Parallel Technique for Medicinal Plant Identification System using Fuzzy Local Binary Pattern

N.N. Kutha Krisnawijaya[1,*], Yeni Herdiyeni[2] & Bib Paruhum Silalahi[3]

[1]Department of Electrical Engineer, Faculty of Engineer and Informatics, Undiknas University, Jalan Bedugul No. 39, Bali 80224, Indonesia
[2]Department of Computer Science, Faculty of Mathematics and Natural Sciences, Bogor Agricultural University, Darmaga Campus, Jalan Meranti, Wing 20 Level 5, Bogor 16680, Indonesia
[3]Department of Mathematics, Faculty of Mathematics and Natural Sciences, Bogor Agricultural University, Darmaga Campus, Jalan Meranti, Wing 20 Level 5, Bogor 16680, Indonesia
*E-mail: ngakankutha@undiknas.ac.id

Abstract. As biological image databases are growing rapidly, automated species identification based on digital data becomes of great interest for accelerating biodiversity assessment, research and monitoring. This research applied high performance computing (HPC) to a medicinal plant identification system. A parallel technique for medicinal plant image processing using Fuzzy Local Binary Pattern (FLBP) is proposed. The FLBP method extends the Local Binary Pattern (LBP) approach by employing fuzzy logic to represent texture images. The main goal of this research was to measure the efficiency of using the proposed parallel technique for medicinal plant image processing and evaluation in order to find out whether this approach is reasonable for handling large data sets. The parallel processing technique was designed in a message-sending model. 30 species of Indonesian medical plants were analyzed. Each species was represented by 48 leaf images. Performance evaluation was measured using the speed-up, efficiency, and isoefficiency of the parallel computing technique. Preliminary results show that HPC worked well in reducing the execution time of medical plant identification. In this work, parallel processing of training images was 7.64 times faster than with sequential processing, with efficiency values greater than 0.9. Parallel processing of testing images was 6.73 times faster than with sequential processing, with efficiency values over 0.9. The system was able to identify images with an accuracy of 68.89%.

Keywords: *fuzzy local binary pattern; high performance computing; parallel processing; MPI Library; image processing; plant identification.*

1 Introduction

Indonesia posesses a natural diversity of more than 38,000 plant species [1]. Groombridge and Jenkins [2] noted that there are 22,500 medicinal plant species in Indonesia. Approximately 1,000 plant species are being used for

Received February 22nd, 2016, Revised December 19th, 2016, Accepted for publication March, 30th, 2017.
Copyright © 2017 Published by ITB Journal Publisher, ISSN: 2337-5787, DOI: 10.5614/itbj.ict.res.appl.2017.11.1.5

medicinal purposes, which means that only 4.4% of all available medicinal plant resources are being benefited from. The reason is that the people lack knowledge about medicinal plants. For them to acquire more knowledge about medicinal plants could be aided by improving existing medicinal plant identification systems.

There are many researches on medicinal plant identification systems based on medicinal plant feature extraction. The simplest identification system using vegetative organs is based on plant leaves. Valerina [3] used the Fuzzy Local Binary Pattern (FLBP) feature extraction method and the Probabilistic Neural Network (PNN) method to identify medicinal plants. Herdiyeni and Wahyuni [4] also used a feature extraction method and the same classification to identify medicinal plants. Their identification system was developed on a Android based mobile application. The system had an accuracy of 74.51% with identification computation time at 12 seconds per image. Laxmi [5] implemented the Multiobjective Genetic Algorithm (MOGA) to identify medicinal plants. Application of the FLBP method for feature extraction had the longest computation time on the system used. This is because FLBP is an extraction method that uses fuzzification to gain texture patterns from the images. The more fuzzy length is used, the more image pixels will be processed, which extends the computation time needed. This process has an accuracy of 81.21% with computation time at 25 seconds per image.

The problem of computing duration when using FLBP feature extraction on a medicinal plant identification system motivated the idea of implementing a high performance computing (HPC) method. The use of HPC is expected to reduce computing duration when running a medicinal plant identification system. The measurement of HPC uses speed-up and efficiency techniques. According to Petryniak [6], applying several parallel architectures for image processing can increase the efficiency of digital image processing.

