コンピュータ・クラスタ「並列統合システム」の開発

辻　修平

川崎医科大学　自然科学教室

（平成30年１月５日受理）

Development of computer cluster “Parallel integrated system”

Shuhei TSUJI

Department of Natural Sciences, Kawasaki Medical School

(Accepted on January 5, 2018)

概　　要

昨今，プリント基板の上に，必要なものに絞ったCPUと周辺部品，入出力インタフェースとコネクタを付けただけのコンピュータ，いわゆるシングルボードコンピュータが注目されている。

今回，Banana PiとRaspberry Pi合わせて24台組み合わせてコンピュータ・クラスタ「並列統合システム」を製作し，実際にモンテカルロ・シミュレーションを走らせた。この並列統合システムの具体的製作方法と計算能力について述べる。

キーワード：シングルボードコンピュータ，コンピュータ・クラスタ，MPI，Raspberry Pi，Banana Pi

Abstract

In recent years, attention has been paid to a so-called single board computer built on a single circuit board, with microprocessor(s), memory, input/output (I/O) and other features required of a functional computer. This time, I combined 23 Banana Pi and a Raspberry Pi together to create a computer cluster “the parallel integrated system” and actually run Monte Carlo simulations. The specific production method and computational capacity of this parallel integrated system will be described.

Key words: single-board computer, computer cluster, MPI, Raspberry Pi,Banana Pi

1　序論

プリント基板の上に，必要なものに絞ったCPUと周辺部品，入出力インタフェースとコネクタを付けただけのコンピュータ，これをシングルボードコンピュータと呼ぶ。このシングルボードコンピュータは，様々なものが売られている。この中で，もっともメジャーなものがRaspberry Piと呼ばれるイギリスのラズベリーパイ財団によって開発されたものである。Raspberry Piの初期のものを複数繋いでコン

Kawasaki Ikaishi Arts & Sci (43)：37−48 (2017)
Correspondence to Shuhei TSUJI
Department of Natural Sciences, Kawasaki Medical School,
577 Matsushima, Kurashiki, 701-0192, Japan
Phone：81 86 462 1111
F A X：81 86 462 1199
E-mail：tsuji@med.kawasaki-m.ac.jp

ピュータ・クラスタを製作した試みがなされている[1,2]。しかしながら，初期のRaspberry Piは，CPUに1コアしかなく，しかもCPUのクロック周波数は700MHzであった。よって，これらを複数台組んだコンピュータ・クラスタは「学習用スパコンをつくる」という目的のもとで作成され，「実際，これを用いて計算させる」といったものではなかった。

今現在のRaspberry Piの最新モデルは，Raspberry Pi 3 Model Bで，このスペックは，CPUがARM Cortex-A53という4コア，クロック周波数1.2GHz，メモリが1.0GHzである。Raspberry Pi 以外にも様々なシングルボードコンピュータが発売されている。その中でもBanana Pi BPI-M3は，8コアのA83T ARM Cortex-A7というCPU，メモリ2GB，ストレージeMMC8GBを搭載しており，現時点のRaspberry Pi 3 Model Bよりもかなりハイスペックな部品で構成されている。また，これらシングルボードコンピュータに搭載されているCPUは，ARM coreであり，スマートフォン，タブレット等で採用されているものである。現在のパソコンはIntel系のCPUで構成されており，ARM coreと比較すると消費電力は大きい。よって，Intel系CPUを使わずに，このシングルボードコンピュータを使ってコンピュータ・クラスタを作るメリットは，消費電力を抑えることができるということである。

今回，計算用にBanana Pi BPI-M3を23台，通信用にRaspberry Pi 3 Model Bを1台組み合わせ，コンピュータ・クラスタ「並列統合システム（Parallel Integrated system:以下PIsと呼ぶこととする。）」を製作した。このPIs製作において，具体的設定方法（ソフトウェアの導入）と消費電力等について述べ，計算能力については比較検討を行った。

2　シングルボードコンピュータのセットアップ

2.1　初期設定，アップデート，アップグレード等

Banana Pi，Raspberry PiにはLinuxのディストリビューションがイメージとして準備されている。Banana Pi BPI-M3には，Ubuntu 16.04 mate[3]のイメージを，Raspberry Pi 3 Model Bには，RASPBIAN STRETCH WITH DESKTOP [4]のイメージをダウンロードし，microSDカードに焼き付けて使用する。ネットがつながった状態から，ターミナルを立ち上げ，以下のコマンドを実行する。

