Vol.2015-IOT-30 No.7
2015/7/4

安価なコンピュータを用いた実験・教育用並列計算機環境の構築

信田 圭哉[1,2]　長谷川 明生[1]

概要：技術者や学生が並列プログラムの実験や学習を行う際，スーパコンピュータなどの並列計算機を用いることは，金銭的なコストや管理コストの点から困難である．本研究では，安価な ARM ボードを用いたコンピュータクラスタを構築し，クラスタ構築の支援，MPI プログラムの動作実験支援を行うシステムを開発した．このシステムにより，課題となる金銭的コストと管理コストを低減し，実験や教育目的で容易に利用できる初学者向け環境の構築を目指す．

Construction of parallel computer for experiments and education using inexpensive computer

Nobuta Keiya[1,2]　Hasegawa Akiumi[1]

1. はじめに

単一のプロセッサの性能は物理的な限界によって高止まりしつつあり，マルチプロセッシングによる処理の高速化が一般的になっている．市販のパーソナルコンピュータ向けにマルチコアの CPU が流通する中，コンピュータサイエンスを学ぶ上で，マルチプロセッシングの知識や概念を習得することは重要である．

スーパコンピュータをはじめとした並列計算機は高価で希少であり，計算資源を活用できる立場の人間が限られている．そのため技術者や学生が並列計算の知識や技術を修得しようとしたとき，実際の環境を用いて並列プログラムの実験を行い，学習することは困難である．また，計算機環境を学習者自身が管理し実験等の操作を行うため，それらにかかるコストはできるだけ低減したいという要求がある．

本研究の目的は，MPI を用いた並列計算機を教育目的で利用する際に課題となる，金銭的コストや管理コストを低減し，初学者が容易に利用できる環境を構築することである．Cox らは，一般的なパーソナルコンピュータ（以下，PC）よりもはるかに安価な Raspberry Pi を使用し，低コストなコンピュータクラスタが構築できることを示した [1]．Cox らは，64 台の Raspberry Pi をネットワーク接続し，その上で MPI を使用した並列計算機を実現している．筆者らは，Cox らの手法に基づき 24 台の Raspberry Pi を用いてクラスタを構築したことを既に報告した [2]．その中で，クラスタを構成するマシンの IP アドレスを自動で収集する仕組みについて述べた．また，同じく ARM ボードである Pandaboard を使用した構成に変更したことを報告した [4]．本稿では，さらに 24 台の Raspberry Pi と，7 台の Pandaboard を追加し，合計 56 台の ARM ボードクラスタを構築したことを報告する．また，クラスタを構成するマシン情報の自動収集と，その情報の表示とクラスタに対する操作を行う Web インタフェース，クラスタ内部の通信と CPU 使用率を監視するモニタシステムについて述べる．

[1] 中京大学
Chukyo University
[2] 株式会社富士通コンピュータテクノロジーズ
Fujitsu Computer Technologies Limited

2. ARM ボードを使用した PC クラスタ

構築したクラスタの写真を図 1 に示す．Pandaboard(ES) と Raspberry Pi の 2 種類の ARM ボードを使用した．Pandaboard は 8 台，Raspberry Pi は 48 台使用し，合計で 56

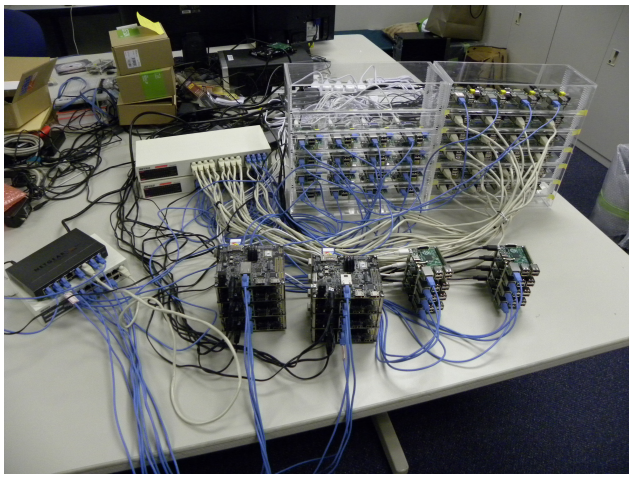

図 1　Pandaboard 8 台，Raspberry Pi 48 台を使用したクラスタ

台のクラスタを構築した．

Raspberry Pi を搭載するために，アクリル板を加工してラックを作成した．この上に，横 4 台，縦 6 段の計 24 台を搭載できる．ラックは 2 台作成し，40 台の Raspberry Pi Model B を搭載した．8 台の Pandaboard と，8 台の Raspberry Pi Model B+は，金属スペーサを使用して積み上げた．

