Case study
UDC 634.711:632.6/.7:595.77
DOI 10.7251/IJEEC1802083M

Image Processing on Raspberry Pi Cluster

Dušan Marković[1], Dejan Vujičić[2], Dragana Mitrović[2], Siniša Ranđić[2]

[1]University of Kragujevac, Faculty of Agronomy, Čačak, Serbia
[2]University of Kragujevac, Faculty of Technical Sciences, Čačak, Serbia

dusan.markovic@kg.ac.rs, dejan.vujicic@ftn.kg.ac.rs, dragana.mitrovic.94@gmail.com, sinisa.randjic@ftn.kg.ac.rs

Abstract – **The development direction of the high-performance computing has been primarily oriented toward improvements in the number of computing units, and their better organization and interconnection. The central processing units in modern mainframes are in some cases inadequate for data parallelization, because a large amount of data requires a large number of processing units. This problem can be partially overcome by introducing embedded devices with enough processing power and with smaller power dissipation, but the new problems emerge, specifically the problem of their interconnection in a distributed environment. One of such devices is a Raspberry Pi board, which has four processing cores in the latest revision, and we combined four of such devices in a cluster. These devices are interconnected via Ethernet router and by using MPI interface, they can be programmed to work as a single unit. We supplied a set of images that were processed on the cluster and measured its performance.**

Keywords – cluster, distributed computing, image processing, Message Passing Interface, parallel computing, Raspberry Pi

I. INTRODUCTION

In the structure of many problems, it is possible to see the built-in parallelism that raises the issue of computer support for solving such problems. This has further contributed to the fact that parallel computers have a significant place in research in the field of computer technology. The intensive development of semiconductor technology and its impact on the development of computing has provided a real basis for parallel computing [1].

Particularly significant progress in parallel computing was achieved through computer clusters realized within the TCP/IP network [2], [3]. In such systems, the parallel computation process takes place through the exchange of messages [4]. The best known systems for parallel computing support within the TCP / IP cluster are MPI (Message Passing Interface) [5] and PVM (Parallel Virtual Machine) [6].

The first computer clusters were based on local personal computer networks. The emerging of microcomputers, realized on the principle of one-board computer, implied the expansion of the application of computer clusters for parallel computing. These computer modules usually have standard communication interfaces (USB, Ethernet, HDMI ...) that allow them to connect to the necessary peripheral devices, including connections to different types of computer networks. Typical representatives of these computer modules are the families of Raspberry Pi [7] and BeagleBone [8] microcomputers. The Raspberry Pi modules have proven to be suitable for implementing a TCP / IP cluster designed to support parallel computing.

Due to processor and memory characteristics, as well as the ability to connect to the Ethernet network, Raspberry Pi is a good basis for the formation of the appropriate computer cluster. The aim of this paper is to demonstrate the possibility of realization of the computer cluster based on the Raspberry Pi 3 module connected in the TCP/IP network.

Experience in working with clusters based on personal computers has demonstrated their significant capabilities in supporting parallel computing. Particular attention should be paid to the acceptability of such systems, as they were significantly cheaper than expensive supercomputers. The ability to realize clusters on the basis of one-board computers, such as the Raspberry Pi modules, has further increased the importance of clusters for parallel computing.

Image processing is one of the applications in which computing time is a very important factor. This type of problem has a significant built-in parallelism that allows the necessary calculations to be realized through the execution of parallel programs. With this in mind, in the Computer Science Laboratory at the Faculty of Technical Sciences in Čačak, a cluster of four nodes based on the Raspberry Pi 3 modules has been developed.

The primary goal of this development was to explore the possibility of forming such clusters and their programming based on the MPI concept. An additional goal was to explore the possibility of parallel programming based on MPI concept using the Python programming language. Finally, the cluster was supposed to be a platform in which students of computer science would become familiar with the basic principles of parallel computing and parallel programming.

At the same time, in this Laboratory, the research in the field of image processing and recognition of forms related to the monitoring of various phenomena in agriculture is carried

out. The high processing requirements of this application have imposed the idea to conduct research within the application of the developed cluster.

The remainder of the paper is organized as follows. The second section describes related work in the field of Raspberry Pi cluster formation and usage. The third section brings out the importance of image processing and gives introduction to the done practical example. The fourth section describes the created Raspberry Pi cluster and its architecture. The fifth section deals with the message passing interface and its implementation on the cluster, with the possibility of using the cluster remotely. The sixth section describes the practical realization, given results, and discussion. The seventh section presents with the educational aspects of the cluster. The final section brings out concluding remarks and future work directions.

II. RELATED WORK

The authors of [9] managed to create a cluster of 300 nodes based on Raspberry Pi Model B computers. These units are equipped with 700MHz ARM11 processor, with 512MB of operating memory. They used it for the research in areas of cloud computing, specifically for the purposes of establishing clusters in the environments without enough financial support for operating with large data centers. Their predecessors were authors of [10], which managed to create a cluster of 64 Raspberry Pi Model B computers. Their Iridis-pi cluster is shown in Fig. 1 [10].

