Developing a single-board computer cluster : A parallel integrated system

Shuhei TSUJI

**Department of Natural Sciences, Kawasaki Medical School,*

577 Matsushima, Kurashiki, Okayama, 701-0192, Japan

(Accepted on October 5, 2019)

Abstract

In recent times, single-board computers (SBCs) are being developed with 8-core CPUs; these SBCs have sufficient computing power. However, neither their hardware nor software production methods have been described in detail. This study describes the process of building and activating a computer cluster that combines 24 ODROID-MC1 Solos. The specific production method and computational capacity of the cluster will be detailed in this paper. Moreover, the experimental result obtained in this study shows that the computer cluster created using SBCs can be practically used for Monte Carlo simulations.

Key words: Single-board computer, Computer cluster, MPI, ODROID-MC1 Solo, EGS5, EGS5-MPI

1 Introduction

Various single-board computers (SBCs), including Raspberry Pi, are available in the market. Many of these SBCs operate on low power and are inexpensive. In addition, a Linux-based operating system can be installed on them and the development environment is also in place. SBCs with an 8-core CPU and 2-GB RAM have also been introduced. Moreover, many SBCs can be linked to create small and low-cost clusters that can replicate some functionality of large data center clusters[1]. A computer cluster, “parallel integrated system" (PIs), with 23 Banana Pis BPI-M3 and one Raspberry Pi 3 Model B (Fig. 1) has been previously developed[2,3]. Banana Pi BPI-M3 is a high-performance SBC with an 8-core CPU and

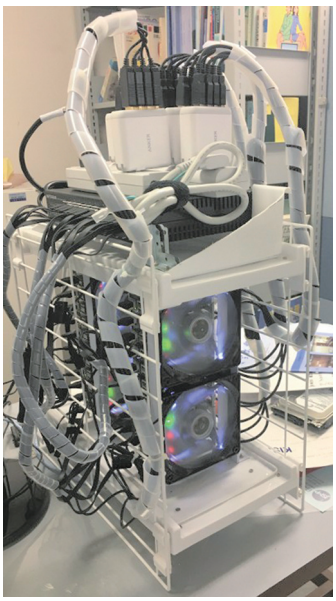

Fig.1 : PIs consisting of 23 Banana Pis and 1 Raspberry Pi on which it is possible to implement the MPI.

Kawasaki Ikaishi Arts & Sci (45) : 83−96 (2019)
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

2-GB RAM.

PIs has the following features:

- it is a computer cluster built with SBCs,
- it is possible to implement the MPI on it,
- it can operate, upload, and download files through an external network (Wi-Fi, etc.),
- the size of the PIs is almost the same as that of a desktop PC.

To create a PIs, it is necessary to use SBCs with an 8-core CPU and 2-GB RAM. This is because if the number of CPU cores is high, the number of processes required for executing the MPI will also be high; moreover, 2-GB RAM is sufficient for almost all my programs to work. PIs constructed with a Raspberry Pi and 23 Banana Pis can execute the simulation code EGS5[4]; however, this PIs is not stable for long-time calculations.

Hardkernel Co. released a single-board computer, ODROID-XU4, with 2-GB RAM and an 8-core CPU, which is the most significant SBC in terms of its capabilities. In addition, the company also sells ODROID-MC1 Solo, a product based on cluster production. ODROID-MC1 Solo has the same 8-core CPU and 2-GB RAM as those in the ODROID-XU4, but the I/O interface is simplified to consist of only one microSD card slot, one Ethernet port, and one USB port. Fig. 2 shows the ODROID-MC 1 Solo; its specifications are shown in Table 1. Several examples of computer clusters using SBCs have been introduced[5-8]. However, neither their hardware nor software production methods have been introduced in detail. Here, I will show the assembly of a computer cluster, PIs, with 24 ODROID-MC1 Solos and describe the PIs production in detail.

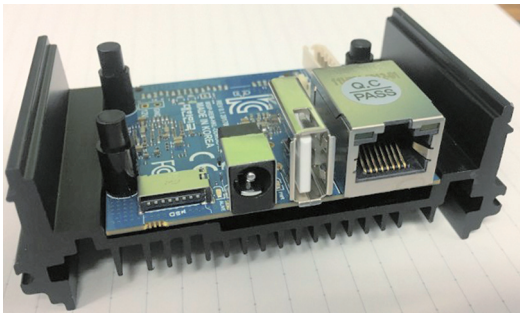

Fig.2 : ODROID-MC1 Solo consisting of only a microSD card slot, power jack socket, USB2.0 port, and Gigabit LAN port. CPU is the same as ODROID-XU4. It is specialized for single-board computer clusters.