2.1 Raspberry Pi

Raspberry Pi は，イギリスの Raspberry Pi Foundation が，コンピュータやプログラミングの教育向けに開発した，クレジットカード程度の大きさのコンピュータである．モニタや TV への映像出力端子を持ち，USB 接続のキーボードとマウスがあれば，家庭用の TV を使用したデスクトップパソコンとして使用できる．

Raspberry Pi は，ARM 向けにビルドされた汎用の Linux OS が動作し，市販の PC と同様に使える．Raspberry Pi はいくつかのモデルがあり，それぞれ性能やハードウェア形状，価格が異なる．本研究で使用したモデルは，Model B，Model B+である．2015 年現在，1 台 5000 円未満で入手可能であり，一般的なデスクトップ PC と比較すると遥かに安価である．また，消費電力は Model B で 3.5W，Model B+で 3.0W 程であり，電力の面でも低コストに抑えることが期待できる。Raspberry Pi の主な仕様を以下に示す．

- プロセッサ 700MHz，シングルコア ARM1176
- RAM 512MB
- Ethernet ポート 100 Base-T
- 電源電圧 5V

Raspberry Pi Model B+は，Model B の後継機種である．Model B からの変更点は，ストレージが SD カードから microSD カードに，USB ポート数が 2 から 4 に，GPIO ピン数が 26 から 40 に増加した点である．ハードウェア形状が異なるため，ボードを搭載するラックを作成する際には注意が必要になる．

Raspberry Pi 用の OS として，Raspbian wheezy を使用した．これは Debian ベースの OS であり，Raspberry Pi の web ページ [8] からダウンロードできる．Raspbian には，デスクトップ環境と，Scratch や Python のようなプログラミング言語が標準でインストールされている．

2.2 Pandaboard

Pandaboard には，動作周波数が 1.0GHz のプロセッサを搭載したモデルと，動作周波数が 1.2GHz の Pandaboad ES というモデルがある．本研究では Pandaboard ES を使用した．Pandaboard ES の主な仕様を以下に示す．

- プロセッサ 1.2GHz, デュアルコア ARM Coretex^{TM}A9
- RAM 1GB
- Ethernet ポート 100 Base-T
- 電源電圧 5V

Pandaboard ES の価格は 29,000 円程で，Raspberry Pi と比較すると高価ではあるが，それでも一般的な PC と比較すれば安価である．Raspberry Pi と同様に，ARM 向けにビルドされた Linux OS が動作する．本研究では，Ubuntu サーバ，12.04 ARM Hard-Float 版を使用した．Ubuntu の CD イメージのダウンロードページ [10] からダウンロードできる．

2.3 MPICH

MPI のライブラリ実装として MPICH を使用した．MPICH2（バージョン 1.4.1p1）を Web ページ [5] からダウンロードしビルドした．MPICH の configure オプションは，インストールディレクトリのみを指定した．

3. ネットワーク構成

ネットワーク構成の概要を図 2 に示す．クラスタを構成するのは Raspberry Pi 0 〜 47，及び Pandaboard 0 〜 7 である．これらを全て同じサブネットに接続する．Pandaboard 0 は MPI プログラムのマスタノードとして使用し，同じネットワークに接続した操作用 PC から SSH ログイン，もしくは本研究において試作した Web システムのインタフェースを通して操作を行う．

また，クラスタ内の通信を監視するために，ポートミラリング可能なスイッチングハブを用い，Pandaboard 0 を接続したポートのパケットを，監視用の PC を接続したポートに転送している．その他のマシンを接続するために，24 ポートのスイッチングハブ 2 台，16 ポートのものを 1 台使用した．

クラスタを構成するマシンは，基本的に固定 IP にはせず，DHCP により IP アドレスを割り当てるようにした．固定 IP アドレスを割り当てる場合，すべてのマシンのネットワークインタフェースに IP アドレスを設定する必要があり，これは管理コストの増大につながる．Pandaboard 0

