

A Study on Distributed System Construction and Numerical Calculation Using Raspberry Pi

Young-ho Ko¹, Gyu-Seong Heo², and Sang-Hyun Lee³

¹Assistant Professor, Department of Electrical Engineering, Jeonbuk National University, Korea
koyh@jbnu.ac.kr

²Bachelor(B.A.), Department of Computer Engineering, Honam University, Korea
hydrogen@naver.com

³Assistant Professor, Department of Computer Engineering, Honam University, Korea
leesang64@honam.ac.kr

Abstract

As the performance of the system increases, more parallelized data is being processed than single processing of data. Today's cpu structure has been developed to leverage multicore, and hence data processing methods are being developed to enable parallel processing. In recent years desktop cpu has increased multicore, data is growing exponentially, and there is also a growing need for data processing as artificial intelligence develops.

This neural network of artificial intelligence consists of a matrix, making it advantageous for parallel processing. This paper aims to speed up the processing of the system by using raspberrypi to implement the cluster building and parallel processing system against the backdrop of the foregoing discussion.

Raspberrypi is a credit card-sized single computer made by the raspberrypi Foundation in England, developed for education in schools and developing countries. It is cheap and easy to get the information you need because many people use it. Distributed processing systems should be supported by programs that connected multiple computers in parallel and operate on a built-in system. RaspberryPi is connected to switchhub, each connected raspberrypi communicates using the internal network, and internally implements parallel processing using the Message Passing Interface (MPI). Parallel processing programs can be programmed in python and can also use C or Fortran. The system was tested for parallel processing as a result of multiplying the two-dimensional arrangement of 10000 size by 0.1. Tests have shown a reduction in computational time and that parallelism can be reduced to the maximum number of cores in the system. The systems in this paper are manufactured on a Linux-based single computer and are thought to require testing on systems in different environments.

Keywords: Raspberry PI, Cloud computing, Parallel computing, Matrix, Numerical computation.

1. Introduction

The recent evolving as digital systems based on wired and wireless networks [1]. In the past, data was

processed sequentially using a single core.

Based on Moore's Law, it was thought that performance would continue to increase, but due to process limitations, performance was no longer increased and it was changed to a multi-core structure.

Multicore processors are now available for embedded systems as well as PCs and servers [2], parallel processing techniques using multicore can be an alternative to meeting both real-time and energy efficiency.

Much research has been done on Multi-task with multicore to perform multiple tasks simultaneously to improve throughput [3] and parallel processing server that can perform one task at a time to improve response time [4]. Recent supercomputers do not simply connect CPUs, but add accelerators such as GPUs to increase speed and efficiency [5]. In this paper, raspberry-pi is connected with a cluster to embody a distributed processing system to reduced required time for numerical computation processing. The raspberry-pi is a single-board computer for education and developing countries made by the raspberry-pi foundation in the U.K., and because it's cheap for \$35, it can build a system at a low cost. It can also be useful for multicore testing because it has quad-core CPU [6, 7].

For the system, Message Passing Interface (MPI), a standard for distributed processing programs, was used. The MPI is made up of message and tag and it is used to communicate between the main PC and the sub-pc. Messages have data, and tag has the order and source of the message, and the calculation results are delivered. The system of this paper is based on the debian linux raspbian operating system and the development language is python.

2. Cluster Architecture

At a low cost, 4 PCs were replaced by raspberry-pi for the design and fabrication of distributed systems.

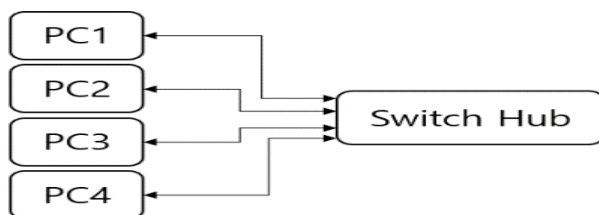


Figure 1. Cluster Structure

Each PC is connected by Ethernet. The switch hub connects each PC and allocates the internal IP for each PC. This communication structure can be developed easily using network equipment.