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo rpi-update（Banana Piには必要ない。）
```

これにより，ソフトウェア等は最新版となる[5]。

2.2　gfortranの導入

イメージをサイトからダウンロード，焼き付けした時点でBanana Pi にはgcc 5.4.0が，Raspberry Pi には gcc 6.3.0が開発環境として入っている。モンテカルロ・シミュレーションコードEGS5[6]を使うためには，Fortranが必要であり，これは，新たに入れなければならない。以下のコマンドを実行することにより，gfortranをインストールできる。

```
$ sudo apt-get gfortran
```

尚，Raspberry Piの場合，GUI環境のAdd/Remove Softwareからもインストールすることができる。Banana Piには，gfortan 5.4.0が，Raspberry Piには，gfortran 6.3.0がインストールされる。この時点で，EGS5をインストールおよび設定すれば，EGS5をシングルボードコンピュータで走らせることができる[7]。

2.3　MPIの導入

Raspberry Pi，Banana Piでは，並列計算用アプリケーションMPI(Message Passing Interface)を走らせることができる。2017年現在では，MPICHのバージョンは3.2で，ソースコードmpich-3.2.tar.gzを取ってきて展開する[8]。以下のコマンドにより，シングルボードコンピュータにインストールする。

```
$ configure --prefix=/usr/local/mpich-3.2
$ make
$ sudo make install
```

また，使うためには，次のようにパスを通す必要がある。

```
export PATH=...../usr/local/mpich-3.2/bin:$PATH
```

3　PIsの設定，組み立て

今回の並列統合システムPIsは，外部のネットワークにつなぐための1台のRaspberry Pi，計算用の23台のBanana Piとして組んだ。具体的設定を以下に述べる。

3.1　ネットワークの設定

Banana Piの各ネットワークの内部ネットワークには,192.168.240.100から192.168.240.122までのIPアドレスを割り振る。またホスト名をbananapi00からbananapi22とした。これらは有線LANコネクタからハブを通じてお互い通信させる。具体的には，以下のように設定する。(bananapi00の設定)

IPアドレスの設定(/etc/NetworkManager/system-connections/Wired connection 1)

```
.................
[ipv4]
address1=192.168.240.100/24,192.168.240.254
dns=8.8.8.8;
dns-search=
methods=
.................
```

ホスト名の設定(/etc/hostname)

```
bananapi00
```

内部ネットワーク内のホスト名の設定(/etc/hosts)

```
.................
192.168.240.100      bananapi00
192.168.240.101      bananapi01
.................
192.168.240.121      bananapi21
192.168.240.122      bananapi22
192.168.240.124      raspberrypi24
```

現時点のBanana Pi BPI-M3のWifiは技適マークがついておらず，日本で無線LANは使えない。そのため，外部との情報のやり取りにはRaspberry Piを使った。Raspberry Piでは，有線LANを使った内部ネットワークと，無線LANを使った外部ネットワークの２つを設定する。RASPBIAN STRETCH WITH DESKTOPのネットワーク設定は，/etc/dhcpcd.confで行う。

Raspberry Piでのネットワークの設定(/etc/dhcpcd.conf)

```
.................
interface enxb***************
static ip_address=192.168.240.124
static routers=192.168.240.254
static domain_name_servers=8.8.8.8
static domain_search=
```

```
interface wlan0
static ip_address=???.???.???.???
static routers=???.???.???.???
static domain_name_servers=???.???.???.???
static domain_search=
```

ホスト名の設定(/etc/hostname)

```
raspberrypi24
```

内部ネットワーク内のホスト名の設定は，/etc/hostsに前述したものと同じものを置く。内部ネットワークのアドレスの設定を表１にまとめておく。

表１：PIsの内部ネットワークのアドレス

IPアドレス	bananapi00:	192.169.240.100
	bananapi01:	192.168.240.101
	⁞	
	bananapi22:	192.168.240.122
	raspberrypi24:	192.168.240.124
ネットマスク		255.255.255.0
ゲートウェイ		192.168.240.254
dnsネームサーバー		8.8.8.8

3.2 イメージのコピー

ホスト名bananapi00とraspberrypi24は，GUIで動かす。全てのノードに対し，自動ログインを禁止にする。raspberrypi24は，GUI環境から設定する。bananapiについては，以下のファイルのオートログインの項目をコメントアウトしすることで，自動ログインを禁止することができる。

オートログインの禁止(/etc/lightdm/lightdm.conf)