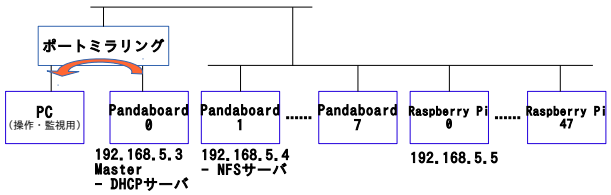

図 2　ネットワーク構成

は DHCP サーバを動作させ，例外的にネットワークインタフェースの設定により固定 IP アドレスにした．

外部のネットワークに本クラスタを接続する際は，ルータを必要とする．多くの家庭用ブロードバンドルータは DHCP サーバ機能を備えているが，同一のネットワークに DHCP サーバが同居すると正常に機能しないため，ルータの DHCP サーバ機能を停止しておく．

Pandaboard 1 と Raspberry Pi 0 は，DHCP サーバの設定により固定の IP アドレスを割当てている．Pandaboard 1 は，NFS サーバを運用するために IP アドレスを固定とした．Raspberry Pi 0 は，MPI プログラムのビルドを行うマシンとして利用するために，IP アドレスを固定とした．

Pandaboard 1 は，NFS サーバとした．その他のマシンでは，自身のファイルシステムに NFS サーバのディレクトリをマウントし，共有ファイルにアクセスする．MPI プログラムなど，クラスタ内の全てのマシンで共通して使用するファイルを共有することで，全てのマシンに同じファイルをコピーする必要がなくなる．

4. MPI プログラム実験支援システム

構築した ARM ボードクラスタを対象に，MPI プログラムの実験を支援するためのシステムについて述べる．

システムは，クラスタを構成するマシン情報の自動収集と，その情報の表示とクラスタに対する操作を行う Web インタフェース，クラスタ内部の通信及び CPU 使用率のモニタシステムで構成される．

4.1 クラスタ構成マシン情報の自動収集システム

MPI プログラムを実行する際，クラスタを構成するマシンのネットワークアドレスと，そのマシンのプロセッサに関する情報などを予め把握しておく必要がある．MPICH を使用する場合，使用する全てのマシンの IP アドレスと，そのマシンで使用するプロセッサの数を列挙したリストファイルが必要になる．このリストファイルのことをマシンファイルと呼ぶ．マシンファイルは 1 度作れば同じ環境を利用し続けられるが，クラスタにマシンを追加したときや，マシンの台数を変化させて実験を行いたい時，再度作成する必要がある．そこで，クラスタを構成するマシンの IP アドレスを収集し，各マシンのプロセッサ情報を取得して，マシンファイルを自動生成するシステムを作成した．

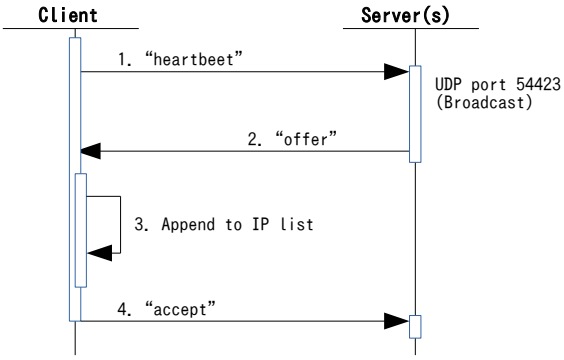

図 3　IP アドレスリスト作成処理の流れ

4.1.1 IP アドレスリストの作成

IP アドレスリストの作成は単純なハートビートの仕組みである．図 3 にその手順を示す．クラスタを構成する全てのマシン (サーバ) は，リストを作成する側 (クライアント) からのメッセージに返信する．クライアントは UDP のブロードキャストパケットによりメッセージを送信する．サーバからの返信の受信に成功したら，そのサーバの IP アドレスを保存し，ソートした上でファイルにリストとして出力する．

UDP ブロードキャストパケットを利用するため，クライアントはサーバと同一のネットワーク内に接続する．

IP アドレスリストは並列計算以外にも，MPI プログラムやその他のプログラムのインストール，再起動やシャットダウン等のコマンドの送信等，全てのマシンに対し同一の操作を行いたい場合に利用できる．