In this research, a HPC method was applied to reduce the computation time of the FLBP feature extraction process. We propose a parallel architecture with division of the image data between processors. Computational time is expected to be reduced by extracting leaf images on each processor simultaneously.

2 Fuzzy Local Binary Pattern

Referring to Iakovidis [7], Local Binary Pattern (LBP) represents the local texture around the environment of a central texture, based on the LBP neighborhood operator. Each LBP texture pattern is represented by nine elements of $P = \{P_{center}, P_0, P_1, \ldots, P_7\}$, where P_{center} is the value of its surrounding pixels (circular sampling). The value of circular sampling can be tagged by binary value $d_i(0 \le i \le 7)$. The resulted binary value then will be converted into a decimal value to acquire the value of LBP using Eq. (1):

$$LBP = \sum_{i=0}^{7} d_i \cdot 2^i \quad , \ LBP \in [0,255] \tag{1}$$

The values of LBP that are generated are represented in a histogram. The histogram shows the frequency of each LBP value that appears. According to Ahonen *et al.* [8], the LBP operator can develop a certain sampling point and radius. For the observed neighborhood pixels, the notation (P,R) is employed, with P as the sampling point and R as the radius. The values of LBP are generated based on the LBP operator used. The smaller the radius and the larger the sampling points used, the more pixels will be processed to acquire the values of LBP.

Fuzzification in the LBP approach is the transformation of input variables into fuzzy variables with regards to certain fuzzy rules. Based on the research of Iakovidis [7], this research also used two fuzzy rules to determine the representatation of the binary values and to seek the fuzzy values. The determination of the fuzzy values is based on the difference in description between circular sampling p_i and center pixel p_{center}. Based on Iakovidis [7], those two rules are:

1. Rule R0: if the value of Δp_i is negative, then the largest exact value of d_i is 0.
2. Rule R1: if the value of Δp_i is positive, then the largest exact value of d_i is 1.

Based on rules R_0 and R_1, two membership functions, $m_0(i)$ and $m_1(i)$, can be defined. Membership function $m_0(i)$ defines the degree of d_i as 0. Membership function $m_0(i)$ is the differential function that is defined in Eq. (2),

$$m_0(i) = \begin{cases} 0 & , if \Delta p_i \ge F \\ \frac{F+\Delta p_i}{2.F} & , if - F < \Delta p_i < F \\ 1 & , if \Delta p_i \le -F \end{cases} \tag{2}$$

Membership function $m_1(i)$ defines the degree of d_i as 1. Membership function $m_1(i)$ is the differential function that is defined in Eq.(3),

$$m_1(i) = \begin{cases} 1 & ,if \Delta p_i \geq F \\ \frac{F+\Delta p_i}{2.F} & ,if -F < \Delta p_i < F \\ 0 & ,if \Delta p_i \leq -F \end{cases} \tag{3}$$

Rules R_0 and R_1 represent Fuzzy Local Binary Pattern (FLBP) threshold (F), which controls the degree of uncertainty. The higher the value of the threshold, the higher the number of pixels that will be processed inside the fuzzy range. The use of a high threshold affects the computing duration at the time of the feature extraction process.

The FLBP method results in one or more LBP codes, while the original LBP method only results in one LBP code. The LBP values generated by FLBP have a different C_{LBP} contribution level, depending on the values of the generated membership function. An illustration of the FLBP process is given in Figure 1.

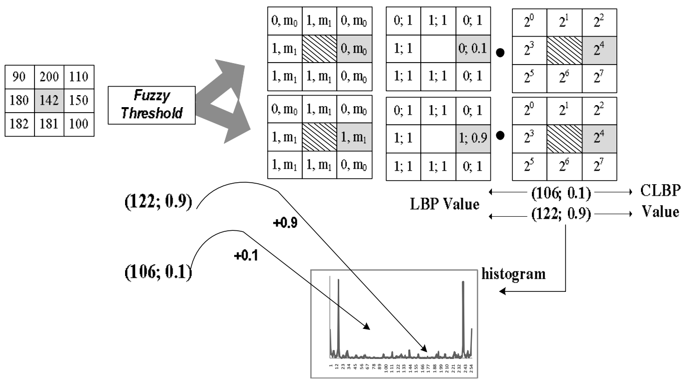

Figure 1 FLBP computing scheme with $F = 10$.