```
# autoligin-user=pi(#をつけてコメントアウトする)
```

bananapi00とraspberrypi24については，ストレージにmicroSDカードを使い，これからOSを起動している。これらは，ネットワークからbananapi00には，Ubuntu 16.04 mate，raspberrypi24には，RASPBIAN STRETCH WITH DESKTOPのイメージをダウンロードし，microSDカードに焼き付けて使う。各々の設定は，OS起動後行う。特にbananapi00は，MPIを行うので，開発環境であるgcc，gfortran，およびMPICHのインストールは必須である。Banana Pi BPI-M3には，8GBのeMMCのストレージがついているので，bananapi01からbananapi22には，このeMMCにOSのイメージを焼き付ける。まず，microSDカードで，OSを起動後，このOS上で，Banana Piのイメージ(Ubuntu 16.04 mate)をダウンロードする。以下のコマンドでeMMCにイメージを焼きこむことができる。

```
$ sudo dd if=????-ubuntu-16.4-mate-????.img of=/dev/mmcblk1 bs=10MB
```

イメージを焼き付けたのち，GPU環境から System → Control Center → GPartedを起動する。

/dev/mmcblk0p2を選択し，Resize/Moveにより，最大限まで拡張する。この後，シャットダウンし，microSDカードを抜いてから起動する。このときは，eMMCから起動されているので，この起動のときにほとんど全ての環境を整える。最低限，MPIに必要なgcc，gfortran，MPICH等の開発環境を整えておく。さらには後述するNFS，ssh等の設定も全て行う。また，オートログインの禁止も設定しておく。敢えて，GUI環境を選択する必要もなく，CLI環境にするならば，以下の命令をしておく。

```
$ sudo systemctl set-default multi-user.target
```

全ての環境を整えたのち，シャットダウンを行い，再度microSDカードで起動する。今回の起

動では，以下の命令で，eMMCのイメージをmicroSDカードに書き込む。

```
$ sudo dd if=/dev/mmcblk1 of=./Banana.img
```

このmicroSDカードを元にbananapi01からbananapi22まで，Banana.imgをeMMCに一気に書き込んでいく。尚，個々のIPアドレス，ホスト名は各々編集する必要がある。

3.3　NFSの導入と設定

複数のノードでMPIを実行する場合，NFS(Network File System)とssh(Secure Shell)の導入と設定を行う必要がある。ここではNFSの導入および設定について述べる[9]。マスタ・ノードをbananapi00とし，スレーブ・ノードをbananapi01～bananapi22，およびraspberrypi24とする。尚，ユーザー名(アカウント名)は共通にする必要があり，これを“pi”とする。

マスタ・ノード:bananapi00への導入と設定

NFSの導入および設定は，スーパーユーザーモードより，以下のコマンドを実行する。

```
# apt-get install nfs-kernel-server
# apt-get install nfs-common portmap rpcbind
```

これらは，既に入っていたり，または，Banana Pi用のUbuntuには準備されていなかったりするが，一応，全てインストールのコマンドを実行しても問題はない。インストール後，以下のコマンドを実行することで，Linux上でこれらのデーモンが動き出す。

```
# service rpcbind start
# service nfs-common start
# service nfs-kernel-server start
```

尚，OS起動時に自動的に起動するには，以下のコマンドを実行する。

```
# update-rc.d nfs-kernel-server enable
# update-rc.d nfs-common enable
# update-rc.d rpcbind enable
```

/etc/exportfsを以下のように追加し編集する。

```
exportfsの編集(/etc/exportfs)
.................
/home    192.168.240.0/255.255.255.0(rw,
sync,no_root_squash,no_subtree_check)
.................
```

以下のコマンドを実行し，/mntの階層下にpiが見えれば成功である。

```
# exportfs -av
# mount bananapi00:/home/mnt
```

スレーブ・ノード:bananapi01～bananapi22，およびraspberrypi24への導入と設定

まず，NFS関連のソフトウェアをダウンロードし，インストール，デーモンとして，動かす。

```
# apt-get install nfs-common rpcbind
# service rpcbind start
# service nfs-common start
# update-rc.d rpcbind enable
# update-rc.d   nfs-common enable
```

次に/etc/fstabに以下のものを追加し編集する。

```
fstabの編集(/etc/fstab)
.................
bananapi00:/home     /mnt/pi_home
nfs noatime 0 0
.................
```

トラブル後の処理，またはraspberrypi24の場合，ネット上からシャットダウンを行うことができるようにするため，もう一つ別のアカウント“pi1”を増やす。

```
# adduser pi1
```

スーパーユーザーモードで以下のコマンドを実行する。

```
# mv /home/pi /home/pi.tmp
# ln -s /mnt/pi_home/pi /home/pi
```

設定後，リブートし，スレーブ・ノードから/home/piにマスタ・ノードと同じものがあれば成功である。

3.4 sshの設定

シングルボードコンピュータにLinuxを入れた時点で，既にsshは導入されている。MPIを複数ノードで使う場合，ホスト-スレーブ間をパスワードなしでログインできる必要がある。マスタ・ノード(bananapi00)を以下のように設定する。