Table 1 : ODROID-MC1 Solo specifications

CPU	Samsung Exynos 5422 (8 cores)
Memory	2-GB RAM
USB	USB 2.0 ×1
LAN	10/100/1000 Mbps Ethernet
Power Input	5V/4A power supply
Etc.	Micro-SD card slot for boot media

2 Hardware of the PIs

2.1 Computer unit

ODROID-MC1 Solo is sold with “an outer frame” that consists of a heat sink. To make it compact, this outer frame was removed and a heat sink was attached to the CPU (Fig. 3, 4). A total of 12 pieces of ODROID-MC1 Solo were connected by hexagonal spacers to make it into one unit (Fig. 5). Two such units were created. The PIs operates on a total of 24 ODROID-MC1 Solo nodes.

2.2 Power supply unit

ODROID-MC1 Solo requires 5V/4A power supply. The capacity of a general USB charger is insufficient for this. Therefore, it was necessary to prepare an AC adapter capable of

5V/4A output for each node. Six AC adapters were connected to make four sets as shown in Fig. 6. Fig. 7. shows the four adapter units of Fig. 6. attached to the chassis. The power supply units account for most of the apparatus rather than the computer units. The system becomes complicated because of the 24 power cords of 100 V.

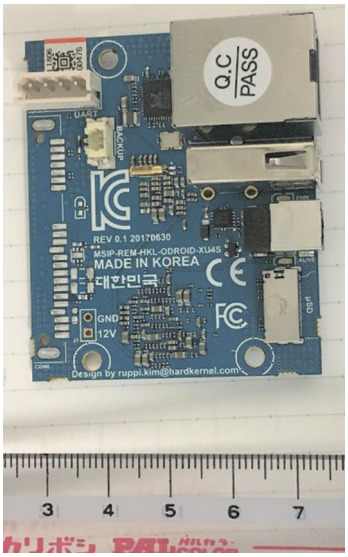

Fig.3 : Top view of the ODROID-MC1 Solo. Size: 56mm × 47mm.

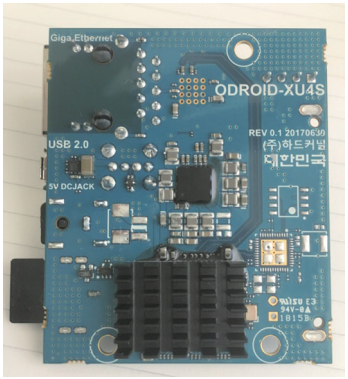

Fig.4 : Bottom view of the ODROID- MC1 Solo. The CPU is attached to a heat sink and a microSD card is inserted.

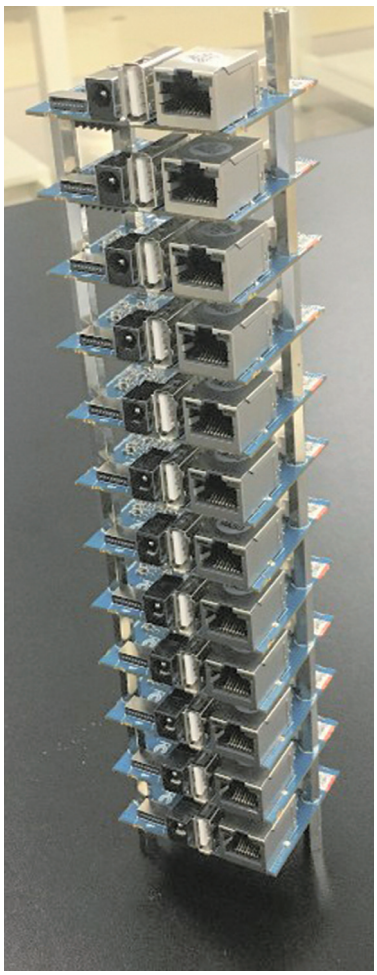

Fig.5 : Twelve pieces of ODROID connected by hexagonal spacers. PIs consists of two such units.

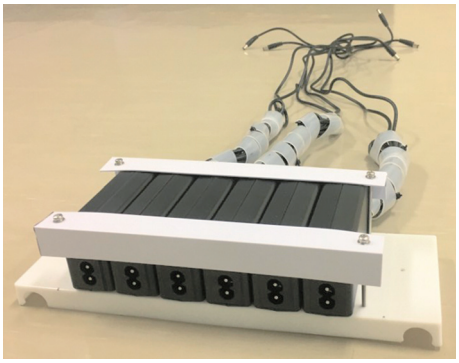

Fig.6 : Six 5V/4A compact AC adapters. PIs needs four such sets.