4.1.2 プロセッサ情報の取得

クラスタを用いた並列処理プログラムを実行する上で，使用するマシンのプロセッサの情報は不可欠である．プログラム実行者は，使用するすべてのマシンについて，プロセッサ情報を把握する必要がある．しかし，本研究で扱うクラスタでは，DHCP により IP アドレスを割り振っているため，任意のマシンを IP アドレスから特定することは困難であり，IP アドレスとマシンのプロセッサ情報を紐付けるシステムが必要になる．

CPU 情報の取得は，IP アドレスリストに登録されたマシンに対し TCP 接続し行う．まずクライアントは，接続したサーバに対しメッセージを送信する．次に，サーバは自身のプロセッサ情報を読み込み，JSON 形式の文字列にして返信する．

送信する JSON は次のような形式になっている．

```
{"CPU-name":NAME, "CPU-num": N}
```

上記の NAME はプロセッサの型名の文字列が入り，N はプロセッサの個数が入る．将来的に，取得できる情報を増やすことを想定し，今回 JSON 形式での返信を行う仕様とした．情報量の増加に対応しやすくするために，UDP で

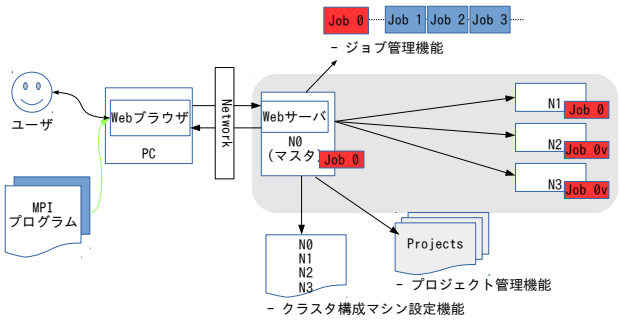

図 4　web システム概要

はなく TCP で通信を行うようにした．

4.1.3　マシンファイルの生成

マシンファイルは以下のような形式になっている．

```
192.168.5.3:2
192.168.5.4:2
192.168.5.5:1
```

各行の「:」で区切られた左側が IP アドレス，右側がプロセッサ数である．これを MPI プログラムの実行コマンドである「mpiexec」の引数として渡すことで，リストで指定したマシンの，指定したプロセッサ数を用いてプログラムを実行できる．

4.2　Web システム

クラスタに対する操作を行うための Web システムについて述べる．Web システムの概要を図 4 に示す． Web システムは，クラスタ上で動作する MPI プログラムのビルドや，ジョブ投入を簡略化するための Web インタフェースをユーザに対し提供する．ユーザは自身の PC の Web ブラウザから本システムを利用することで，クラスタシステムに対する統合的な操作を実現できる．

今回試作した Web システムで扱う主な機能は以下の通りである．

(1) クラスタ構成マシン設定機能
  (a) 4.1 節で解説した，クラスタ構成マシンの IP アドレスのリストと，各マシンのプロセッサ情報の表示機能
  (b) 上記リストから使用するマシンをユーザが選択し，マシンファイルを生成する機能

(2) プロジェクト管理機能
  (a) プロジェクトの作成
  (b) ファイルのアップロード機能
  (c) テキストファイル編集・削除機能
  (d) MPI プログラムのビルド機能
  (e) マシンファイル設定機能

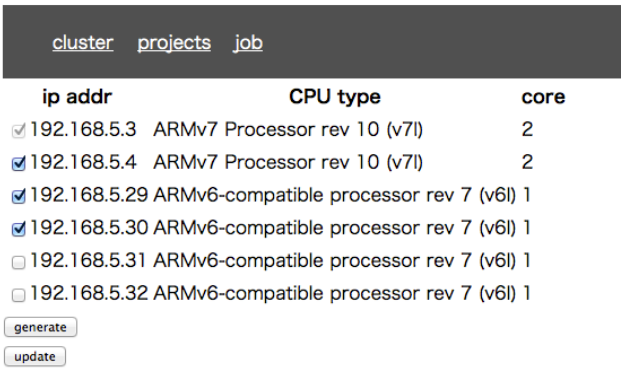

図 5　クラスタ構成の設定画面

(3) MPI プログラムのジョブ管理機能

以降，システム構成と各機能について概要を説明していく．

4.2.1　Web システムの構成

Web サーバをマスタマシン上で動作させ，ユーザは自身の PC 上の Web ブラウザからマスタマシンにアクセスし操作を行う．システムの実装には，Python に標準で付属する Web Server Gateway Interface(WSGI) のモジュールを使用した．