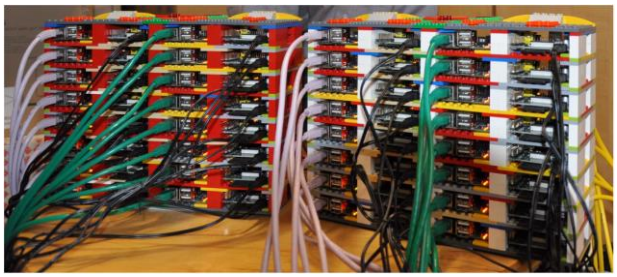

Figure 1. Iridis-pi cluster [2]

With the same computer model, but with less nodes (33 nodes, of which 32 are computational nodes), the author of [11] succeeded in creating a cluster. Their Beowulf cluster is shown in Fig. 2 [11]. The authors of [12] also used the same model, but with 56 nodes separated in four racks. Their goal was to create a cloud data center with the support of virtualization and resource management.

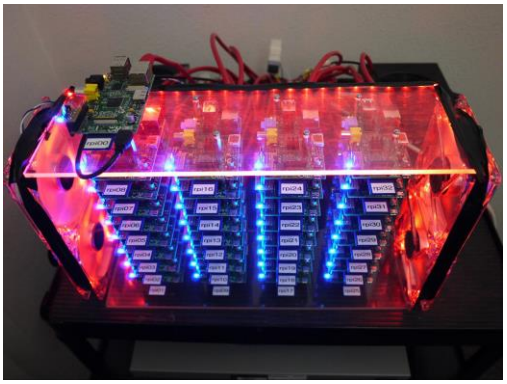

Figure 2. The Beowulf cluster [11]

The possibility of using Raspberry Pi cluster as a container in an edge cloud architecture was investigated in [13]. Cluster in their work has virtualization possibilities in order to satisfy PaaS (Platform-as-a-Service) architecture demands, with the special attention on the portability and container interconnections. The authors of [14] used 6 Raspberry Pi 1 Model B units to create a cluster for usage in Big Data manipulation in tourism. They have successfully demonstrated cluster possibilities in the process of people geolocation.

A cluster of 10 Raspberry Pi Model B nodes was used in [15] as a honeypot cluster in researching possibilities for detection and prevention of the SQL injection intruders. In [16], the authors compared a cluster of 14 Raspberry Pi modules with Intel Core i5 and Core i7 CPU computers.

Raspberry Pi 2 Model B was used in the construction of cluster with 25 nodes in [17]. Their goal was to create a cluster for power measurement and educational purposes, proving that the excellent performances can be obtained with low power consumption. Their cluster is shown in Fig. 3 [17]. Previously, the authors of [18] had the same idea in mind.

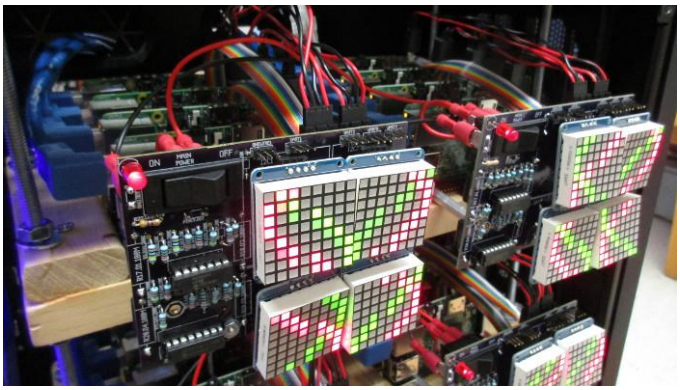

Figure 3. A cluster of 25 nodes

In [19], the authors demonstrated an intrusion detection system for Internet of Things devices by deploying Snort on Raspberry Pi module. Although they didn't form a cluster, they suggested that a cluster would have greater possibilities in a smart city environment. The author of [20] constructed a cluster of 8 Raspberry Pi 2 modules in order to facilitate video streaming in small data centers. In [21], authors made a cluster consisting of 10 Raspberry Pi 2 Model B, benchmarked it, and compared its performance with custom supercomputer.

III. PRACTICAL EXAMPLES OF IMAGE PROCESSING

Today, in the world of information and ubiquitous data obtained by different type of measurements, images represent one important subset of data. Examples of gathering images could be satellite maps, radar images, computer tomography imaging, microscopy slides, forest area, agriculture area, or other images in the agriculture field that record current condition in the observed production or research process.

Example of image processing was shown in [22] as a scientific package for electron microscopy images. Another application could be found in analysis on images obtained via X-ray microtomography [23] for problems in porous media research. Image processing could find a role in food industry [24]. In the area of agriculture, image analysis could have important application such as plant extraction in the field [25].

Also, in agriculture, image processing could be used to estimate leaf area index from fruit trees [26].

One of the activities in the image analysis is objects detection and their count. Also, in the research area of an agronomist, the appliance of object counting as a means to find parameters of interests in an effective way is of great importance. Detecting grains of wheat and their number on some area represent an important factor to determine the yield. In this paper, as practical example of image processing, counting of wheat grains was presented.