3 High Performance Computing

HPC can be used to solve high complexity problems, e.g. a complicated algorithm, considering the workload for a large volume of data. HPC is able to reduce the computing duration when it is affected by high complexity, so that the work of the system becomes more efficient and the resulting information will be gathered faster. Parallel and distributed computing are the two main techniques that are applied in the HPC. HPC has been developed and is largely used for bio-information, image processing, and other fields. The parallel architecture is divided into three parts [9], i.e.:

1. *Parallel processing hardware (shared memory)* – an architecture that uses one computer with several processors (multiprocessors). In this sort of

architecure, two or more computer processors execute the computations together and access the same memory.
2. *Distributed sytem* – an architecture that uses more than one computer (multi-computer) connected with a network and working in parallel. The difference between shared memory and distributed memory is that each processor has its own local memory and the computations are executed in each of the local memories. Distributed memory requires network communication to link the memories of the separate processors.
3. *Hybrid system* – a combination of distributed memory and shared memory. Recently, the development of this type of architecture is preferred for generating faster and larger systems.

Referring to Quinn [10], Foster in 1995 devised a new method to design a parallel system. The method starts with partitioning data (computing), determining the communication of the intersection (communicating). If an observed intersection communication occurs that is too big, then two sections that have more intensive communication with each other will be grouped together (agglomerating). The last step is mapping the groups onto the processors.

According to Quinn [10], the acceleration of parallel processing not only depends on the number of processors used. The process is also influenced by the fraction ratio between the sequential computing instruction and the whole instruction of a program. If the speed-up value is equal to the total number of processors used, then the parallel architecture works optimally. As stated by Grama, *et al.* [11], the ideal speed-up is gained when the speed-up value is equal to the total number of processors used.

Apart from the speed-up calculation, the efficiency is taken into account to measure parallel program performance. Efficiency is measured by looking at the use of the processors in a parallel architecture. If the efficiency value approaches 1, it means that there are no idle processors, i.e. the processors work optimally.

Quinn [10] states that besides using Amdhal's law, parallel performance may also be measured by calculating the scalability (isoefficiency). Scalability is the performance of a parallel architecture to maintain efficiency values in response to additional data and processors. The measurement of isoefficiecy is aimed at determining the total number of effective running processors with a certain data workload.

4 The Proposed Parallel Technique

In this research, images of medicinal plant leaves in Indonesia were used. The number of images used was 1440 images, representing 30 medicinal plant species in Indonesia. Each of the medicinal plant species was represented by 48 images. The size of every image was 240×270 pixels. According to Prajapati and Vij [12], application of parallel processing of digital images can be done with parallel computing. Parallel processing of digital images can be done by dividing the data between the processors. The ideal division is to send the same amount of data to each processor.

In this study, the FLBP method was used for parallel computing, implemented using the C++ programming language. The parallel technique concept was developed with a message-sending model using MPI Library.

The parallel program was implemented on a cluster computer consisting of 8 computer units (called-on processors). The computers that were used all had the same specification, i.e. Intel Core-i3 processor CPU, 3.10 Gigahertz, 4 Gigabytes RAM, 500 Gigabytes harddisk, running the Linux Ubuntu 12.04 operating system. The processors were connected with a LAN cable with 100 Megabytes per second speed and a switch unit. The latency was used to measure the average time needed by a master processor to send data to the slave processors or in the opposite direction, inside the cluster architecture.

Sequential FLBP. The FLBP feature extraction method applied in this research uses two operators, i.e. the $(8,1)$ and the $(8,2)$. Each operator had a threshold ranging from 1 to 10. The process of the medicinal plant feature extraction was run by using a single processor only. The following code is the pseudocode for the FLBP feature extraction used in this research. Figure 2 shows the algorithm of the sequential process.

```
for(a number of species) {
    for(a number of leaf images){
        preprocessing(RGB to Grey scale)
        for(a number of operators){
            for(a number of thresholds){
              output_file = FLBP(leaf image)
                SaveFile(output_file)
            }end for
        }endfor
    }endfor
}endfor
```

Figure 2 Algorithm of the sequential process.