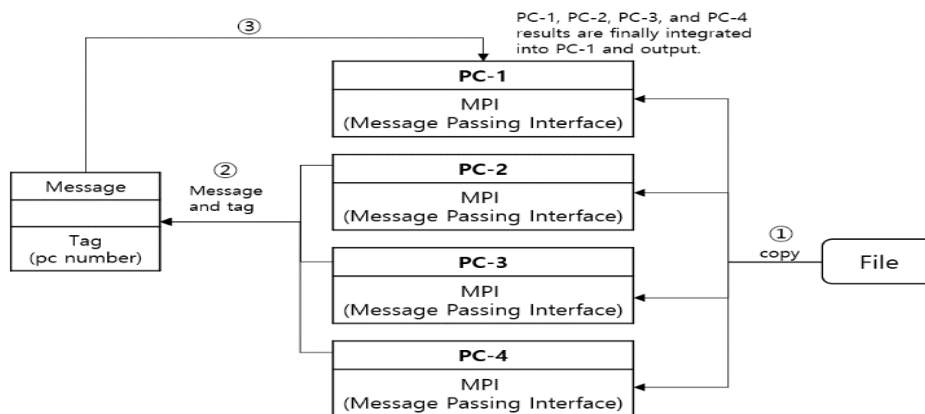


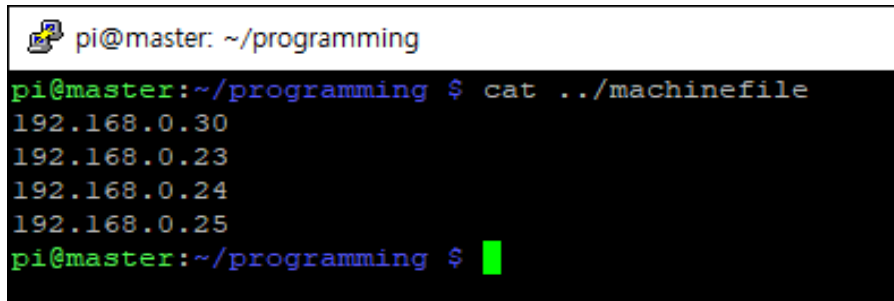
Figure 2. MPI Networking

As shown in fig. 1 four raspberry-pi are connected by a switch hub. The raspberry-pi is assigned an internal IP through the switch hub and communicates with each other.

The Fig. 2 is communication structure of MPI. The MPI operates in the following structure. ① Copy the source cord file to PC-1, PC-2, PC-3, PC-4 ② when the MPI is working, the process works on each PC and performs the calculation of the quota set by the programmer and it's going to be transmitted to PC-1 in PC-2, PC-3 message. In this time, message has DATA and status, tag is using to distinguish messages. ③ PC-1 receives information from PC-2, PC-3, and PC-4 and consolidate and print out the results.

3. Implementation

3.1 Node Connection

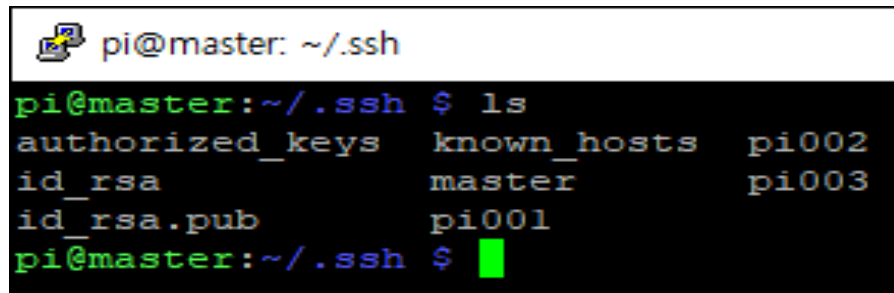


```

pi@master: ~/programming
pi@master:~/programming $ cat ../machinefile
192.168.0.30
192.168.0.23
192.168.0.24
192.168.0.25
pi@master:~/programming $ █