The main goal in breeding, selection, and production of wheat is to obtain as much grain yield as possible. The main components of wheat grain yield are the number of ears per unit ground area, the number of grains per ear and their mass, and the weight of 1,000 grains.

During the process of breeding, in the creation of new varieties, it is necessary to determine the number of grains in many wheat ears that belong to different varieties. Besides that, it is also important to select particular varieties with a higher number of grains in the ear for further selection. In laboratories, the number of grains per ear was determined by hand counting individual grains. However, that could take a lot of time, and represents an additional effort for the laboring people who perform this action. Nowadays, there are various counters for determining the number of grains in the ear and masses of 1,000 grains.

Mostly, the grains are poured into the counter, and on the basis of their passage between the sensors, the counting of the grain is performed. This operation takes a certain amount of time depending on the speed of the counter. The advantages of the counter over the image analysis are the shortened procedure time and duration of the grain counting.

All the grains from the ears are poured in one container without taking into account their arrangement. The container is placed beneath the camera with image processing device which is used to determine the number of grains from a particular ear in only a few seconds. In this way, the work of people in laboratories is improved, the number of grains is determined in a shorter time period, which is very important when it comes to the large number of relatively small samples.

IV. RASPBERRY PI CLUSTER

A cluster can be thought of as a set of interconnected computers. By connecting large number of computers in a cluster, one can gain potential increase in performance by doing operations in parallel and distributed environment. A cluster is usually formed using TCP/IP network with the support of MPI and PVM.

We constructed a cluster containing four Raspberry Pi 3 Model B modules, whose characteristics are given in Table I, and its appearance is presented in Fig. 4 [27].

As can be seen from Table I, Raspberry Pi 3 Model B computer board has significant processing power with quad-core ARM Cortex A53 processor and 1GB of operating memory. Furthermore, it possesses great connecting potential, since it has integrated Bluetooth and Wi-Fi transceivers, as well as Ethernet port and four USB ports. Of course, its great capabilities for expansion are seen from large number of GPIO pins, with various and numerous header expansion boards available on the market.

TABLE I. RASPBERRY PI 3 MODEL B MODULE CHARACTERISTICS

Feature	Description
CPU:	Quad-core 64-bit ARM Cortex A53, 1.2GHz
GPU:	400MHz VideoCore IV multimedia
Memory:	1GB LPDDR2-900 MHz SDRAM
USB:	4 ports
Video outputs:	HDMI, composite video (PAL and NTSC)
Network:	10/100Mbps Ethernet and 802.11n Wireless LAN
Peripherals:	17 GPIO, HAT ID Bus
Bluetooth:	Yes, v4.1
Power source:	5V via MicroUSB or GPIO header

Figure 4. The appearance of Raspberry Pi 3 Model B

The individual computers in a cluster are called nodes, and depending on the connection method, they can be interconnected in several different ways [28] – [30]. In our case, we have one head node and three computational nodes (Fig. 5). They are connected in a local network via Ethernet router, and the appearance of the cluster is given in Fig. 6.

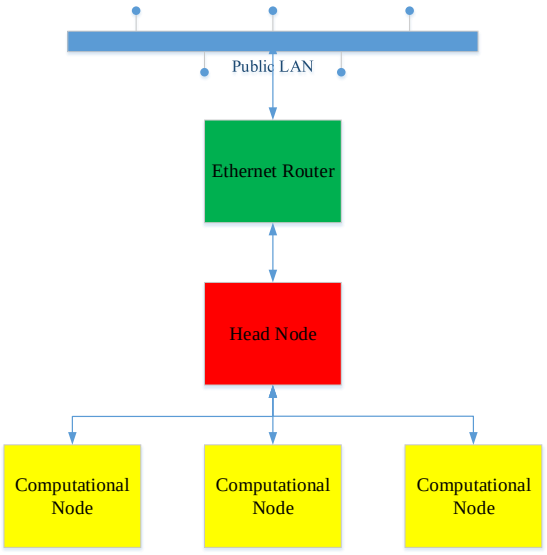

Figure 5. The architecture of the cluster

Figure 6. The appearance of the Raspberry Pi 3 cluster

The important aspect of this cluster is its modularity. New nodes can be added without any restrictions, with only few files being modified in this process. However, due to the realized design shown in Fig. 5, new nodes should probably be added in the racks of four units. Furthermore, this scenario involves adding new Ethernet switches or routers, and maybe even the cooling framework. Based on the literature review, the cluster cooling can be optimally realized as active cooling with fans and proper air-flow. Of course, this would further require even a separate space for storing such a device, depending on the number of units involved.

V. PARALLEL PROCESSING WITH MPI FRAMEWORK

Due to the advancement of computer technology, parallel processing, as an approach to speeding up the computation process, is gaining importance. It is particularly important to use networked computers for parallel execution of loosely coupled processes. Therefore, each local computer network can be used as a base for parallel computing. In loosely coupled processes, the parallel computing is based on message exchange. In order to provide conditions for parallel computing, appropriate software support has been developed. The most famous systems are MPI and PVM [31].