The use of the operator and the threshold did influence the computing duration of the FLBP feature extraction. The time needed to extract features from 1440 images was 29,131 seconds. The medicinal plant identification system needed 25 seconds to identify one image.

Parallel FLBP. The parallel design in this research used distributed memory combined with data decomposition. This architecture design implemented the Foster method to design a parallel method.

The partition stage was carried out to search independent tasks in the process of FLBP feature extraction. An independent task is a process that has no relation to other processes. The process of feature extraction from an image is an independent task, therefore parallel processing can be done by dividing the leaf images to be extracted between different processors. The division work was carried out by the master processor, by dividing and distributing the leaf image data to a number of processors. Thus, in an ideal condition, the processor extracts $^{N}/_{p}$ leaf image data, where N is the total number of leaf image data and p is the total number of processors.

The next stage after partitioning is communicating. The FLBP feature extraction algorithm applied the following steps: dividing the image data between processors, extracting features by using FLBP on each processor, and collecting the histogram of the feature extraction result from all slave processors to the master processor. The following code is the pseudeocode of the FLBP parallel feature extraction. Figure 3 shows the algorithm of the parallel process.

```
for(a number of species) {
  scatter the leaf images from master to slave
  preprocessing leaf image at each processor
      for(a number of operators){
          for(a number of thresholds){
             output_file = FLBP(leaf image)
             SaveFile(output_file)
          }end for
      }endfor
      gather output_file from slave to master
}endfor
```

Figure 3 Algorithm of parallel process.

Based on the parallel algorithm above, it is inferred that the communication process is held at the time of the scattering process of the leaf images and the gathering process of the feature extraction result histogram. The scattering process uses the operation *MPI_Scatterv* (Figure 4(a)), which was preserved by

Library MPI, and the gathering process uses the operation *MPI_Gatherv* (Figure 4 (b)).

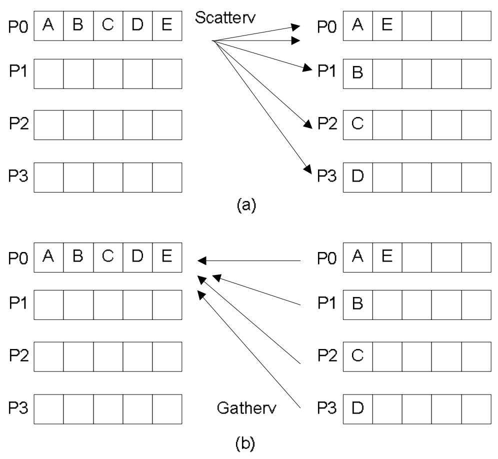

Figure 4 Communicating process using (a) *MPI_Scatterv* and (b) *MPI_Gather*.

MPI_Scatterv is a communication method to distribute different data amounts to different processors. The inverse of the *MPI_Scatterv* process is the *MPI_Gatherv* process. *MPI_Gatherv* was used to collect the histogram of the feature extraction result. The process of histogram collecting happens from the slave processor to the master processor. The parallel design used did not reach the agglomeration stage and the mapping stage, because the communication process only required minimal processing and therefore this design did not need agglomeration and mapping.

5 Results and Discussions

The parallel technique proposed in this study was implemented on a cluster architecture. The cluster architecture used a local area network as the networking system. On the network only communication from the processors took place, without communication from any other network type. The result of the latency measurement on the cluster architecture was 0.113 milliseconds. This means that the resulted latency value was very good. The lower the latency value of a cluster achitecture, the better performance is gained.

5.1 Evaluation of Parallel Performance for Training Images

Speed-up. The evaluation result is shown in Table 1 and Figure 5. Accelerating the performance of the FLBP feature extraction process achieved an ideal result, as was shown by a comparison between the parallel duration and the sequential computing duration.

Table 1 Result of Speed-up Evaluation and Efficiency of Parallel FBP.