```
$ ssh-keygen -t rsa
$ cat /home/pi/.ssh/id_rsa.pub >> /home/pi/.ssh/authorized_keys
$ chmod 600 /home/pi/.ssh/authorized_keys
```

これで，公開鍵，秘密鍵全てが各ノードへ配布される。(/home/piが共通なため。) また，/etc/ssh/sshd_configを以下のように編集する。

```
sshd_configの編集(/etc/ssh/sshd_config)
.................
AuthorizedkeysFile %h/.ssh/authorized_keys
.................
PasswordAuthentication yes
.................
```

リブートまたは，sudo /etc/init.d/ssh restartを実行後，

```
% ssh pi@bananapi??(??には01から22を入れる)
```

または，

```
% ssh pi@raspberrypi24
```

として，パスワードの要求なしにログインできれば成功である。

3.5 MPI用ホストファイルの設定

/home/pi/mpihosts/hostsに以下の内容を記述する。

```
bananapi00
bananapi01
bananapi03
.........
bananapi22
```

尚，以下のように記述すると各ノードで使用するCPUコアを制限できる。

```
bananapi00:6
bananapi01:8
bananapi03:3
.........
bananapi22:5
```

また.bashrcに次の一行を記述する。

```
export HYDRA_HOST_FILE=~/mpihosts/hosts
```

これで，MPIを実行する環境を全て整えることができる。あとは，ホストであるbananapi00から，mpicc，mpifort等のコンパイラでコンパイルした後，

```
mpirun -np 184 execute_file
```

で複数ノードをまたがるMPIを実行できる。尚，184という数値は，MPIで使うbananapi00～bananapi22までの23ノード×8コアを意味する。

3.6　ntpの設定

外部につながっているノードはraspberrypi24なのでこのノードをタイムサーバーとして，ba-nanapi00～bananapi22までの時刻の同期を行う。raspberrypi24以外のbananapiの/etc/ntp.confに次の一行を加え，時刻の同期のために他のサーバーにアクセスする項目は，削除またはコメントアウトして編集する。

```
server 192.168.240.124  iburst
```

PIs起動後，しばらくすると，システム全体は，時刻を同期するが，今すぐ同期したいときは以下のコマンドを実行する。

```
$ sudo service ntp restart
```

3.7　PIsのシャットダウン

PIs全体をシャットダウンさせるスクリプトを作る必要がある。まず各ノードで次のコマンドを実行する。

```
$ sudo visudo
```

次に以下のように編集する[10]。

```
pi[tab]ALL=NOPASSWD:[tab]/sbin/shutdown
```

bananapi00にshutdown.shというシェルスクリプトファイルを作り，次のように記述する。

```
#!/bin/bash
ssh pi@bananapi01  “sudo shutdown -h now”;
ssh pi@bananapi02  “sudo shutdown -h now”;
.....................................
ssh pi@bananapi22  “sudo shutdown -h now”;
ssh pi@raspberrypi24 “sudo shutdown -h now”;
sleep 30;
shutdown -h now
```

また，sshなどで外部ネットワークからこのPIsに接続し，シャットダウンさせる場合，raspberrypi24のpi1アカウントからシャットダウンさせる必要がある。raspberrypi24のpi1にshutdown_ras.shというシェルスクリプトファイルを作り，次のように記述する。

```
#!/bin/bash
ssh pi@bananapi01  “sudo shutdown -h now”;
ssh pi@bananapi02  “sudo shutdown -h now”;
.....................................
ssh pi@bananapi22  “sudo shutdown -h now”;
ssh pi@bananapi00  “sudo shutdown -h now”;
sleep 30;
shutdown -h now
```

3.8　その他，ネット関係で重要なこと

数日間にわたる計算を行うとき，PIsは，以下のようなエラーを起こすことがある。

```
$ packet_write_wait: connection to IP_ADDRESS port 22:Broken pipe
```

この場合，bananapi00上での計算は止まってはいないが，IP_ADRESSが示すノードの計算は止まっている。これは，sshがプログラム実行中に切れることが原因らしい。これを防ぐために/home/pi/.ssh/configに以下の内容を記述する。

/home/pi/.ssh/configの編集(例えば，10日間継続計算の場合)