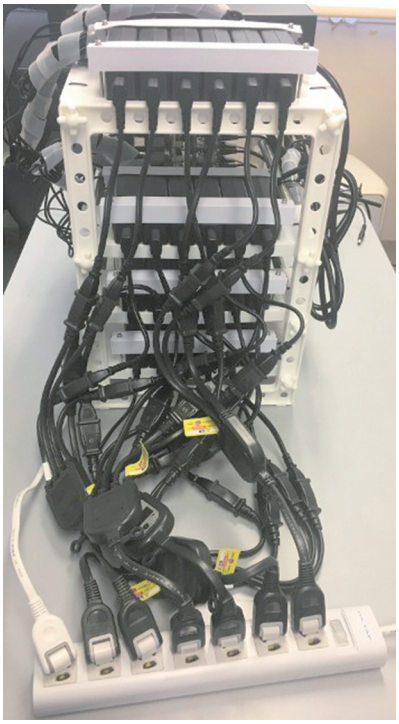

Fig.7 : Back view of the PIs. Twenty- four 100 V power cords combined into a single power strip.

2.3 Whole assembly

The PIs chassis was used as a free rack below the sink. Two fans were installed on the front of the PIs for cooling. The fan had an air flow of 89.04 CFM and a size of 12 cm × 12 cm. Because the power supply voltage was 12 V, a USB power connector that converts 5 V to 12 V was used. The network hub used a 24-port 1000 Mb/s switch. This created an internal network. The network for outside interaction (to operate, upload, and download files) was a Wi-Fi network. The Wi-Fi module was connected to the master node's ODROID-MC1 Solo's USB port. The specifications of the Wi-Fi module conformed to the IEEE 802. 11ac standard, allowing high-speed network exchange with the outside world. Two units of ODROID-MC1 Solos, two fans, a switching hub, and power adapters were installed in the chassis; these power supplies were combined at a switched power tap and the power switch turned the entire device on/off. An overview of the PIs is shown in Fig. 8, Fig. 9 and Fig. 10; the components are shown in Table 2.

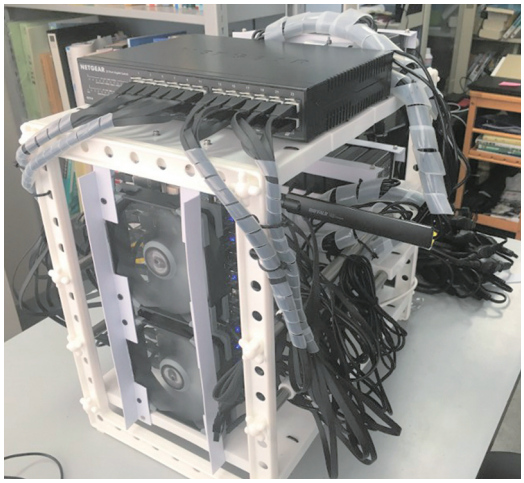

Fig.8 : Overview of the PIs.

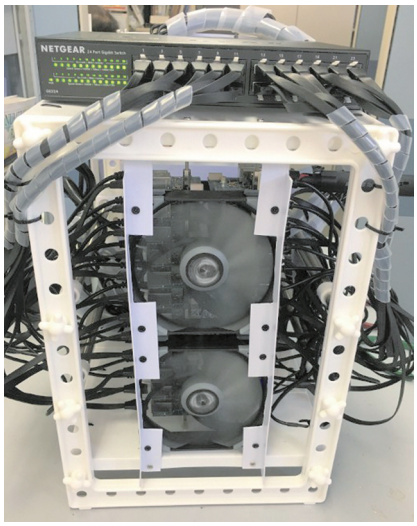

Fig.9 : Front view of the PIs.

Table 2 : Components of the PIs

SBC	ODROID-MC1 Solo	×24	￥ 139299
MicroSD	Toshiba microSDHC 32GB	×24	￥ 17952
5V AC power adapter	Anthin switching power adapter	×24	￥ 44096
Switching hub	NETGEAR GS324-100JPS	×1	￥ 9990
Wi-Fi module	BUFFALO WI-U2-433DHP	×1	￥ 2481
Fun	Scythe KAZE FLEX 120 SU1225FD12H-RP	×2	￥ 1854
Fun power supply	Scythe AS-71	×2	￥ 2902
Power strip, USB charger	ELECOM T-Y3A-3720WH	×1	￥ 1179
	ELECOM MOT-U06-2134WH	×1	￥ 1362
	SANWA SUPPLY TAP-EX4BKN	×6	￥ 4806
Etc.	Sink free rack, LAN cables, power cables, etc.		
Total			￥ 225921

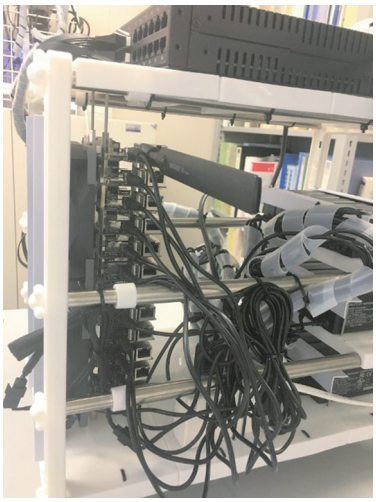

Fig.10 : Side view of the PIs. The LAN cable on one side has been removed.