Total number of the processors (p)	**Parallel time/duration** (T_{par})	**Speed-up** (S)	**Efficiency** (E)
2	14742	1.97	0.97
3	9955	2.92	0.97
4	7468	3.90	0.97
5	6249	4.66	0.93
6	5062	5.75	0.95
7	4428	6.57	0.93
8	3812	7.64	0.95

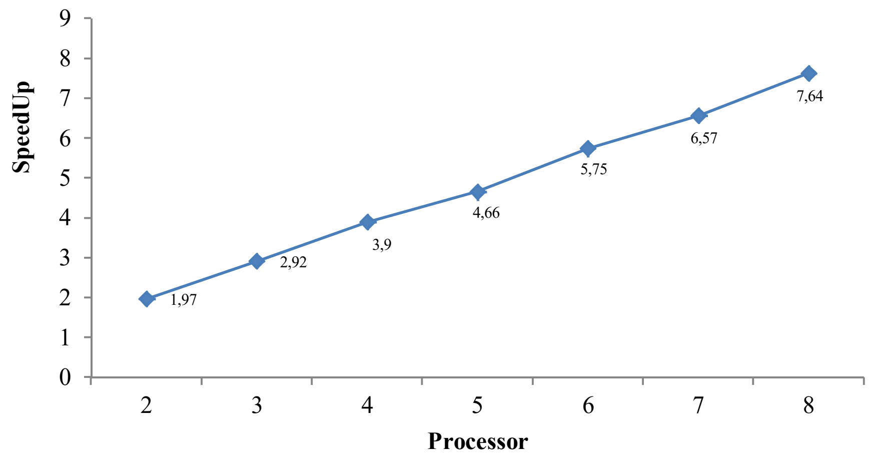

Figure 5 Parallel architecture speed-up chart.

The use of multiprocessors, as shown in Figure 5, was able to reduce the duration of the medicinal plant leaf image feature extraction. The experiment using 2 processors took 14,472 seconds, which means a speed-up of up to 1.97 times compared with the sequential computing time. The experiment using 8 processors took 3,812 seconds with a speed-up of up to 7.64 times compared with the sequential computing time. The speed-up that was achieved in all experiments approached the ideal condition, i.e. each of the experiments resulted in speed-up values (S) that approached the total number of processors used (p).

Efficiency. The efficiency was affected by the communicating and computing processes in the parallel program. The communicating process involves the partitioning and distribution of the 48 images per species from the master processor to the slave processors. The computing process involves the processing of the division result of the data extraction result. The efficiency measurement was carried out to monitor how optimal the process was that worked inside the architecture used, as shown in Figure 6.

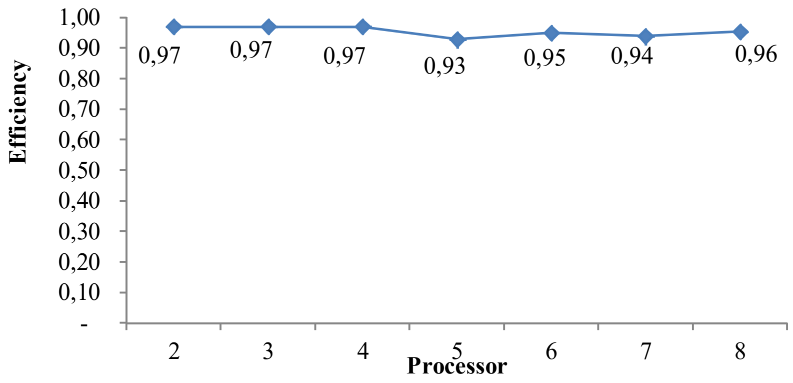

Figure 6 Parallel architecture efficiency chart.

The efficiency value produced on the parallel architecture approached the ideal value (Figure 6). The highest efficiency value was 0.97 at the time of the use of processors 2, 3, and 4. When 5 processors were used, the efficiency value decreased to 0.93.

The decline of this efficiency value was caused by the result of data partitioning not being ideal. To process 48 image data for every species, there were 3 processors that processed 10 image data, while the other 2 processors processed 9 image data.

This image data partitioning put 2 processors into an idle condition, while the other 3 processors were processing data, which influenced the efficiency value of the parallel processing. However, the decline of this efficiency value was not significant. The use of multiprocessors to extract features from 1440 medicinal plant leaf images can be considered ideal enough.

Isoefficiency. The number of processors in a parallel program architecture is precisely calculated. The use of several processors affects the communication cost. Effective use of the processors can reduce the communication cost spent by the parallel architecture. The measurement of isoefficiency is shown in Table 2.