```
ServerAliveInterval       100
ServerAliveCountMax       8640
```

3.9　ネット関係のシステムダイアグラム

これまでのまとめという意味で，図1にネット関係のシステムダイアグラムを示す。まず，NFS関連では，bananapi00の/home/piをネット上にファイルシステムとして置き，他のbana-

napi01～bananapi22およびraspberrypi24がそれをマウントして使うということである。この場合，bananapi00がマスタ，他がスレーブといった関係になる。ntpに関しては，PIsのシステムの外部への「窓口」の役割をraspberrypi24に任せているので，ここが外部のntpサーバーから時刻情報を取ってきている。bananapi00～bananapi22は，raspberrypi24をntpサーバーとして同期している。外部とのやり取り，または，内部とのやり取りは白矢印で示している。

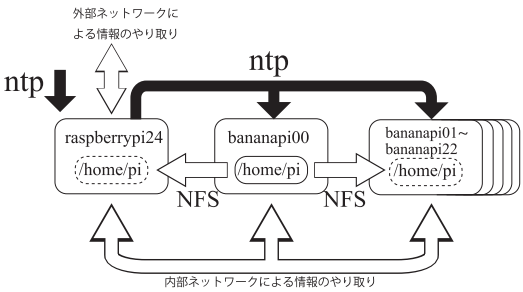

図１：PIsのネット関係のシステムダイヤグラム。bananapi00の/home/piを他のノードが参照する，といった形をとっている。時間の同期については，raspberrypi24→bananapi??という流れになっている。

3.10 PIsの組み立て

図２のように，Banana Pi，Raspberry Pi上の主なICにはヒートシンクを張っておく。１枚１枚のシングルボードは実行時，熱を持つため，これを空気中に解放させる必要がある。図３のように，12個をスペーサーで１列につなげ，２列準備する。この１列につき冷却用ファンを２個あてがう。よって冷却用ファンは４個搭載した。ファン以外に24ポートのスイッチングハブ１個，4ポートのUSB充電器７個，その他，LANケーブル24本，シングルボードコンピュータ台数分と冷却ファン台数分の電源ケーブル，７個口の一括スイッチ付き電源タップ１個，スイッチ無し電源タップ１個でPIsを構成した。

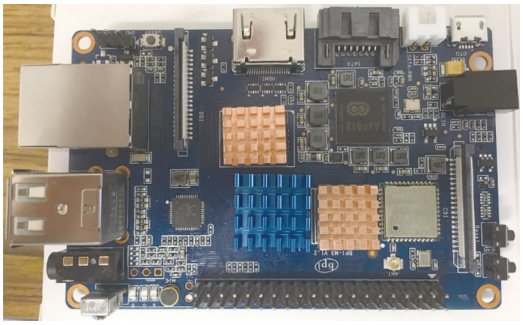

図２：Banana Pi上の主要なIC（CPU，メモリ等）にヒートシンクを張り付けている。

図３：Banana Pi1 12個をつなげている様子。もう一列にはBanana Pi 11個＋Raspberry Pi 1 個をつなげる。これら２列，計24個でPIsを構成する。

冷却用ファンは12Vで稼働するため，ファン用のUSBケーブルは5Vを12Vに変換するものを使用した。1シングルボードにつき5W必要なため，USB充電器も1ポートあたり5W以上供給できるものを選ぶ必要がある。これらのシングルボードコンピュータは電源が入ると，起動す

る仕組みなので電源タップも一括スイッチ付きを選んだほうがよい。これらを組み上げたものを，図４に示す。

１枚１枚のシングルボードは実行時，非常に熱くなる。これらの熱を空気中に解放させるため，ヒートシンクを張り，冷却ファンを実行時に常に稼働することは非常に重要である。

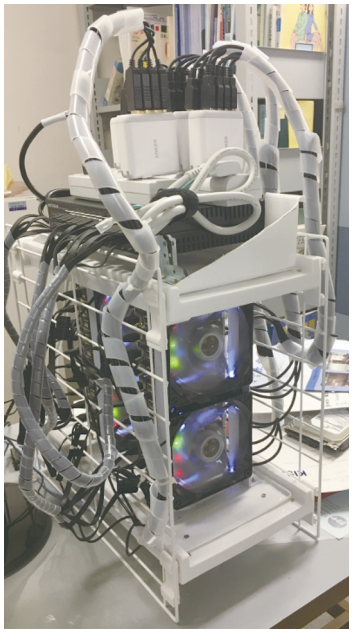

図４：PIsの全体図。冷却ファンを各シングルボードコンピュータ列の前面に設置している。上には24ポートのスイッチングハブ１個，およびUSB充電器を多数設置している。

４　PIsのパフォーマンス

4.1　使用したプログラム

