計算に必要な濃度情報を更新するため，隣接する区間を持っているプロセスどうしで通信を行う．最後に各プロセスの計算結果を結合する．

以上の処理を PC クラスタと PrimePower 600 上で行った結果を Fig. 6と Fig. 7に示す．並列化を行わない場合は，PC クラスタで約 6246 秒，PrimePower 600 では約 6752 秒であった．処理を並列化した場合は，PC クラスタで 14 並列の時に約 814 秒，PrimePower 600 で 8 並列のときに約 1025 秒であった．また，Speed-up factor は，PC クラスタで 14 並列の時に約 7.6 倍，PrimePower 600 で 8 並列の時に約 6.7 倍であった．PC クラスタで 8 並列の時に通信時間が 73 秒であるのに対し，PrimePower 600 では，17 秒である．PrimePower 600 は通信によるオーバーヘッドが小さいため，台数が増加しても，Speed-up factor が飽和せずに台数効果に近い結果が得られていることがわかる．

3.3　位相限定相関法を用いた指紋照合の並列化

並列処理時にで通信を要しない例として位相限定相関法 (Phase-Only Correlation: POC) を用いた指紋照合を取り上げた．POC は，画像の周波数領域における位相情報を使った高性能画像照合技術である．

POC を使った指紋照合 [9]は，以下のような手順で行われる．

1) 登録画像と入力画像を読み込む
2) -40°から 40°まで登録画像を回転させる
3) 回転した登録画像と入力画像を POC で照合し，最もスコアの高かった画像から回転角度を決定する
4) 回転補正した登録画像と入力画像の共通部分を抽出する
5) POC で共通部分を照合する

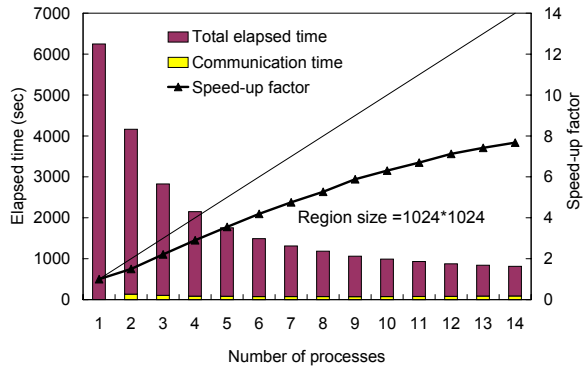

Fig. 6　興奮波伝搬のシミュレーションにおける経過時間と Speed-up factor (PC クラスタ)

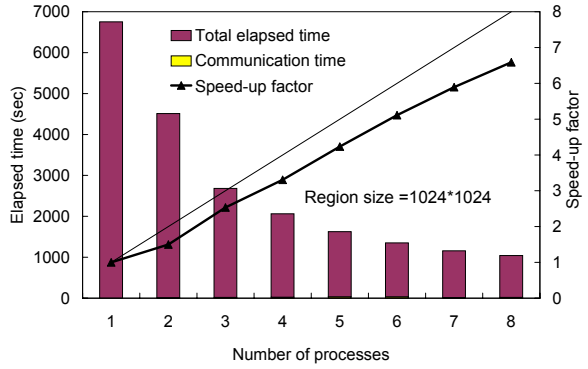

Fig. 7　興奮波伝搬のシミュレーションにおける経過時間と Speed-up factor (PrimePower 600)

POC を使った指紋照合を並列化したときの処理手順を Fig. 8に示す．POC を用いた指紋照合において，以下の部分が実行時間の大部分を占めている．

1) 登録画像の回転

2) 回転した登録画像と入力画像の照合

3) POC による共通部分の照合

これらの部分は，お互いのデータに依存せずに処理が繰り返し行われるため，容易に処理を分割することができる．ただし，3) は，2) の計算結果を利用するため，2) と 3) の間でプロセス全体の同期待ちが加わる．実験は，384 × 384 の 10 枚の指紋画像を用い，そのすべての組み合わせ (45 通り) に対して行った．

```
1)  procedure POC を用いた指紋照合
2)    begin
3)      回転角度 := -40 から 40 度まで 1 度刻み;
4)      begin
5)        for 全ての指紋画像ペアに対して do
6)        begin
7)          分担する回転角度の範囲を決定する
8)          for 与えられた回転角度に対して do
9)          begin
10)           回転させた登録画像と入力画像の
11)           相関スコアを POC で調べる
12)         end
13)       end
14)       各プロセス間で相関スコアを通信し，
15)       最も相関が高くなる回転角度を決定する
16)       照合処理を各プロセスに再配分する
17)       for 与えられた処理に対して do
18)       begin
19)         回転した登録画像と入力画像の共通部分を
20)         抽出した後に POC で照合スコアを算出する
21)       end
22)     end
23)   end.
```