Table 2 Result of Isoefficiency

n	p=2	p=3	p=4	p=5	p=6	p=7	p=8
540	0.97	0.97	0.88	0.87	0.95	0.82	0.72
600	0.99	0.94	0.97	0.97	0.83	0.90	0.80
720	0.98	0.97	0.96	0.92	0.95	0.82	0.83
840	0.97	0.93	0.97	0.90	0.89	0.94	0.83
900	0.99	0.99	0.92	0.97	0.96	0.84	0.90
1020	0.99	0.94	0.93	0.95	0.92	0.80	0.84
1140	1.00	0.96	0.93	0.93	0.88	0.88	0.90
1200	1.00	0.94	0.98	0.97	0.92	0.92	0.96
1320	0.89	0.97	0.99	0.96	0.90	0.88	0.89
1440	0.99	0.98	0.98	0.93	0.96	0.94	0.96

This table shows the efficiency values from the set of experiments using different numbers of image data and processors. The efficiency tended to decrease with the provision of extra processors. The efficiency can be increased by adding image data to the parallel architecture. The experiment in which the number of processors and the number of image data were increased could keep the efficiency values constantly above 0.95. This shows that the parallel architecture used was stable and scalable.

5.2 Evaluation of Parallel Performance for Testing Images

The developed identification system was als designed to accept input images from a user. Features will be extracted from these images and the extraction result is in the form of an FLBP histogram. The testing images feature extraction process was conducted using the parallel architecture. The partition process was performed by dividing and distributing the FLBP operator combination and threshold towards each processor. Each processor extracted testing images using a different combination. The communication process used for extracting testing images was similar to the communication process used for extracting leaf medicinal plant features.

Evaluation of parallel performance on testing image extraction was done based on speed-up and efficiency. The measurement of speed-up was performed by comparing the sequential time and parallel time needed for extracting the testing images. The sequential time for extracting one testing image was 24.9 seconds.

Speed-up. Table 3 and Figure 7 show that parallel processing of the testing images was able to accelerate the feature extracting process. The speed-up reached the ideal condition when using 2 to 5 processors.

Table 3 Result of Speed-up Evaluation and Efficiency for Testing Images

Number of *processors* (p)	**Parallel time** (T_p) **(seconds)**	***Speed-up*** (S)	***Efficiency*** (E)
2	12.1	2.06	1
3	8.3	3.00	1
4	6.1	4.08	1
5	4.9	5.08	1
6	4.8	5.19	0.86
7	3.7	6.73	0.96
8	3.7	6.73	0.84

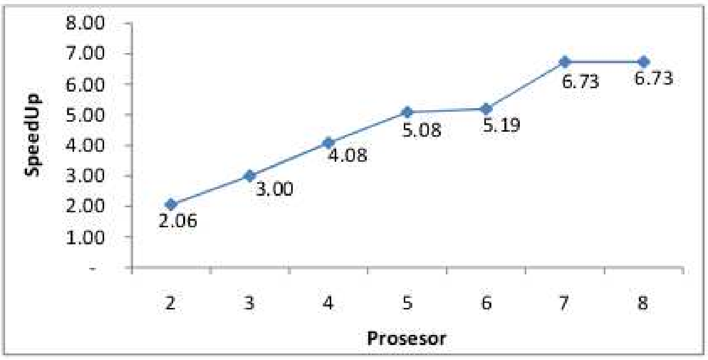

Figure 7 Parallel architecure speed-up for testing image extraction.

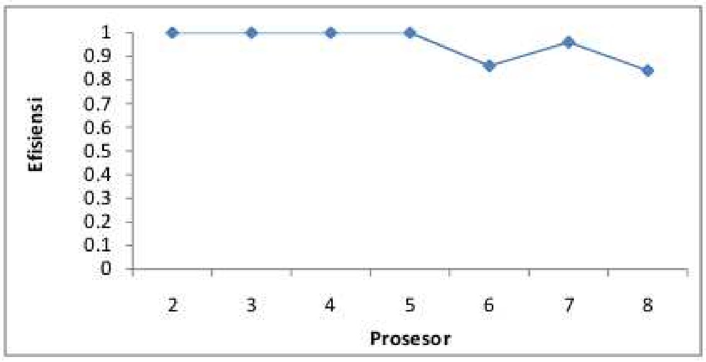

Figure 8 Parallel architecure efficiency for testing image extraction.