マイクロセレクトロンHDR-v2は，小線源治療で使われるイリジウム線源である。これを使ったモンテカルロ・シミュレーションでPIsの計算能力を確かめた。モンテカルロ・シミュレーション・プロトコルはEGS5を使い，これをMPI[11]で走るようにしたものである（プログラム名：test_small_Ir_mpi.f）。このプログラムの計算結果の１つである線源から光子が出ている様子を図５に示す。このモンテカルロ・シミュレーションは，線源から一本一本，光子を発射させ，エネルギーが１eV以下になるまで追跡を繰り返すものである[12,13]。

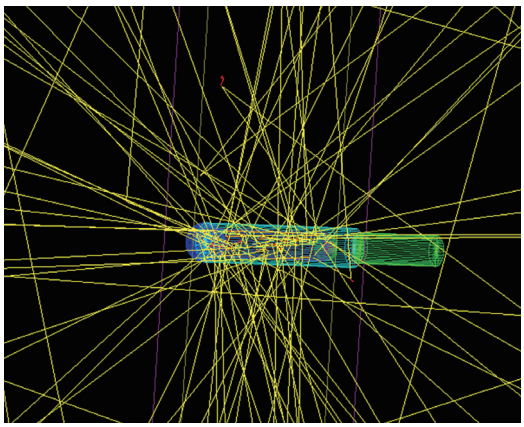

図５：test_small_Ir_mpi.fの計算結果。黄色の線は線源から発射される光子を示している。尚，右の光子が発射されていない円柱部分は，線源に接続されているステンレスワイヤーを示している。

4.2　PIsとPCとの比較

図６は，横軸に線源から発射させた光子数（イベント数），縦軸に経過時間を示したものである。比較のため，Windows 10 Proを搭載したPC（Core i7，クロック周波数4.5GHz）と比較した。モンテカルロ・シミュレーションプロトコルは，EGS5を使い，仮想PC上のLinuxで走らせた。尚，これはMPIを使わずに実行させたものである。発射光子数が少ない場合（イベント数＝10^6），PIsの計算時間は，PCとあまり変わらないが，イベント数が増えるに従い，処理速度に開きがみられる。PIsの計算速度は，イベント数が多くなると，イベント数に比例してくる。10^9イベントのこのモンテカルロシミュレーションをPCで計算した場合，約11.5日かかると予想されるが，PIsの場合，わずか13.6時間で計算できる。PCで比較した場合，PIsの

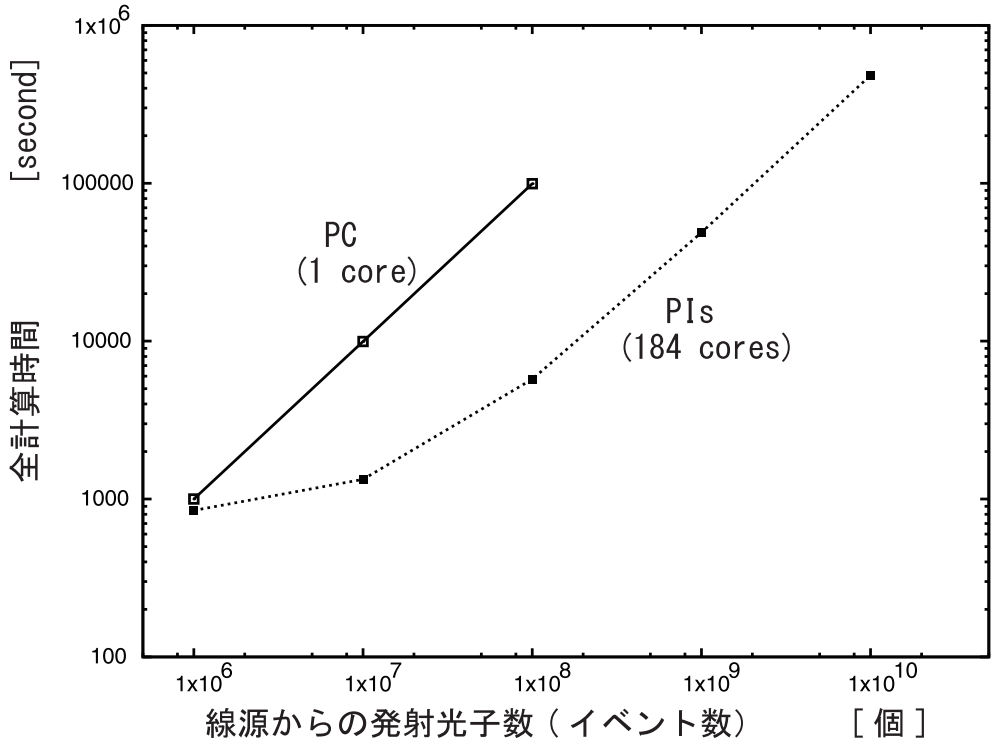

図６：線源からの発射光子数（イベント数）と，全計算時間の関係。PCとの比較のため，PCとPIsの両結果を示す。PIsは，イベント数が少ないとき，計算時間はそれほど早くならない。PIsは，イベント数10^8あたりから，比例してくる。PCの10^9イベントの結果は10^6秒（＝11.5日）ほどになることが読み取れる。

処理速度は，PCの20.4倍（PC1コアと比較した場合）となる。PCは，4コアのCore i7を搭載しているため，MPIを利用して４並列同時に計算させることができるが，それでも2.9日はかかることとなる。尚，Windows 10 ProのOS上に仮想PCを走らせ，その上でMPIを走らせた場合，理想的な処理速度よりはずっと遅くなることは確認している（仮想PC上で4コアを使った場合，処理速度は2.1倍にしかならない）。

4.3 PIsとスーパーコンピュータとの比較

以前，九州大学で借りていたスーパーコンピュータ，高性能演算サーバシステム（tatara）との比較を行った。高性能演算サーバシステム（tatara）は，「Fujitsu PRIMERGY CX400」で理論演算性能は，1476ノード，345.6GFLOPSである。このうちの16ノードを借りていた。16ノードの総コア数は，256コアである。これに前述のtest_small_Ir_mpi.fを走らせた。PIsの計算に使う総コア数は184（＝8×23）なので，tataraも10^9イベントを184コアで計算させると，1.7時間かかった。PIsでは，13.6時間かかるため，tataraのほうが8倍速い。尚，tataraで256コアフルに使うと，1.2時間で処理できることとなる。10^{10}イベントをPIsで計算させた場合，5.6日かかるが，tataraで256コアフルに使うと12時間で処理できると予想される。スーパーコンピュータで比較した場合，PIsの処理速度は，スーパーコンピュータの1/8倍（同じ184コアで比較した場合）から1/11倍（全ノード256コア