MPI is a communication protocol for programming parallel computers. Originally, MPI was designed for distributed memory architectures, but now it can run on almost any hardware platform, distributed memory, shared memory, hybrid, and so on. Some advantages of using MPI are: support for full asynchronous communication, grouping of processes based on the context, flexibility, and portability.

The emergence of small, but sufficiently powerful computer modules, along with their good communication features, such as Raspberry Pi and BeagleBone, enabled the formation of computer clusters suitable for parallel processing. In order to form a cluster, there has to be at least 2 or 3 nodes (Raspberry Pi) and if it is not enough, it is possible to add more later on. Programs on the Raspberry Pi module are executed under the control of the Raspbian Jessie operating system, which is based on the Linux / Debian operating system [32].

After installing Raspbian on each node of a cluster, it should be possible to generate SSH keys for their IP addresses. SSH stands for secure shell and it is used as the encrypted remote login protocol and a way to communicate with other nodes on the same network. SSH can be configured over Wi-Fi and once configured SCP (Secure Copy) and SFTP (Secure File Transfer Protocol) can be used for transferring files and directories directly from one node to another [33].

It is also possible to directly run commands on the selected node via SSH, change host names of the nodes, or even shutdown a node. At this point, it is good to have another SD card with more memory to serve as a disk for the cluster. Depending on the type of the cluster that is being made, this additional memory can be available to all nodes or just the head node.

In addition to installing and configuring the operating system, it is necessary to install MPI software on the SD card on each of the Raspberry Pi modules. In this case, the MPICH3 version is installed. Also, an MPI4PY library is installed that allows programming clustered nodes in the Python programming language.

Usage of the identical system software on all nodes is enabled by cloning the SD module. Also, each node is assigned a unique name (HostName) and nodes are enabled to work with the SSH protocol. The IP addresses of each node are stored in the so-called machine file. This file has to be located on each node and uses MPICH3 to communicate and send/receive messages between nodes [34].

Finally, it is necessary to generate SSH keys to allow management of each Raspberry Pi module without using the username and password. Fig. 7 shows a schematic representation of the MPI cluster structure with hostname, IP addresses, and SSH keys used.

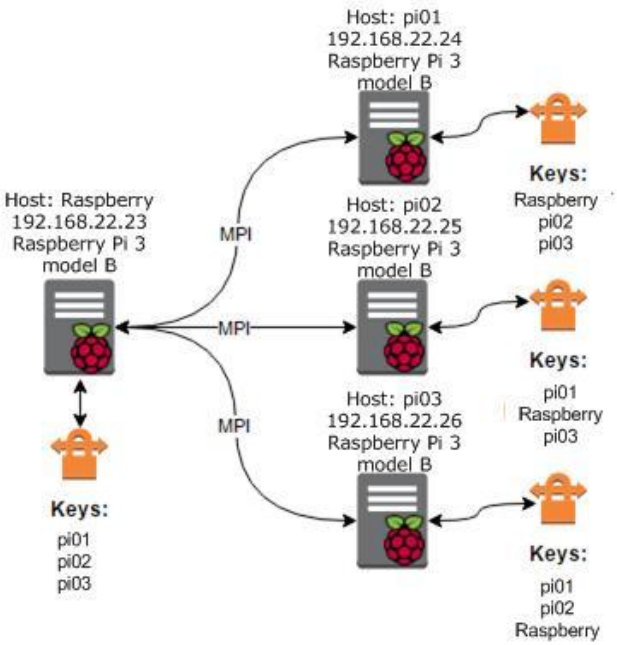

Figure 7. The example of hostname and SSH keys in the MPI cluster based on the Raspberry Pi 3 module

The research with Raspberry Pi cluster for image processing has showed wider aspects of the possibilities of such cluster. Accordingly, the future research has been defined. It is primarily focused on the development of the parallel algorithms and corresponding parallel applications. Also, the further research implies the development of the cluster with larger number of nodes.

The positive feedback from the usage of this cluster and the development of parallel applications has given an idea to provide this cluster to the end users via Internet. The cluster would be a part of a server environment. It would serve the user demands for the software support based on the parallel processing. Regarding this idea, the corresponding software support was realized and satisfactory preliminary results were obtained. The structure of such system is shown in Fig. 8.

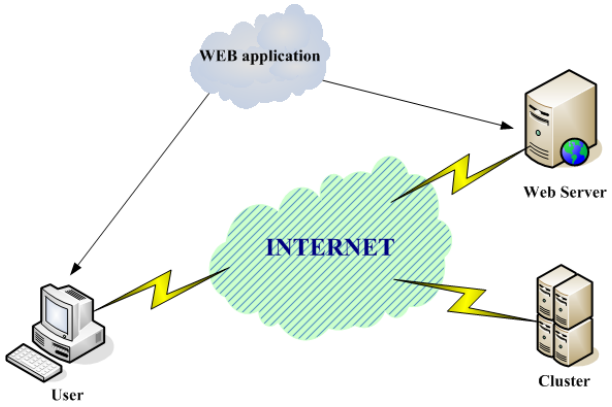

Figure 8. The structure of the system supporting access to the cluster via Internet

Within the proposed system of using the computer cluster based on the Raspberry Pi modules, it was intended that the end user can access the corresponding Web application on the remote server. This process would be done in the Web browser. After the successful authentication, the user is presented with the possibility to deploy his parallel application to the cluster nodes and start it. The results obtained by parallel processing can be forwarded to the user or put on the server storage. In the latter case, the user would access the results via custom Web application.