3 Software of the PIs

3.1 System diagram

The PIs master nodes can exchange internal and external information through the network. The PIs master node is named “odroid00”. The other 23 nodes are slave nodes and have only an internal network. The slave nodes are “odroid01” to “odroid23”. The account name of each node is “odroid” and the home directory is common to the network file system (NFS). The system diagram of the PIs is shown in Fig. 11.

3. 2 Settings common to master and slave nodes

For setting ODROID-MC1 Solo, ODROID-XU4 is required, which necessary until the network is set up. Ubuntu 16.04.3 LTS is the selected OS. The following steps are required to be performed for the set up. First, download the image ubuntu-16. 04. 3-4. 14-mate-odroid-xu4-20171212.img.xz from the odroid site[9]. After expanding the downloaded file, start the terminal and execute the following command[10].

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt dist-upgrade
$ sudo apt install linux-image-xu3
$ sudo reboot
```

The above command can use a Wi-Fi module with the RTL8811AU or RTL8821AU chipset. Because gfortran is not installed, install gfortran.

```
$ sudo apt install gfortran
```

MPI is introduced from the source files; the MPI adopted is MPICH 3.2.1. Download mpich-

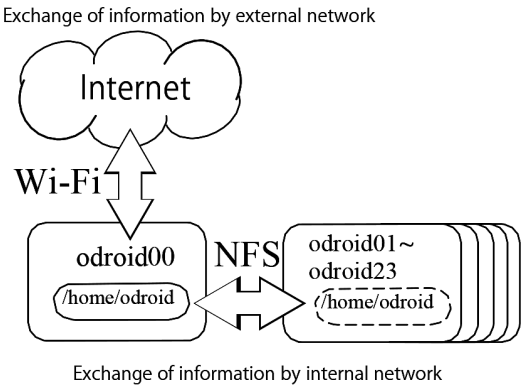

Fig.11 : Only the master node odroid00 can exchange information through Wi-Fi. The home directory of each node is that of the master node.

3.2.1 source files from the MPICH site[11] . Execute the following command.

```
$ tar xzvf mpich-3.2.1.tar.gz
$ cd  mpich-3.2.1
$ configure --prefix=/usr/local/mpich-3.2.1
$ make
$ sudo make install
```

Write in .bashrc to pass the path.

```
(/home/odroid/.bashrc:)
export PATH =..... /usr/local/mpich-3. 2. 1
/bin:$PATH
```

We now have a development environment to run MPI.

3.3 Internal network address

Define the internal network address as shown in Table 3. Set up hosts as follows:

```
(/etc/hosts:)
.................
192.168.240.100        odroid00
192.168.240.101        odroid01
.................
192.168.240.122        odroid22
192.168.240.123        odroid23
```

Modify sshd_config as follows:

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

Table 3 : Internal network of the PIs

IP address	odroid00:	192.168.240.100
	odroid01:	192.168.240.101
		⋮
	odroid23:	192.168.240.123
net mask		255.255.255.0
gateway		192.168.240.254
DNS name server		8.8.8.8
host		odroid00
slave		odroid01 ~ odroid23

3.4 Master node settings:odroid00

Network set up:

Network set up requires ODROID-XU4. Launch Ubuntu with a GUI in an environment with a Wi-Fi network. In the upper right corner of the desktop, there is an icon for network settings. Select and click on the SSID; enter a password. To configure the network manually without using DHCP, modify SSID_file as

follows:

```
(/etc/NetworkManager/system-connections
/SSID_file:)
.................
[wifi]
mac-address-blacklist=
mac-address-randomization=0
mode=infrastructure
ssid=SSID
.................
[ipv4]
address1=IP_ADRESS/NET_MASK,GATEWAY
dns=DNS
dns-search=
method=manual
.................
```

Modify Wired connection 1 as follows for wired LAN settings:

```
(/etc/NetworkManager/system-connections
/Wired connection 1:)
.................
[ethernet]
duplex=full
mac-address-blacklist=