4.2.2　クラスタ構成マシン設定機能

クラスタ構成マシン設定機能は，クラスタを構成するマシンの IP アドレスとプロセッサ情報を表示し，マシンファイルを生成するための機能である．クラスタ構成の設定画面を図 5 に示す．リストには，各マシンの IP アドレス，CPU の型，各マシンのプロセッサ数が表示されている．ユーザは表示されたリストから使用するマシンを選択し，マシンファイルを生成できる．

4.2.3　プロジェクト管理機能

ファイルはプロジェクトという単位で管理する．これは単純に，ファイルの集まりをディレクトリとして分けるというだけであるが，今後 Git 等のバージョン管理システムと連携することを想定し，このような形態をとっている．プロジェクトを作成し，作成したプロジェクトにファイルを追加 (アップロード)，編集等を行える．追加したファイルはプロジェクト管理画面にリンクとして表示される．追加したファイルが C ソースコードであれば，MPI のビルドコマンドである「mpicc」を使用してコンパイルできる．

図 6 に，MPICH に付属するテスト用の C ソースコードをアップロードし，編集画面を表示した様子を示す．

4.2.4　MPI プログラムのジョブ管理機能

MPI プログラムのジョブ管理機能について解説する．MPI プログラムを初めとする並列計算プログラムは，システムの性能を最大限に引き出すことを要求される．少なくとも，プログラマはそれを望んでいるはずである．そのため，汎用 OS のようなタイムシェアリングシステムでは，可能な限り同時に実行するプロセスを減らしておきたい．

MPI プログラムのジョブ管理機能は，複数の MPI プロ

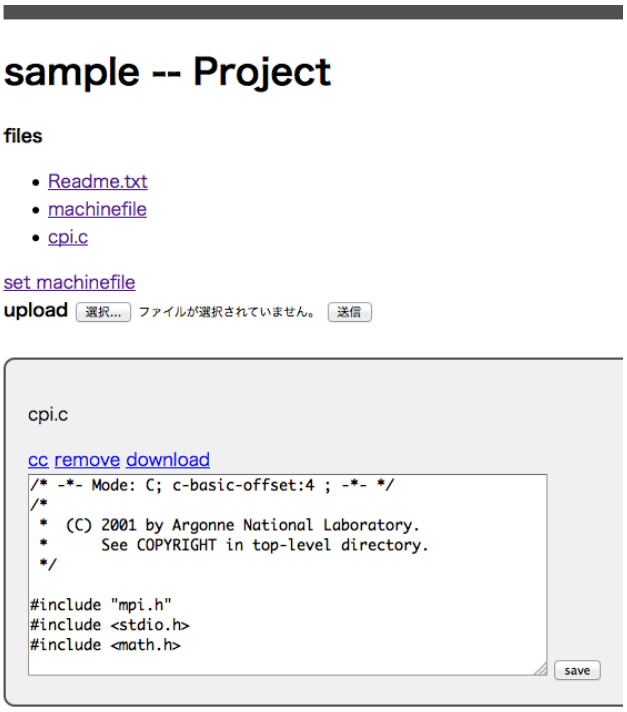

図 6　アップロードしたファイルの編集画面

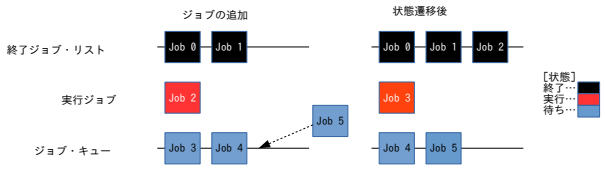

図 7　ジョブ・キュー

グラムのジョブを同時に実行しないよう，待ち行列を用いて管理する．ユーザインタフェースでは投入されたジョブのリストを表示し，ジョブの状態と，実行開始時間・終了時間の情報，実行結果へのリンクを表示する．

ジョブの待ち行列 (ジョブ・キュー)

ジョブは投入された順番でジョブ・キューに格納される．本システムでは，ジョブは待ち状態，実行状態，終了状態の 3 つの状態を持つ．図 7 に，ジョブの状態とジョブ・キューの関係を示す．