VI. RESULTS AND DISCUSSION

A cluster of four Raspberry Pi (RPi) devices was established with software support and configuration for parallel processing. In this purpose, Mpi4Py was used for message passing between head node and other computing nodes intended for the program execution in the parallel mode. Taken images were streamed to the cluster and distributed to the computing nodes.

On every node, a software support for image processing was also installed and that was achieved with *scikit-image* library. This is a library that contains a collection of algorithms and utilities for image processing, written in Python programming language, with some sections implemented in Cython. *Scikit-image* has Open Source license and represents software tool that could be used without restrictions under Modified BSD license [35].

The example of image processing, which was running as a test application, is represented by objects detection and their counting in arrival array of images.

Figure 9. Grains of wheat

The image of grain wheat is shown in Fig. 9 and it was used as a sample for testing image processing on the cluster. Test case for processing images in parallel mode was implemented by using the appropriate algorithm from *scikit-imag*e library on every node.

$$num\_objects = len(np.unique(label\_image)) - 1 \qquad (1)$$

The images are analyzed in order to detect the edges of the objects and to label them. After that, the Python code to count labeled image is relatively easy, as shown in (1). Presented line code (1) calculates variable *num_object* (number of objects) obtained by function **len()** which gives the number of items in an array. In this case, *np.unique*(*label_image*) returns the sorted unique elements of an array, and that would be labeled image. Final value would be result of the **len()** function subtracted by 1, because the background is labeled by 0.

Another way to test image analysis on Raspberry Pi cluster is to use *Watershed* algorithm for segmentations. This process represents a method for separating objects from background and also separating the objects. Image analysis in the process of segmentation has the aim to decide which pixel belongs to which object.

After this step, usually there are small holes on the image that have to be filled. This is necessary to carry out to avoid the detection of small holes as real objects. On segmented objects by *Watershed* algorithm, their labels can be set and then using Python expression (1) it is possible to easily count objects on the image.

Initially, the testing was performed by processing 3 images on one Raspberry Pi device using *scikit-image* library for Edge Detection (ED) of the objects. Other case was processing of the same images on one Raspberry Pi device using *Watershed* segmentation (WS) algorithm. The same process was conducted on three computing nodes using one CPU core. Results of execution times for overall processing on 3 images are shown in Table II. The biggest time of 14 seconds is needed for execution of *Watershed* algorithm and for 3 images. The same process transferred to parallel analysis on three nodes takes about 4.9 seconds.

TABLE II. RESULTS OF PROCESSING A SEGMENT OF 3 IMAGES

Number of images	WS algorithm 1 RPi	ED 1 RPi	WS algorithm 4 RPi Cluster	ED 4 RPi Cluster
	Time (s)			
3	14.0	11.2	4.9	4.9

In the next step, a new segment of 6 images was used and the testing process was repeated with addition of using two CPU cores on every Raspberry Pi computing node. According to the results (Table III), the sequence of 6 images was processed in 28 seconds, and in parallel mode it was done in 5 seconds. In this case, the time for parallel processing does not change significantly in regard to the first cluster test, because there is also one CPU per image in the test frame.

TABLE III. RESULTS OF PROCESSING A SEGMENT OF 6 IMAGES

Number of images	WS algorithm 1 RPi	ED 1 RPi	WS algorithm 4 RPi Cluster	ED 4 RPi Cluster
	Time (s)			
6	28.0	22.3	5.0	5.1

The same situation is in the third test on the cluster, which results are shown in Table IV. There were 9 images transferred to three computing nodes and every computing node was using 3 CPU cores to process arriving images. There are the same analogies in the result set with Raspberry Pi nodes using 4 CPU cores each to process 4 transferred images.

TABLE IV. RESULTS OF PROCESSING A SEGMENT OF 9 IMAGES

Number of images	WS algorithm 1 RPi	ED 1 RPi	WS algorithm 4 RPi Cluster	ED 4 RPi Cluster
	Time (s)			
9	42.0	33.5	5.3	5.2

The complete test results from processing a sequence of images on one Raspberry Pi device and a cluster of four Raspberry Pi devices is presented in Fig. 10.