Efficiency. The measurement of efficiency when extracting features from testing images reached the ideal condition (Figure 8) when using 3, 4, and 5 processors. In the trial, using 6 or 8 processors made the efficiency values decrease. This is due to processors in idle condition at the time parallel processes took place.

5.3 Classification and Evaluation

Feature extraction on the parallel architecture was divided into training data and testing data at a respective percentage of 80% and 20%. 1140 images were used as training data and 300 images were used as testing data. Classification using PNN classifier showed that the amount of images that were successfully identified was 206 out of all testing images, thus the obtained accuracy was 68.89%

6 Conclusions

In this study, parallel processing of training images and testing images was conducted. A parallel architecture for training images implemented parallel computing for the process of partitioning leaf image data to several processors. Feature extraction was carried out on each processor. The highest measured speed-up (S) was up to 7.64 faster than sequential feature extraction. Overall, the efficiency value (E) was kept above 0.9, which means that the parallel architecture that was used worked optimally and efficiently. Parallel processing of the testing images was 6.73 faster than sequential processing, with efficiency values above 0.9. The accuracy of identifying images was 68.89%. In sequential FLBP using MOGA, the accuracy was about 81.21%, but this research focused on parallel speed-up and efficiency evaluation.

In a follow-up research more processors could be added to the parallel architecture used. The implementation of parallel computing for feature extraction by using FLBP is also recommended for future research. It is also important to increase the accuracy of image identification.

References

[1] The Bureau of National Planning and Development, Indonesia, *Biodiversity and Action Plan 2003-2020*, Jakarta, 2003.

[2] Groombridge, B. & Jenkins, M., World Atlas of Biodiversity. *Earth's Living Resources in the 21st Century*, Berkeley University of California Press, 2002.

[3] Valerina, F., *Comparison of Local Binary Pattern and Fuzzy Local Binary Pattern for Tropical Medicinal Plant Extraction,* Undergraduate Thesis, Departement of Computer Science, Faculty of Mathematics and Natural Sciences, Bogor Agricultural University, Bogor, 2012. (Text in Indonesian)

[4] Herdiyeni, Y. & Wahyuni, N.K.S., *Mobile Application for Indonesian Medicinal Plants Identification using Fuzzy Local Binary Pattern and Fuzzy Color Histogram*, International Conference on Advanced Computer Science and Information System, December 1^{st}-2^{nd}, 2012 Jakarta, Indonesia, 2012.

[5] Laxmi, G.F., *Optimization of Fuzzy Local Binary Pattern in Threshold and Operator Selection using Multi Objective Genetic Algorithm*, Master Thesis, Departement of Computer Science, Faculty of Mathematics and Natural Sciences, Bogor Agricultural University, Indonesia, 2012. (Text in Indonesian)

[6] Petryniak, R., *Analysis of Efficiency of Parallel Computing in Image Processing Task,* Czasopismo Techniczne, pp. 185-193, 2008.

[7] Iakovidis, D.K., Keramidas, E.G. & Maroulis, D., *Fuzzy Local Binary Patterns for Ultrasound Texture Characterization*, ICIAR, LNCS 5112, pp. 750-759, 2008.

[8] Ahonen, T., Hadid, A. & Pietikainen, M., *Soft Histogram for Local Binary Patterns*, CCV 2004, LNCS 3021, pp. 469-481, 2004.

[9] Nasir, A.F.A., Rahman, M.N.A. & Mamat, A.R., *A Study of Image Processing in Agriculture Application under High Performance Computing Environment*, International Journal of Computer Science and Telecommunications, **3**, pp. 16-24, 2012.

[10] Quinn, M.J., *Parallel Programming in C with MPI and Open MP*, McGraw-Hill Education, Singapore, 2004.

[11] Grama, A., Gupta, A., Karypis, G. & Kumar, V., *Introduction to Parallel Computing,* Pearson Education Limited, England, 2003.

[12] Prajapati, H.B. & Vij, S.K., *Analytical Study of Parallel and Distributed Image*, International Conference on Image Information Processing (ICIIP), November 3^{rd}-5^{th}, India, 2011.