```

Figure 3. Machuinefile

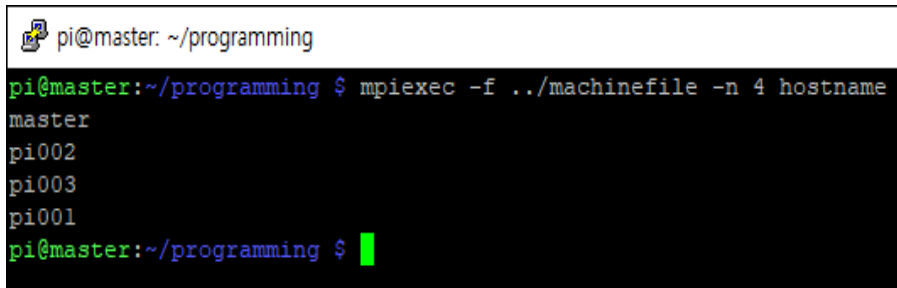


```

pi@master: ~/.ssh
pi@master:~/ .ssh $ ls
authorized_keys  known_hosts  pi002
id_rsa          master       pi003
id_rsa.pub      pi001
pi@master:~/ .ssh $ █

```

Figure 4. RSA key



```

pi@master: ~/programming
pi@master:~/programming $ mpiexec -f ../machinefile -n 4 hostname
master
pi002
pi003
pi001
pi@master:~/programming $ █

```

Figure 5. Node connection test

Internally, use the mpich library to send and receive messages through the IP on each node.

Fig. 3 shows an IP of the raspberry-pi stored in the main chain file. To access accounts on each node, you must go through a login process, but it can be omit able by using RSA public key password.

The public key is stored in the id_rsa.pub file in fig. 4, and the secret key is stored in the ID_RSA file. Store each node's public key in the authorized_ keys file. Running these steps will pass the login process. The authorized keys file stores the public keys of the remaining nodes except itself. As shown in fig. 5, the four hosts are connected properly.

3.1 Node Connection

```

pi@master: ~/programming
pi@master:~/programming $ mpiexec -f ../machinefile -n 1 python test2.py
=====
Running 1 parallel MPI processes
20 iterations of size 10000 in 21.60s: 0.93 iterations per second
=====
pi@master:~/programming $ mpiexec -f ../machinefile -n 2 python test2.py
=====
Running 2 parallel MPI processes
20 iterations of size 10000 in 11.12s: 1.80 iterations per second
=====
pi@master:~/programming $ mpiexec -f ../machinefile -n 3 python test2.py
=====
Running 3 parallel MPI processes
20 iterations of size 9999 in 7.32s: 2.73 iterations per second
=====
pi@master:~/programming $ mpiexec -f ../machinefile -n 4 python test2.py
=====
Running 4 parallel MPI processes
20 iterations of size 10000 in 5.66s: 3.53 iterations per second
=====
pi@master:~/programming $ █
    
```

Figure 6. Numerical computation time

Fig. 6 shows the implementation of the proposed study and results from running test codes in the cluster.

Run the parallel processing program test2.py with the mpiexec command referring to the IP of the machine file in each pc.

Table 1. Test results

Sequence	Number of Clusters	Time sent on calculation(sec)	Remarks
1	1	21.24	
2	2	11.12	
3	3	7.36	
4	4	5.56	
		...	
15	15	2.01	
16	16	1.70	Best
17	17	2.79	

Table 1 shows the results of distributions of 10000 two-dimensional arrays multiplied by 0.1.

As shown in fig. 6, the calculation time was reduced as the number of processes increased, and able to get a time shortening effect. Also, best results were achieved when the number of processes was set to 16, and the multiple of 4 was increased.

With two processes, it was reduced by 10 seconds and with three processes, it was reduced by 4 seconds. These results show that performance increases as the process increases, but increases nonlinearly.

4. Result

Fig. 7 shows a cluster connected with SSH for distributed processing. The results are printed by integrating them from PC-1 by executing file copied to each PC. Each PC communicates with the RSA password and IP. Physically, each PC is connected to a switch hub and connected to an internal network to communicate.