で比較した場合）となる。

4.4　PIsの消費電力

PIsの全消費電力は，MPIの計算をさせていない状態で93W，MPIを使い184コアフルに稼働した場合，173Wであった。ちなみにPCの場合，普段の使用で44W，１コアを使い，フルに計算させた場合80W，４コアフルに計算させた場合，125Wであった。

4.5　冷却ファン有り，無しでのPIsの比較（重要）

前述したようにBanana Pi，Raspberry Piは，稼働させると，熱をもつ。これを解放させるために，こまめにヒートシンクを張り，冷却ファンで風を送る必要がある。冷却ファンを稼働させた状態でtest_small_Ir_mpi.fを10^8イベント走らせると，1.6時間かかるのに対し，冷却ファンを止めた状態で同じことを実行させると，7.5時間もかかった。熱処理を考慮することは，計算速度に大いに影響することがわかる。

4.6　ネットからの操作

Banana PiおよびRaspberry Piには，画面出力のためのHDMI端子と，USB端子がついているので，これらにモニター，キーボード，マウスをつなぐことで，各々を操作することができる。今回PIsの製作において，bananapi00とraspberrypi24はGUI環境にし，他はCLI環境とした。bananapi00にモニター，キーボード，マウスをつないだ様子を図７に示す。raspberrypi24に搭載したOS，RASPBIAN STRETCH WITH DESKTOPには標準で，VNC serverがインストールされているので，PC側にVNC Viewerをインストールすれば，ネットからGUI環境で操作できる（図８）。この場合，raspberrypi24に接続したこととなる。ただし，非常に動作が遅くなるので，PCからはTera Term等でraspberrypi24に接続し，ネットからCLI環境で操作する方が無難である（図９）。ファイルのやり取りは，SFTPプロトコルを使ったソフトでやり取りできる。PIsのシャットダウンについては，キーボード，モニター等をbananapi00に直接接続した場合は，3.7章で述べたshutdown.shでシステム全体をシャットダウンすることができる。ネット上からPIsをシャットダウンするためには，まずアカウント“pi1”でraspberrypi24に接続をし，そこから3.7

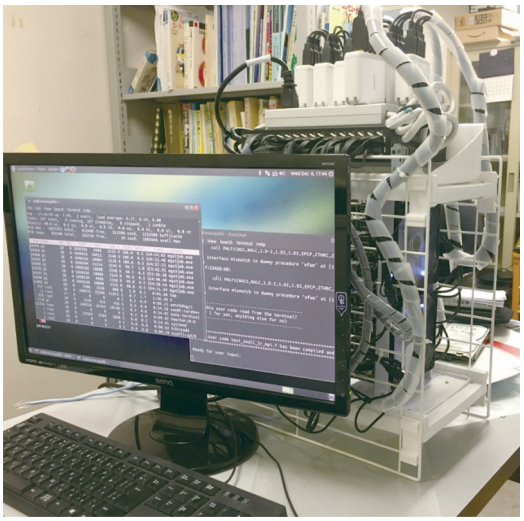

図７：PIsのデスクトップ画面。bananapi00にキーボード，マウス，モニタをつないでいる。ターミナルを立ち上げ，MPIを使い並列計算をさせている。

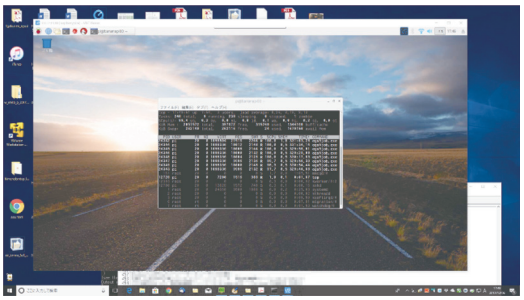

図８：PCからVNC viewerを使い，raspberrypi24に接続した様子。デスクトップからターミナルを立ち上げ，並列計算の実行をチェックしている。

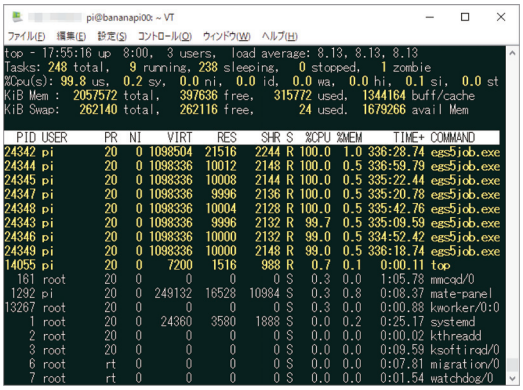

図９：PCからTera Termを使い，raspberrypi24に接続した様子。CLI環境で接続される。並列計算の実行をチェックしている。

章で述べたshutdown_ras.shを使って，システム全体のシャットダウンすることができる。