[ipv4]
address1=192.168.240.100/24,192.168.240.
254
dns=8.8.8.8;
dns-search=
method=manual
.................
```

When a LAN cable is connected to the wired LAN connector, wired LAN is given priority. To give priority to the wireless LAN, modify NetworkManager.conf as follows[12]:

```
(/etc/NetworkManager/NetworkManager.conf:)
.................
[connection-wifi-wlan0]
match-device=interface-name:wlan0
ipv4.route-metric=0
.................
```

After rebooting, the network using wireless LAN will connect to the outside world. Modify hostname as follows:

```
(/etc/hostname:)
odroid00
```

Setting the NFS:

It is necessary to set up the NFS to have a common home directory for all nodes[8]. Execute the following command from the superuser mode.

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

Modify exports as follows:

```
(/etc/exports:)
/home    192.168.240.0/255.255.255.0(rw,
sync,no_root_squash,no_subtree_check)
```

Execute the following command; if the home directory of odroid is visible under the /mnt directory, the command was successful.

```
# exportfs -av
# mount odroid00:/home     /mnt
```

Setting SSH:
When using the MPI on multiple nodes, it is necessary to be able to login to the host and slave without a password[13]. Execute the following command.

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

Because the home directories of all nodes are common, public key and secret key can be distributed to all nodes. Setting the host file for the MPI: Modify `mpihosts/hosts` as follows[14]:

```
(/etc/home/odroid/mpihosts/hosts:)
odroid00
odroid01
........
odroid22
odroid23
```

Note that the CPU core of each node can be selected by writing the following.

```
(/etc/home/odroid/mpihosts/hosts:)
odroid00:6
odroid01:8
........
odroid22:3
odroid23:5
```

Add the following statement to `.bashrc`:

```
(/home/odroid/.bashrc:)
export HYDRA_HOST_FILE=~/mpihosts/hosts
```

Setting the VNC server:
To use a GUI environment, it is necessary to install the VNC server. Execute the following command.
`$ sudo apt install vnc4server`
After setting the password, start the VNC server.

```
$ vncpasswd
Password:
Verify:
$ vncserver :1
```

Start the VNC viewer software from Windows and connect to "IP address: 5901". To start the MATE desktop environment, modify `xstartup` as follows[15].

```
(/home/odroid/.vnc/xstartup:)
[ -x /etc/vnc/xstartup ] && exec /etc/vnc/
xstartup
[ -r $HOME/. Xresources ] && xrdb $HOME/
.Xresources
xsetroot -solid grey
exec mate-session &
```

Execute the following command to stop the VNC server.
`$ vncserver -kill :1`

3.5 Slave node settings:odroid01

The slave nodes start from odroid01 to odroid23; however, here odroid01 is treated as a representative slave node. The remaining nodes may be changed in the same way by changing the node number.
Setting NFS:
Enable communication with the outside world

using a wired LAN (or wireless LAN). Execute the following command to build the NFS for the slave node.

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

Add the following statement to fstab:

```
(/etc/fstab:)
........
odroid00: /home    /mnt/odroid_home nfs
noatime 0 0
```

It is better to create another "odoroid1" account in case you cannot login using the "odoroid" account.

```
# adduser odroid1
```

Execute the following command in the superuser mode[13].

```
# mv /home/odroid  /home/odroid.tmp
# ln -s /mnt/odroid_home /home/odroid
```

If successful, after rebooting, the slave node's home directory will be the same as the master node's home directory.

Network set up:

Modify Wired connection 1 as follows for wired LAN settings:

```
(/etc/NetworkManager/system-connections
/Wired connection 1:)
.................
[ethernet]
duplex=full
mac-address-blacklist=
[ipv4]
address1=192.168.240.101/24,192.168.240.
254
dns=8.8.8.8;
dns-search=
method=manual
.................
```

Edit hostname as follows:

```
(/etc/hostname:)
odroid01
```

The slave node does not need a GUI environment; therefore, use the following command to create a CLI environment.

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

Setting NTP:

Because the slave node is not connected to an external network, it is necessary to synchronize the master node as a time server. Comment out the line "pool" in ntp.conf and add the following statement:

```
(/etc/ntp.conf:)
server 192.168.240.100 iburst
```

3.6 Shut down and reboot the PIs

To shut down or reboot the entire PIs, it is necessary to create a script file for it. Execute the following command on each node[16].

```
$ sudo visudo
```

Then, modify it as follows.

```
(/etc/sudoers:)
odroid [tab] ALL = NOPASSWD: [tab] /sbin/
shutdown
```

```
odroid[tab]ALL=NOPASSWD:[tab]/sbin/reboot
```

Create a file called shutdown.sh in /home /odroid and write the following.