Fig. 8　POC を用いた指紋照合を並列化したときの処理手順

PC クラスタと PrimePower 600 の実験結果をそれぞれ Fig. 9 と Fig. 10に示す．並列化を行わない場合は，PC クラスタで約 3691 秒，PrimePower 600 で約 8373 秒のであったのに対し，並列化した場合は，PC クラスタで 14 並列の時に約 330 秒，PrimePower 600 で 8 並列の時に約 1150 秒であった．Speed-up factor は，PC クラスタが 14 並列で 11.2 倍，PrimePower 600 が 8 並列で 7.2 倍であった．POC を用いた指紋照合では，それぞれのデータに対して処理の依存性がないため，PC クラスタでも PrimePower 600 でも同様にほぼ理想的な台数効果が得られた．

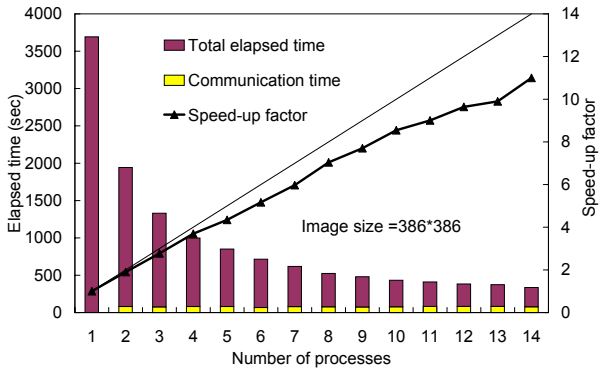

Fig. 9 指紋照合処理における経過時間と Speed-up factor (PC クラスタ)

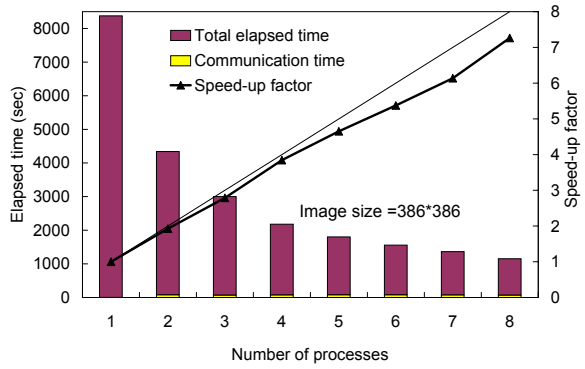

Fig. 10 指紋照合処理における経過時間と speed-up factor (PrimePower 600)

4. むすび

本稿では，GNU Octave を用いて並列信号処理環境を構築し，その性能評価を行った結果ついて述べた. 開発した並列 Octave は，並列信号処理アルゴリズムを記述・検証・評価することができる高水準な並列プログラミング環境として活用できると考えられる．今後は，並列計算用の関数群の追加や通信性能の向上などを行っていく予定である．

今回作成した並列 Octave のソースは以下の URL にて GPL の下に公開している．

`http://www.aoki.ecei.tohoku.ac.jp/octave/`

参考文献

1) GNU Octave Home Page
`http://www.octave.org/`.

2) Parallel Octave
`http://www.aoki.ecei.tohoku.ac.jp/octave/`.

3) Message Passing Interface (MPI) Forum
`http://www.mpi-forum.org/`.

4) PVM : Parallel Virtual Machine
`http://www.csm.ornl.gov/pvm/pvm_home.html`.

5) HPF: The High Performance Fortran Home Page
`http://crpc.rice.edu/HPFF/`.

6) OpenMP: Simple, Portable, Scalable SMP Programming
`http://www.openmp.org/`.

7) J. D. Murray, *Mathematical Biology*,Springer-Verlag, Berlin, 1993.

8) K. Ito, T. Aoki, and T. Higuchi,“Digital reaction-diffusion system —A foundation of bio-inspired texture image processing—,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E84-A, No. 8, pp. 1909 – 1918, August 2001.

9) K. Ito, H. Nakajima, K. Kobayashi, T. Aoki, and T. Higuchi,“A fingerprint matching algorithm using phase-only correlation,” *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, March 2004,(to be appeared).