５　結論

本研究では，多数のシングルボードコンピュータを組み合せ，コンピュータ・クラスタを製作し，これが，実用レベルで計算できることを示した。

２章では，シングルボードコンピュータの設定について，３章では，コンピュータ・クラスタとして機能させるためのノウハウを，４章では，コンピュータ・クラスタPIsのパフォーマンスについて述べた。PIsの計算能力は，一般的なPCの１コアあたりと比較すると20倍，スーパーコンピュータと比較すると，ほぼ1/10倍であった。スーパーコンピュータの1/10しか計算能力を持たないが，これが，机の上に載ること，消費電力が高々170Wほどであることを考えると個人レベルで所有できる「スーパーコンピュータもどき（＝スーパーパソコン）」としては秀逸である。

参考文献

１）http://www.cenav.org/raspi1/(2017.12.13)

２）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. Springer, Cluster Computing 17:349-358, 2014

３）http://www.banana-pi.org/m3-download.html (2017.12.13)

４）https://www.raspberrypi.org/downloads/raspbian/(2017.12.13)

５）http://blog.shibayan.jp/entry/20150228/1425121187(2017.12.13)

６）H. Hirayama, Y. Namito, A.F. Bielajew, S. J. Wilderman, W. R. Nelson: The EGS5 Code System. SLAC-R-730 and KEK Report 2005-8 :2005
http://rcwww.kek.jp/research/egs/egs5.html(2017.12.13)

７）S.Tsuji : Running EGS5 with Raspberry Pi. Proseedings of the Twenty-Fourth EGS Users' Meeting in Japan 4:1-6, 2017

８）https://www.mpich.org/(2017.12.13)

９）https://ameblo.jp/takeoka/entry-11543214113.html(2017.12.13)

10）https://qiita.com/moroishi/items/a17bc20d3a3856b08d90(2017.12.13)

11）https://unit.aist.go.jp/rima/ioniz-rad/egs5mpi/index.html(2017.12.13)

12）S.Tsuji, N.Narihiro : ABOUT g(r) FUNCTION IN AAPM TG-43. Proseedings of the Eighteenth EGS Users' Meeting in Japan 6:36-42, 2011

13）S. Tsuji, N. Narihiro : Comparison about Brachytherapy of Simulation EGS5 and Treatment Planning Based on AAPM TG-43. Proseedings of the Nineteenth EGS Users' Meeting in Japan 7:85-93, 2012