実行中のジョブは，何も無いか，もしくは 1 つのみである．図の左側はジョブを投入する様子を示している．新たに投入されたジョブは，ジョブ・キューの最後尾に格納される．実行中のジョブは終了すると終了リストに格納され，終了状態に遷移する．次に，ジョブ・キューの先頭から 1 つジョブが取り出され，実行状態に遷移する．図右側は各状態遷移後の様子を示している．

ジョブ管理機能のユーザインタフェース

MPI プログラムをビルドし，出力された実行形式ファイルを実行する際の表示例を示す．まず，プログラム実行直後の画面を図 8 に示す．このとき，ジョブの投入時間

図 8　ジョブ・リストの表示例

図 9　実行済みのジョブ表示例

```
Process 0 of 8 is on panda-master
Process 1 of 8 is on panda-master
Process 6 of 8 is on raspberrypi
Process 5 of 8 is on raspberrypi
Process 7 of 8 is on raspberrypi
Process 2 of 8 is on panda-nfs
Process 4 of 8 is on raspberrypi
Process 3 of 8 is on panda-nfs
pi is approximately 3.1415926544231247, Error is 0.0000000008333316
wall clock time = 0.015351
```

図 10　プログラム実行結果ログファイルの表示例

(ジョブがジョブ・キューに追加された時間) が表示される．ジョブの状態は w，e，d のいずれかで表示され，それぞれ待ち状態，実行状態，終了状態を意味している．

次に，プログラムが終了した後の画面を図 9 に示す．このとき，プログラムの開始時間，終了時間と，プログラムの実行結果が格納されるログファイルへのリンクが表示される．ログファイルの例を図 10 に示す．

5. 通信及び CPU 使用率のモニタシステム

本研究に関連するシステムとして，2013 年度に林が「並列計算効率化のための通信モニターシステム」[3](以降，モニタシステムと呼ぶ) を開発した．モニタシステムは，クラスタを構成する各マシン間の通信と，各マシンの CPU 使用率を可視化するシステムである．プログラマはこのモニタシステムを使うことで，自身の書いた並列プログラムの通信と処理を視覚的に解析できる．

並列計算プログラムは，すべてのプロセスに処理を均等に分割することで処理能力を高めることができる．プロセス間の処理や通信に偏りがあると本来の性能を発揮できない．例えば，あるマシンのプロセッサが高負荷の処理を行い，他のプロセッサが休んでいるというような状態は偏りがある．また，特定のプロセス間でのみ通信を行うような状態は通信の偏りがある．

プログラマにとって，自身の書いた並列プログラムの，実行時の CPU 付加や通信の偏りを発見することは，より効率的なプログラムを開発を行う上で重要となる．並列プログラムの偏りを発見するために，実行するプログラムがどのようにプロセッサ間に分散されたか，また通信状況がどのようになっていたかを確認するツールがあると良い．

MPI プログラムの通信ロギングツールとして，MPI Parallel Environment(MPE)[7] が挙げられる．MPE は，MPI

プログラムのビルド時に静的にログ取得用コードを埋め込み，プログラム実行後にログの解析・可視化を行うためのツール群である．MPE は，MPI プログラムを実行する全てのプロセス間の通信を解析できるため，作成した MPI プログラムの偏りを確実に判断できる．

しかし，プログラム開発途中のデバッグ段階では，偏りをプログラム実行後に手順を踏んで確認するよりも，実行中に確認できる方が効率が良い．プログラムの実行中にリアルタイムに偏りを可視化するツールにより，開発効率を向上させることが期待できる．

モニタシステムは，マシン間の通信量と，それぞれのマシンのもつプロセッサの使用率を，リアルタイムでグラフィカルに表示する．これにより並列プログラムの偏りを，並列プログラムの動作中に一見して判断することが可能になる．

5.1 モニタシステムの構成

ユーザは，ユーザの PC 上で動作する Python で書かれたプログラムを通して本システムを操作する．このプログラムは，IP パケットをキャプチャする機能と，各マシンの CPU 使用率を算出する機能を持っている．

クラスタを構成するマシンは，ユーザの PC 上のプログラムから UDP ブロードキャストされたメッセージに応じて，自身の CPU 使用率の算出に必要な情報を返信する．ユーザ PC のプログラムは，全てのマシンから返信を受けとり，それぞれについて CPU 使用率を算出し，ディスプレイに表示する．