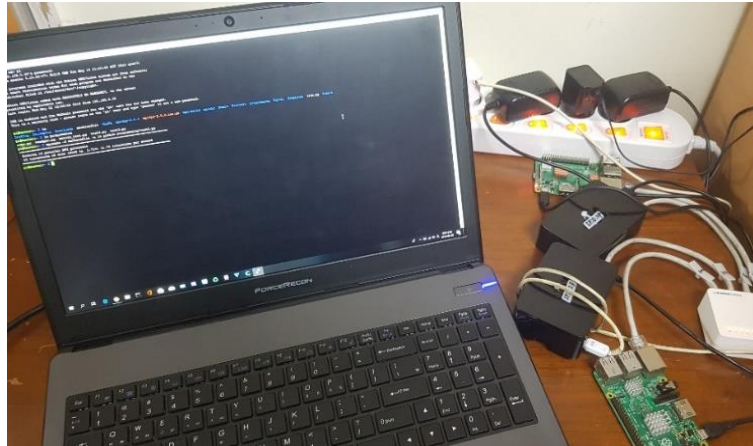


Figure 7. SSH Communication

5. Conclusion

In the past, there was a limit to increasing calculation speed in the sequential form of single-core CPUs. To improve these shortcomings, the CPU has evolved to increase the core, supercomputers have been developed to improve performance by connecting multiple PC.

Designing and implementing the proposed content in this paper, we can see that the calculation time decreases as the process increases. The time-shortening effect of the number of processes was best achieved when the number of processes was generally increased by a multiple of 4 or when the number of processes was set to 16.

As the number of CPU cores and PCs increase, so does the performance of the system. However, since this performance gain does not increase linearly, it seems impossible to increase it infinitely.

Therefore, it is important to find a compromise between performance and cost when build the system. The study used raspberry-pi to build clusters at low cost and found that it could reduce the time required to perform matrix operations. The research content of this paper is that artificial intelligence can be converted into matrix structure because each node is connected to each other, and this matrix structure is advantageous to the distributed processing system.

The system was developed using MPI. Therefore, it seems possible to transfer to various systems, and it is possible to further reduce the calculation speed by adding the system's PC. In addition, PCs with lower specification CPU will likely be able to be reused, and high-quality distributed processing systems will be able to be deployed at a relatively.

References

- [1] F. Nilsson and A, "Communications, Intelligent Network Video: Understanding Modern Video Surveillance Systems," Taylor & Francis, January 2008.
- [2] M. Levy and T. Conte, "Embedded Multicore Processors and Systems," IEEE Micro, Vol. 29, No. 3, pp. 7-9, May

2009.

DOI: <https://doi.org/10.1109/MM.2009.41>

- [3] J. Hennessy and D. Patterson, "Computer Architecture: A Quantitative Approach- Fourth Edition," Elsevier, Dec. 2007.
- [4] I. Ahmad, Y. He, and M. Liou, "Video Compression with Parallel Processing, Parallel Computing," Vol. 28, No. 7, pp. 1039-1078, August 2002.
DOI: [https://doi.org/10.1016/S0167-8191\(02\)00100-X](https://doi.org/10.1016/S0167-8191(02)00100-X).
- [5] K.-W.Cho, S.-M.Seo, J-H.Na, J-W.Kim, J-H.Kim, J.Lee, J-H.Park, Y-J.Lee, H-J.Kim, S-Y.Kang, J-Y.Joo, S-M.Park, W-G.Jung, K-H.Im, J-J.Lee. Heterogeneous supercomputer technology trends and the development of the supercomputer 'Chendung'. Communications of the Korean Institute of Information Scientists and Engineers, pp. 34-41, April 2013.
- [6] S.B. Park, J.-Y. Lee, K.-D. Jung, "The Design of Library System using the Cloud Environment Based on the Raspberry pi," International Journal of Advanced Smart Convergence (IJASC), Vol. 4, No. 1, pp. 31-34, 2015.
DOI: [10.7236/IJASC.2015.4.1.31](https://doi.org/10.7236/IJASC.2015.4.1.31)
- [7] M.B. Choi, S.K. Park, "Design and Implementation of Vehicle Internal Alarm System using Raspberry-pie Multi-sensor," International Journal of Advanced Smart Convergence (IJASC), Vol. 7, No. 2, pp. 112-118, 2015.
DOI: <https://doi.org/10.7236/IJASC.2018.7.2.112>