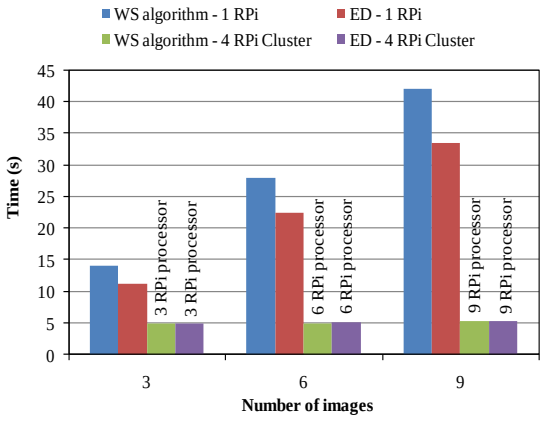

Figure 10. Results after testing of image processing on Raspberry Pi cluster

According to the processing time of image sequence on one node, there are linear dependences of the number of images in testing segment. One segment of the processing could be divided on cluster nodes and executed in an acceptable time frame. In our example, this processing time frame was approximately 5 seconds. In practical use, every segment for processing that contains 3 to 12 images could be processed on cluster in the expected time frame. In this way, the result of image analysis and object counting could be obtained near the location where cluster was placed and close to real time.

In the form presented in this paper, cluster could be deployed on location where images were captured. Immediately, the images could be processed without the need to send all of them to the remote server or cloud. Powerful computer systems on cloud platform would be ideal for more or less complex image processing. A cluster of Raspberry Pi single board computers can respond to the similar request, but with the longer execution time of processing. The advantage of the cluster would be its portability in terms of physical setup near the location where the images are taken. The cluster could be easily extended with additional computing nodes by making small changes in configuration of the head node.

In our example, three computing nodes were used for execution of the sequence of images. According to the presented results, the cluster with 3 Raspberry Pi computing nodes using maximal processing power can respond to the load of 12 images in 5 seconds, which is equal to 2.4 images per second. Accordingly, this dependence could be used to calculate necessary number of computing nodes in the cluster according to the number of images loaded in the cluster. This could be accomplished by (2), and it is written in Python programing language.

$$N_{compute} = math.ceil\ (\ (\ N_{images} * T_{seq}\ )\ /\ N_{cpu\text{-}act}\ ) \qquad (2)$$

where:

- $N_{compute}$ – number of required computing nodes in the cluster;
- N_{images} – number of images per seconds that could be loaded in sequenced order to the cluster system;
- T_{seq} – time to process one segment of arrived images where one image has reserved one CPU core;
- $N_{cpu\text{-}act}$ – number of CPU cores on Raspberry Pi that are reserved for processing from head node.

Also, in (2), the math function **ceil()** from Python is used to round the result to the higher integer value. So, if the calculated value exceeds any integer value, the result will be the next integer value. This means that one additional computing node is needed regardless of the value after the decimal point.

Testing with the cluster of one head and three computing nodes gave results shown in Fig. 10 and also represent pattern of values from which further usage of cluster in image processing could be planned. Images from one location are not required for sending to a remote computer. Users could be satisfied with image analysis on local cluster consisting of Raspberry Pi if execution times were acceptable. Calculation of required number of computing nodes in a cluster is a necessary step to form an appropriate cluster that could respond to user needs in terms of active image processing.

VII. EDUCATIONAL ASPECTS OF THE CLUSTER

Since the Raspberry Pi cluster was created within a research conducted on the master studies, the further research should also emphasize the educational aspect of the cluster. The students of the undergraduate studies of Computer Engineering are engaged in the Parallel Computer Systems course in which the realized cluster can be used in many ways. First of all, the students can get to know the basic concepts of loosely coupled computer systems based on message passing interfaces. The realized cluster enables students to gain additional knowledge regarding the setting of the local computer network parameters. Also, by using the cluster of four Raspberry Pi modules as a starting point, the students can get to know the ways of the cluster realization based on larger number of nodes.

The second important aspect of using the cluster in educational purposes is acknowledgement of MPI concept and possibilities of its deployment on Raspberry Pi based cluster. This includes the installation of MPI libraries in order to obtain the proper environment for the execution of parallel applications.

The developed cluster is a part of the local computer network in the Computer Science Laboratory and represents the fair platform for the development of parallel programs. The possibilities of the cluster were first benchmarked on the matrix multiplications examples, and later was used in image processing.

Starting with the fact that Raspberry Pi operating system has built-in Python interpreter, this programming language was used for the development of parallel programs. In this way, the students can get closer inspection to the Python programming language. Likewise, the students can gain the experience in using Python for development of parallel applications, based on the MPI concepts on the Raspberry Pi cluster.

VIII. CONCLUSION

Computer clusters designed for parallel computations have gained additional significance by the emergence of computer modules, such as Raspberry Pi and BeagleBone. Good process and network characteristics of these modules allow clusters with a large number of nodes to be formed based on them. This article presents the MPI Cluster, which consists of four nodes based on the Raspberry Pi 3 module and its use for parallel image processing.

Through the conducted research, within which the shown cluster was formed, the aspects of using the Raspberry Pi 3 module for this purpose, the installations of the required software, and the methods and requirements for configuring nodes within the cluster were examined. Also, the experience has been gained in terms of the knowledge in programming such a computer using the MPI concept.

The realized research has shown that the construction of this cluster helped students of master studies in acquiring additional knowledge in the field of parallel processing. Consequently, the developed cluster, as well as the future research on this plan, will provide a good basis for the education of students in the field of parallel processing.