5.1.1 IP パケットキャプチャ機能

クラスタ内のネットワーク内の通信を監視するために，このプログラムは IP パケットをキャプチャする．通常のスイッチングハブを用いると，操作用 PC を接続したポートへのパケットだけしかキャプチャできない．そのため，ポートミラーリング可能なスイッチングハブを使用している．

この手法の問題点は，ネットワーク内の全ての通信を監視できるわけではなく，マスタノードのマシンとその他のマシンとの通信しか監視できない点である．すべての通信をリアルタイムで把握するには，上述した「MPE」等を用いて MPI プログラムに静的にログ取得コードを埋め込むか，すべてのポートをミラリングし監視するなどの方法を用いる必要がある．

5.1.2 モニタシステムのユーザインタフェース

ユーザインタフェースを図 11 に示す．図の上部左側に並ぶ四角形は，クラスタを構成しているマシンを表している．左上隅がマスタノードとなるマシンを表している．今回，林のシステムから一部改良を行った．表示するマシンの数が固定だったものを，4.1 節で解説したシステムを用いて，クラスタを構成するマシンの台数に合わせて表示できるようにした．

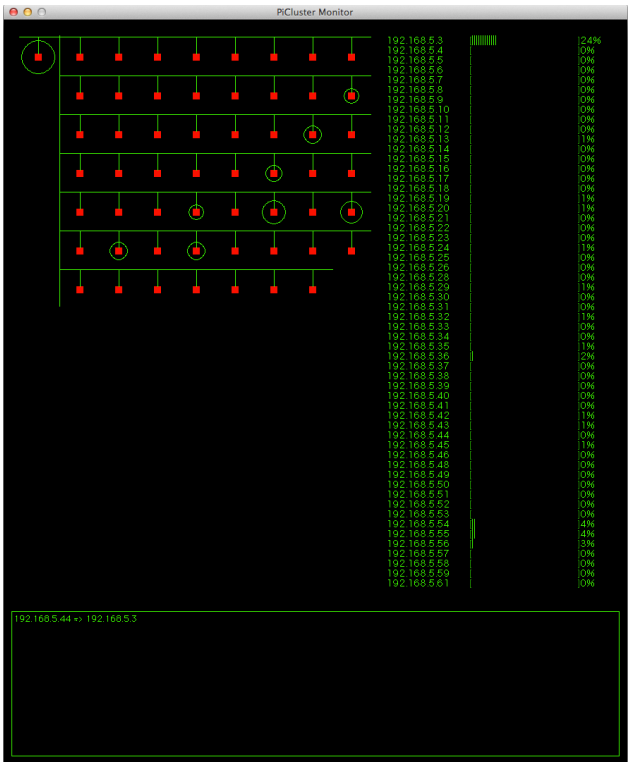

図 11　並列計算モニタシステムのユーザインタフェース

この四角形をマウスでクリックすると下側の枠内に，そのマシンがどこと通信しているかを，それらの IP アドレスと通信方向を表す矢印で表示する．

クラスタ内のあるマシンが別のマシンと通信を行うと，それぞれ対応する四角形を中心に線で円を描く．この円の大きさは通信量を表し，一定時間あたりの通信量が多ければ円が大きく，通信量が少なければ円が小さく描かれる．

右側には CPU 使用率を表示している．使用率は，バーの長さと数値で表示し，全てのマシンの CPU 使用率を確認することができる．

6. おわりに

6.1 評価と考察

本研究では，2 種類の ARM ボードを用いたコンピュータクラスタを構築し，そのクラスタを対象としたシステムの構築を行った．

まず，金銭的コストを低減させるために，Raspberry Pi と Pandaboard という 2 種類の ARM ボードを用いて，コンピュータクラスタを構築した．使用した ARM ボードは市販の PC と比較すると非常に安価であるため，これらを使用することで金銭的コストを低減できた．

次に，クラスタを扱う上で生じる管理コストの低減を目的とし，MPI プログラムの実験を支援するシステムを作成した．システムは，クラスタ構成マシン情報の自動収集システム，Web システム，モニタシステムで構成される．