```
(/home/odroid/shutdown.sh:)
#!/bin/bash
ssh odroid@odroid01 "sudo shutdown -h now";
ssh odroid@odroid02 "sudo shutdown -h now";
    .....
ssh odroid@odroid23 "sudo shutdown -h now";
sleep 30;
shutdown -h now
```

The reboot file reboot.sh is written as follows.

```
(/home/odroid/reboot.sh:)
#!/bin/bash
ssh odroid@odroid01 "sudo reboot";
ssh odroid@odroid02 "sudo reboot";
    .....
ssh odroid@odroid23 "sudo reboot";
sleep 30;
sudo reboot
```

In addition, it is convenient to prepare a script file netcheck.sh for network check.

```
(/home/odroid/netcheck.sh:)
#!/bin/bash
ssh odroid@odroid01 "exit";
echo "odroid01 OK"
ssh odroid@odroid02 "exit";
echo "odroid02 OK"
    .....
ssh odroid@odroid23 "exit";
echo "odroid23 OK"
```

Another setting is to stop unnecessary daemons.

4 PIs Application

4.1 PIs GUI

PIs needs to be remotely controlled through the network. If the user connects to Windows via SSH, he/she can operate it in a CLI environment. From this terminal, the GUI environment with VNC viewer can be used by executing the following command.

```
$ vncserver :1
```

From VNC viewer in Windows, the user can connect to "IP_address:5901". Fig. 12 shows the appearance of the PIs projected by the VNC viewer. The MPI can execute up to 192(=8×24) processes with the following command after processing the program with mpicc or mpifort.

```
mpiexec -n 192 EXECUTE_FILE
```

4. 2 Configure and run high-performance Linpack (HPL)

HPL was used to confirm the computational power of the PIs calculation. Download hpl-2.3.tar.gz from the HPL site[17]. The introduction of HPL is as follows[7,18].

```
$ sudo apt install libatlas-base-dev
$ tar xzvf hpl-2.3.tar.gz
$ cd hpl-2.3/setup
$ sh make_generic
$ cp Make.UNKNOWN ../Make.odroid
```

Modify Make.odroid as follows.

```
(/home/odroid/hpl-2.3/Make.odroid:)
    .....
ARCH    =odroid
```

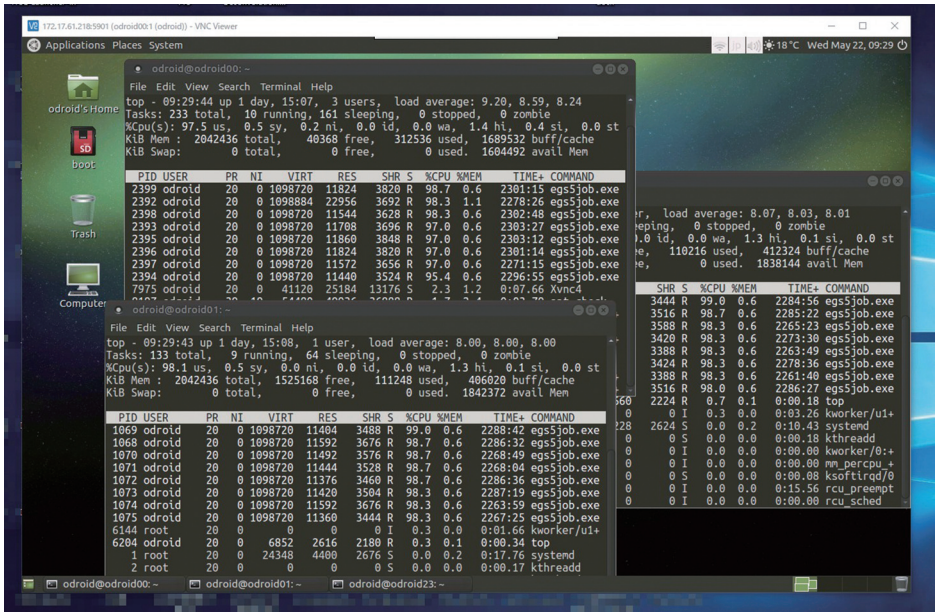

Fig.12 : When launching the VNC viewer on Windows 10, the desktop screen of master node odroid00 is launched. The terminal shows that each node is running a parallel calculation through the MPI.