Starting from the assumption that a developed computer cluster can be used for image processing, an application has been developed in which the number of wheat grains is counted in a photographed sample. Parallel computation using the Raspberry Pi cluster was carried out through an application written in the Python programming language and the MPI concept. In the development of the application, two algorithms were applied – Watershed and Edge Detection. The application speed was tested first on one Raspberry Pi 3 module, then on the Raspberry Pi 3 cluster of four nodes. During the testing, the processing was carried out on 3, 6, and 9 images with the engagement of 1, 2, or 3 processor cores.

As can be seen from Table I – IV and Fig. 10, the processing time on one Raspberry Pi 3 module is proportional to the number of processed images for both algorithms. On the other hand, cluster processing time is approximately the same for both algorithms and in the processing of different number of images.

It is planned that, as a part of further research, a cluster with a large number of Raspberry Pi nodes would be formed. It is also planned to use the programming of a more complex algorithm to use such a cluster and MPI concept, and to conduct an appropriate comparative analysis in relation to the use of the sequential program.

Also, it will be interesting for further research to compare performance times and other performance factors for different platforms in which software applications for the same algorithms would be executed. The further research would imply the use of different programming languages for the development of parallel applications and the implementation of an appropriate comparative analysis of the results obtained.

ACKNOWLEDGEMENT

The paper presents the results of the research within the Hardware – Software Co-Design course at the Master studies in the field of Computer Engineering at the University of Kragujevac, Faculty of Technical Sciences in Čačak. Financial support to the research was realized through the TR32043 project funded by the Ministry of Education, Science, and Technological Development of the Republic of Serbia.

REFERENCES

[1] Z. J. Czech, “Introduction to parallel computing,” 1st Edition, Cambridge University Press, 2017.

[2] C. S. Yeo, et al., “Cluster computing: High – performance, high – availability and high – throughput processing on a network of computers", In Zomaya, A. Y. (eds) Handbook of Nature – Inspired and Innovative Computing, Springer, Boston, MA, pp. 521 – 551, 2006.

[3] J. Levesque, and G. Wagenbreth, “High performance computing: programming and applications,” 1st edition, Chapman and Hall/CRC, 2010.

[4] S. Bova, et al. “Parallel programming with message and directives”, Computing in Science and Engineering, vol. 3, no. 5, pp. 22 – 37, September/October 2001.

[5] N. Desai, R. Bradshaw, A. Lusk, and E. Lusk, “MPI Cluster System Software”, Proceedings of 11th European PVM/MPI User’s Group Meeting Budapest, Hungary, pp. 277 – 286, September 19 – 22, 2004.

[6] A. Geist, et al., “PVM: Parallel virtual machine: A user's guide and tutorial for network parallel computing,” The MIT Press, 1994.

[7] A. Robinson, M. Cook, „Raspberry Pi Projects,” 1st Edition, Wiley, 2013.

[8] D. Molloy, “Exploring BeagleBone: Tools and techniques for building with embedded Linux,” 1st Edition, Wiley, 2014.

[9] P. Abrahamsson et al., “Affordable and energy-efficient cloud computing clusters: The Bolzano Raspberry Pi cloud cluster experiment,” 2013 IEEE 5th International Conference on Cloud

Computing Technology and Science (CloudCom), vol. 2, pp. 170-175, 2013.

[10] S. J. Cox et al., "Iridis-pi: a low-cost, compact demonstration cluster," Cluster Computing, vol. 17, no. 2, pp. 349-358, 2014.

[11] J. Kiepert, "Creating a Raspberry Pi-based Beowulf cluster," Boise State University, pp. 1-17, 2013.

[12] F. P. Tso, D. R. White, S. Jouet, J. Singer, and D. P. Pezaros, "The Glasgow Raspberry Pi cloud: A scale model for cloud computing infrastructures," 2013 IEEE 33rd International Conference on Distributed Computing Systems Workshops (ICDCSW), IEEE, pp. 108-112, 2013.

[13] C. Pahl, S. Helmer, L. Miori, J. Sanin, and B. Lee, "A container-based edge cloud PaaS architecture based on Raspberry Pi clusters," IEEE International Conference on Future Internet of Things and Cloud Workshops (FiCloudW), pp. 117-124, 2016.

[14] M. d'Amore, R. Baggio, and E. Valdani. "A practical approach to big data in tourism: a low cost Raspberry Pi cluster," Information and Communication Technologies in Tourism 2015, Springer, Cham, pp. 169-181, 2015.

[15] S. Djanali, F. X. Arunanto, B. A. Pratomo, H. Studiawan, and S. G. Nugraha, "SQL injection detection and prevention system with Raspberry Pi honeypot cluster for trapping attacker," 2014 International Symposium on Technology Management and Emerging Technologies (ISTMET), pp. 163-166, 2014.

[16] A. Ashari, and M. Riasetiawan, "High performance computing on cluster and multicore architecture," TELKOMNIKA (Telecommunication Computing Electronics and Control), vol. 13, no. 4, pp. 1408-1413, 2015.