クラスタ構成マシン情報の自動収集システムでは，クラスタを構成するマシンの，ネットワークアドレスとプロセッサ情報をリストアップする仕組みを用意し，これらの収集に掛かるコストを軽減した．モニタシステムでは，クラスタ内の通信や CPU の使用率を可視化することで，MPI プログラムのデバッグ等の作業を補助し，開発効率の向上につなげられることを示した．Web システムでは，クラスタ構成マシン情報を自動で収集し，MPI プログラムのビルドと実行を補助するシステムを，ユーザが Web インタフェースを通して統合的に操作できる仕組みを試作した．この Web システムにより，クラスタ操作にかかる管理コストを低減した．

また，Web システムを実際に学生に利用してもらい，感想という形で以下のような評価を得た．

- 操作は，扱いに慣れた人の指導の元では可能
- 動作が遅い (レスポンスが遅い)

操作に熟練者を必要としている点で，初学者向けという目標は達成できていない．

6.2 課題と今後の展望

課題として，本研究で構築したシステムの改善点，追加機能の案を挙げる．

まず改善すべき点は，システムの利便性を高めることである．操作の容易さを高めるために，WeB システムの機能，表示内容やレイアウト等はさらに検討が必要である．これはさらに多くの人に利用してもらい，フィードバックを得る必要がある．

また，動作の遅さについて，原因を特定する必要がある．ハードウェアが原因の場合，Web サーバマシンのみをより高性能なマシンに切り替える等の措置が必要になる．

本研究で試作した Web システムと，モニタシステムのユーザインタフェースは統合されていない．そのため Web ブラウザ上で，リアルタイムに MPI プログラムの動作を解析するまでには至っていない．統合されたユーザインタフェースは，システムの利便性を高める上で重要になる．Web ブラウザ上でリアルタイムに情報を表示するには，Web ブラウザと Web サーバが密に通信を行う必要があるため，Javascript や Flash 等を使用した，高度なアプリケーションを開発する必要がある．

追加する機能の案として，MPE を用いた静的解析ツールの利用，TOP100 で利用されている Linpack ベンチマークを取る機能を追加することを挙げる．静的解析のツールである MPE を用いて出力される情報は，MPI プログラム開発者にとって有用であるため，Web システムで利用したい．

7. まとめ

単一のプロセッサによる性能向上が見込まれなくなり，マルチプロセッシングを用いることが一般的になっている．本研究では，MPI を用いた並列計算機を教育用途で利用することを想定し，課題点の整理と課題解決の手法を示した．

まず，実験や教育用途で並列計算機を利用するには，金銭的コスト，管理コストが課題となり，これらを低減する必要があることを述べた．

金銭的コストを低減することを目的として，Raspberry Pi をはじめとした安価な ARM ボードを用いて，コンピュータクラスタを構築した．

次に，クラスタを扱う上で生じる管理コストの低減を目的とし，クラスタを構成するマシンの情報を自動で収集するシステム，クラスタに対する操作を簡素化するための Web システム，クラスタ内部の通信と処理を可視化するシステムについて，それぞれ解説した．今後の課題として，より多くの利用評価を得た上で，更なる機能改善が必要であることを述べた．

参考文献

[1] Simon J. Cox, James T. Cox, Richard P. Boardman, Steven J. Johnston, Mark Scott, Neil S. O'Brien: Iridis-pi: a low-cost, compact demonstration cluster, Cluster Computing, DOI, 10.1007/s10586-013-0282-7, 2013

[2] 信田 圭哉, 長谷川 明生: 安価なコンピュータを用いた実験・教育用並列計算機環境の構築, 研究報告 インターネットと運用技術, Vol2014-IOT-24 No. 17, 2014

[3] 林瑶太: 並列計算効率化のための通信モニターシステムの開発, 中京大学情報理工学部情報システム工学科卒業論文, 2014

[4] 信田 圭哉, 長谷川 明生: 2 種類の ARM ボードを用いたコンピュータクラスタの構築と並列計算実験支援システムの検討, 平成 26 年度 電気・電子・情報関係学会東海支部連合大会講演論文, 2014

[5] MPICH: http://www.mpich.org/static/tarballs/1.4.1p1/

[6] HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers http://www.netlib.org/benchmark/hpl/

[7] Performance Visualization for Parallel Programs: http://www.mcs.anl.gov/research/projects/perfvis/

[8] Raspberry Pi: http://www.raspberrypi.org/downloads

[9] Pandaboard: http://pandaboard.org/

[10] ubuntu 12.04 ダウンロードページ: http://cdimage.ubuntu.com/releases/12.04/release/