```
    ....................................
TOPdir  =$(HOME)/hpl-2.3
    ....................................
MPdir    =/usr/local/mpich-3.2.1
MPinc    =-I $(MPdir)/include
MPlib    =$(MPdir)/lib/libmpich.so
    ....................................
LAdir    =/usr/lib/atlas-base
LAinc    =
LAlib    =$(LAdir)/libf77blas.a  $(LAdir)
/libatlas.a
    ....................................
```

If libatlas-base-dev is installed on all the nodes and dynamic libraries are used, the following should be performed:

```
LAlib    =$(LAdir)/libf77blas.so  $(LAdir)
/libatlas.so
```

After modifying `Make.odroid`, execute the following command:

```
$ make arch=odroid
```

If the make-command passes, the `xhpl` executable file is created in the folder `~/hpl-2.3/bin/odroid`. The calculation result when using 192 processes can be seen with the following command.

```
$ mpiexec -n 192 ./xhpl
```

The best value after several iterations is 65.7 GFLOPS as shown next (using dynamic libraries):

T/V	N	NB	P	Q	Time	Gflops
WR11C2R4	60000	512	12	16	2190.34	6.5746e+01

The power consumption of this calculation is about 300 W. Therefore, it becomes 2.2×10^2 MFLOPS/W.

4.3 Execution of EGS5

EGS5[4] is one of the Monte Carlo simulation codes for electron transport. The calculation results using this were compared for a PC and PIs. The specifications of the PC were Core i7-6700K with 32-GB memory. The CPU frequency was rated at 4.0 GHz but overclocked to 4.5 GHz. Virtual computer VMware was running on Windows 10 Pro. EGS5 was running on Scientific Linux on VMware. The PC used only a one-core CPU; PIs used 192 cores in 24 nodes. I used the sample program ucphantomcgv.f[19] of EGS5 to compare the computational power. I used an unmodified program for the PC, but improved the PIs for MPI. In addition, to use EGS5 for MPI, the EGS5-MPI[20] package was required. The aim of the program was to inject a photon with 1.253 MeV energy into water and measure the absorbed dose by the depth of the water. Increasing the incident photons increases the accuracy but requires more computation time. The relationship between the incident photon number and calculation time for PC and PIs is shown in Fig. 13. In case of the PIs, because initializing for generating events requires time, the number of photons from 10^6 to 10^8 required the same calculation time. The extrapolation of the PC having 10^{10} incident photons results in a calculation time of 2.46×10^6 s(= 28.5 days). Based on this value,

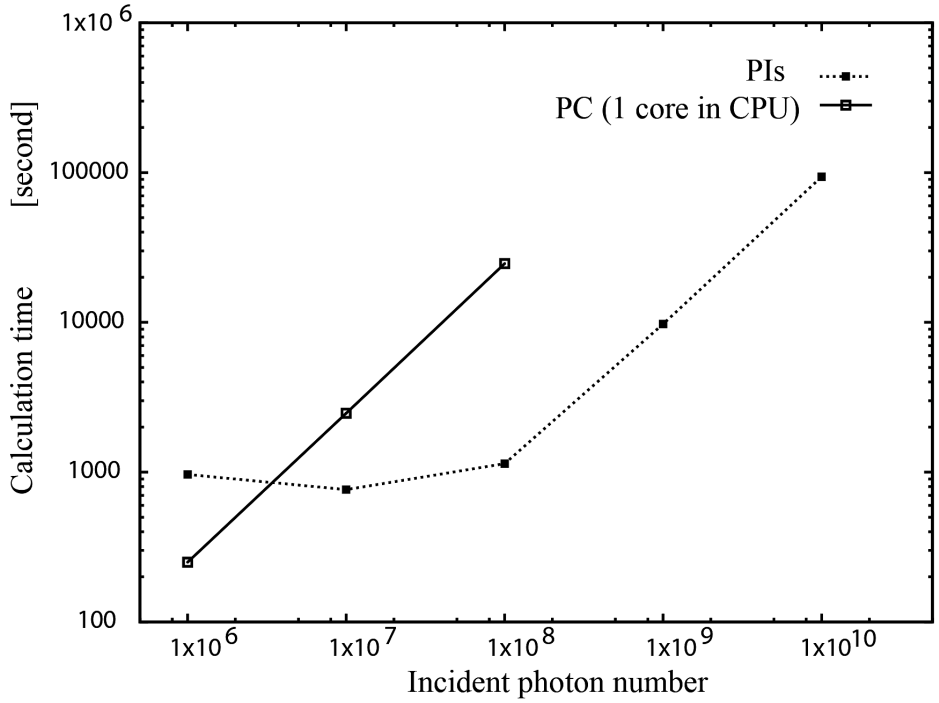

Fig.13 : Relationship between incident particle and calculation time when using the ucphantomcgv. f program. PC measures the number of incident photons as 10^6 to 10^8 and PIs measures 10^6 to 10^{10}. When the photon number is small (= 10^6), calculation using a PC with 1 core in the CPU is faster.

the calculation capacity of the PIs was estimated to be 26.2 times more than that of the PC.

4.4 Power consumption

If the PIs is loaded with calculations, the power consumption is approximately 300 W (Fig. 14). It may momentarily exceed 300 W, but it settles at around 300 W when the calculation load is applied by the MPI.

Fig.14：Power consumption when performing parallel calculations is 303 W. The lower numbers indicate the amount of power.

5 Conclusion