[17] M. F. Cloutier, P. Chad, and V. M. Weaver, "A Raspberry Pi cluster instrumented for fine-grained power measurement," Electronics, vol. 5, no. 4, p. 61, 2016.

[18] A. M. Pfalzgraf and J. A. Driscoll, "A low-cost computer cluster for high-performance computing education," IEEE International Conference on Electro/Information Technology, Milwaukee, WI, pp. 362-366, 2014.

[19] A. Sforzin, F. G. Mármol, M. Conti, and J. M. Bohli, "RPiDS: Raspberry Pi IDS—A fruitful Intrusion Detection System for IoT," In Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences, pp. 440-448, 2016.

[20] P. Velthuis, "Small data center using Raspberry Pi 2 for video streaming," Proc. 23th Twente Student Conf. IT, 2015.

[21] E. Wilcox, P. Jhunjhunwala, K. Gopavaram, and J. Herrera, "Pi-crust: a Raspberry Pi cluster implementation," Technical report, Texas A&M University, 2015.

[22] G. Tang, et al., "EMAN2: An extensible image processing suite for electron microscopy," Journal of Structural Biology, vol. 157, pp. 38-46, 2007.

[23] S. Schlüter, A. Sheppard, K. Brown, and D. Wildenschild, "Image processing of multiphase images obtained via X-ray microtomography: A review," Water Resour. Res. vol. 50, pp. 3615-3639, 2014.

[24] Q. Dai, J. H. Cheng, D.-W. Sun and X. A. Zeng, "Advances in feature selection methods for hyperspectral image processing in food industry applications: A Review," Critical Reviews in Food Science and Nutrition, vol. 55, no. 10, pp. 1368-1382, 2015.

[25] E. Hamuda, M. Glavin, and E. Jones, "A survey of image processing techniques for plant extraction and segmentation in the field," Computers and Electronics in Agriculture, vol. 125, pp. 184-199, 2016.

[26] M. Mora, et al., "Automated computation of leaf area index from fruit trees using improved image processing algorithms applied to canopy cover digital photography," Computers and Electronics in Agriculture, vol. 123, pp. 195-202, 2016.

[27] Raspberry Pi 3 Model B, available at: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ (last accessed on: November 2018)

[28] K. Iyer, "Learn to build your own supercomputer with Raspberry Pi 3 cluster", Post on TechWorm, 2018.

[29] A. K. Dennis, "Raspberry Pi super cluster", Packt Publishing, 2013.

[30] C. R. Morrison, "Build supercomputers with Raspberry Pi 3", Packt Publishing – ebooks Account, 2017.

[31] W. Gropp, E. Lusk, and A. Skjellum, "Using MPI: Portable parallel programming with the message – passing – interface," 3rd Edition, The MIT Press, 2014.

[32] R. Golden, "Raspberry Pi Networking Cookbook," Packt Publishing, 2013.

[33] V. Govindaraj, "Parallel programming in Raspberry Pi cluster," A Design Project Report, School of Electrical and Computer Engineering, Cornel University, 2016.

[34] G. Dorr et al, "Introduction to parallel processing with eight node Raspberry Pi cluster", Midwest Instruction and Computing Symposium (MICS), The University of Wisconsin – La Crosse in La Crosse, 7 – 8 April 2017.

[35] S. van der Walt, et al. "Scikit-image: image processing in Python," PeerJ 2:e453, 2014.

Dušan Marković was born in Serbia in 1982. He received Dipl. Ing. degree at the University of Kragujevac, Faculty of Technical Sciences in 2006. He is currently a PhD student in the field of Computer Science at the Faculty of Technical Sciences in Čačak. His research interests include distributed computing, IoT, and Cloud-Fog computing.

Dejan Vujičić was born in Čačak, Serbia in 1988. He received his B.Sc. and M.Sc. in Computer Science at the University of Kragujevac, Faculty of Technical Sciences in 2011 and 2012, respectively. He also received M.Sc. in Astronomy in 2016, at the Faculty of Mathematics, University of Belgrade. He is currently a PhD student in the field of Computer Science, at the Faculty of Technical Sciences in Čačak, where he is employed as a teaching assistant since 2014. His research interests include computer architecture, parallel computing, wireless sensor networks, neural networks, and applications of computer science in astronomy.

Dragana Mitrović was born in Belgrade, Serbia in 1994. She received B.Sc. and M.Sc. in Computer Science at the University of Kragujevac, Faculty of Technical Sciences in 2017 and 2018, respectively. She is now employed by Riitech Solutions in Čačak. Raspberry Pi Cluster presented in this article has been developed within her master thesis. The areas of her interest are parallel programming, WEB and mobile computing.

Siniša Ranđić was born in Čačak, Serbia in 1953. He received the Dipl. Ing. and M.Sc. degree at Faculty of Electrical Engineering University of Belgrade 1977, 1984 respectively. PhD degree received from University of Kragujevac, Technical faculty in Čačak, in 1999. Since 2018, he is a retired professor. His research interests include computer architecture, operating systems, parallel processing and design of VLSI circuits. He has more than 40 years of professional experience in the field of computer system design.