In this research, a computer cluster, PIs, was created by using ODROID-MC1 Solos and the hardware and software was explained in detail. The HPL calculation showed a value of 65.7 GFLOPS, but when comparing the calculations of PIs and PC (with one core of Core i7 6700 K (4.5 GHz), 32-GB memory) for EGS5, PIs was observed to be 26.2 times faster. Longer computation times stand out in the PIs performance. This shows that the computer cluster made of SBCs can be practically used for Monte Carlo simulations. A limitation is that 24 AC adapters occupy a considerable part of the device; to overcome this, several 5V/50A PSUs can be used to make the PIs compact. Moreover, here, cooling is very important; although it is cooled by a fan, it will be better to install it in a room equipped with air conditioning.

At the time of developing this computer cluster with SBCs, there was not much information on this. I believe this paper would be useful for those planning to create a computer cluster with SBCs and would lead to the emergence of high-performance, compact computers in the future.

6 Acknowledgment

I would like to thank Editage (www.editage.com) for English language editing.

The author declares no conflicts of interest associated with this manuscript.

References

1）S. J. Johnston, P. J. Basford, C. S. Perkins, H. Herry, F. P. Tso, D. Pezaros, R. D. Mullins, E. Yoneki, S. J. Cox, J. Singer: Commodity single board computer clusters and their applications, Future Generation Computer Systems 89: 201–212, 2018

2）S. Tsuji: Development of computer cluster “Parallel integrated system” (in Japanease), Kawasaki Igakkai Shi Liberal Arts & Sciences 43:37–48, 2017

3）S. Tsuji: Running EGS5 with the Computer Cluster PIs, Proc of the 25rd EGS Users’ Meeting in Japan 2018–13:1–10, 2019

4）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 (2019.6.27)
5）S. J. Cox, J. T. Cox, R. P. Boardman, S. J. Johnston, M. Scott, N. S. O 'Brien: Iridis-pi: a low-cost, compact demonstration cluster, Cluster Comput. 17:349-358, 2014
6）P. Stevens, Raspberry Pi Cloud: 2017,
https://blog.mythic-beasts.com/wp-content/uploads/2017/03/raspberry-pi-cloud-final.pdf (2019.6.27)
7）http://www.cenav.org/raspi1/ (2019.6.27)
8）https://ameblo.jp/takeoka/entry-11543214113.html(2019.6.27)
9）Index of /ubuntu 16.04lts ubuntu-16.04.3-4.14-mate-odroid-xu4-20171212.img.xz: 2017
http://odroid.in/ubuntu_16.04lts (2019.6.27)
10）Release Note of Ubuntu 16.04.3 LTS (v3.1): 2018,
https://wiki.odroid.com/odroid-xu4/os_images/linux/ubuntu_4.14/20171212 (2019.6.27)
11）Mpich: 2017,
https://www.mpich.org/downloads (2019.6.27)
12）Networkmanager.conf(5):2019,
https://manpages.debian.org/unstable/network-manager/NetworkManager.conf.5.en.html (2019.6.27)
13）http://earth.sci.ehime-u.ac.jp/~kameyama/How_To_Setup_PC_Clusters/index7.html (2019.6.27)
14）Using the Hydra Process Manager: 2015,
https://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager(2019.6.27)
15）https://qiita.com/pochy9n/items/271b1ac0f608c1d43715 (2019.6.27)
16）https://qiita.com/moroishi/items/a17bc20d3a3856b08d90 (2019.6.27)
17）HPL - A Portable Implementation of the High-Performance Linpack Benchmark for Distributed-Memory Computers: 2018,
https://www.netlib.org/benchmark/hpl/ (2019.6.27)
18）https://qiita.com/hinemoss/items/304e506f71f1ab828cc9 (2019.6.27)
19）EGS activities at KEK, Code for Source, Nai, Phantom (uccg-150303.tar.gz): 2015,
http://rcwww.kek.jp/research/egs/kek/ (2019.6.27)
20）M.Shimizu: EGS5-MPI: 2005,
https://unit.aist.go.jp/rima/ioniz-rad/egs5mpi/index_eng. html
https://unit.aist.go.jp/rima/ioniz-rad/egs5mpi/doc/egs5mpi_manual_eng. pdf (2019.6.27)