농업 이미지 처리를 위한 빅데이터 플랫폼 설계 및 구현

반퀴엣뉘엔, 신응억뉘엔, 둑티엡부, 김경백
전자컴퓨터공학부, 전남대학교
e-mail : kyungbaekkim@jnu.ac.kr

Design and Implementation of Big Data Platform for Image Processing in Agriculture

Van-Quyet Nguyen, Sinh Ngoc Nguyen, Duc Tiep Vu, Kyungbaek Kim
Dept. of Electronics and Computer Engineering, Chonnam National University

요 약

Image processing techniques play an increasingly important role in many aspects of our daily life. For example, it has been shown to improve agricultural productivity in a number of ways such as plant pest detecting or fruit grading. However, massive quantities of images generated in real-time through multi-devices such as remote sensors during monitoring plant growth lead to the challenges of big data. Meanwhile, most current image processing systems are designed for small-scale and local computation, and they do not scale well to handle big data problems with their large requirements for computational resources and storage. In this paper, we have proposed an IPABigData (Image Processing Algorithm BigData) platform which provides algorithms to support large-scale image processing in agriculture based on Hadoop framework. Hadoop provides a parallel computation model MapReduce and Hadoop distributed file system (HDFS) module. It can also handle parallel pipelines, which are frequently used in image processing. In our experiment, we show that our platform outperforms traditional system in a scenario of image segmentation.

1. Introduction

There are many applications of image processing in agriculture such as plant pest detection and fruit grading [1]. The image processing techniques can be used to enhance agricultural practices, by improving accuracy and consistency of processes while reducing farmer's manual monitoring. Often, it offers flexibility and effectively substitutes the farmer's visual decision making. In recent years, however, with the rapid growth of agriculture, a large amount of image data has been accumulating from greenhouses. When processing this massive data resource has been limited to single computers, computational power, and storage ability quickly become bottlenecks. Alternately, processing tasks can typically be performed on a distributed system by dividing the task into several subtasks. The ability to parallelize tasks allows for scalable, efficient execution of resource-intensive applications.

Recently, there are several researchers focused on Hadoop/MapReduce [2] platform which provides a system for computationally intensive data processing and distributed storage. There are three main frameworks that designed for image processing in Hadoop: HIPI (Hadoop Image Processing Interface) [3], OpenIMAJ (Open Intelligent Multimedia Analysis for Java) [4], and MIPr (Mapreduce Image Processing) [5]. HIPI is a framework that is specifically designed to enable image processing in Hadoop. OpenIMAJ is a set of Java libraries for image and video analysis, some of OpenIMAJ tools have Hadoop implementation. The MIPr framework provides the image representations in the internal Hadoop formats, the input/output tools for image processing integration into Hadoop data workflow, and the image processing API for developers who are not familiar with Hadoop. However, these frameworks are required to modify the image storage such as HIP files in HIPI framework, which creates additional overhead in programming. In our work, we get automatically the images from the remote sensors and store them in HDFS, in which there is no additional programming overhead for users to handle image storage.

In this paper, we have developed a Hadoop-based system with the aims of providing IPABigData platform specific enough to contain a relevant framework applicable for image processing in agriculture. We use Hadoop because this framework is designed to store and process big data on large-scale distributed systems with simplified parallel programming models.

Our work makes the following contributions:

- First, we proposed a new design of big data platform for image processing by using Hadoop. The platform provides the modules for processing the image data.
- Second, we implemented algorithms in MapReduce for image processing which are used to interpret the image content such as image grayscale and image segmentation, in parallel.
- Final, the experimental results showed that processing big image dataset of our platform is faster than a traditional system. The results also show the viability of application of our system in agricultural data.

2. Background

2.1. Image Processing Algorithms

We consider implementation of multiple variations of widely-used current image processing algorithms which are essential for data analysis in agriculture such as plant pest detection and fruit grading.

1) Image Histogram

A histogram is an array of numbers in which each element, bin, corresponding to the frequency of a range of values in the given data [6]. For instance, each bin counts the number of pixels having the same color values in the case of an image histogram. Thus, a histogram is a mapping from the set of data values to the set of non-negative real numbers.

2) Image Segmentation using Otsu's method

Image segmentation is the process of partitioning a digital image into multiple segments (sets of pixels). Each of the pixels in a region is similar with respect to some characteristic or computed property, such as color, intensity, or texture. Adjacent regions are significantly different with respect to the same characteristics.

Otsu's method is used to automatically perform clustering-based image thresholding or the reduction of a gray level image to a binary image [7]. The idea of Otsu's method is to find out a threshold which maximizes the between-class variance. This idea can be represented by the algorithm as follows.

Step 1: Compute histogram and probabilities of each intensity level

Step 2: Set up initial class probability $w_i(0)$ and class mean $\mu_i(0)$ with threshold t = 0

Step 3: Loop through all possible thresholds from t = 1 to maximum intensity

-Update $w_i(t)$ and $\mu_i(t)$, and compute class variance $\sigma_b^2(t)$

Step 4: Get threshold corresponding to the maximum $\sigma_b^2(t)$

2.2. Hadoop/MapReduce

Hadoop consists of two components of HDFS and MapReduce, which are respectively the implement of Google's GFS and MapReduce. MapReduce is a programming model that supports to run programs in parallel on large distributed system. This model uses a map function that processes a key/value to generate a set of intermediate key/value pair and a reduce function that gathers all values with the same intermediate key to process and returns the results. A MapReduce program is automatically parallelized and executed on a large cluster of commodity machines. A MapReduce job usually splits the input dataset into independent chunks that are processed by map tasks. The outputs of map tasks are sorted, then they are used as inputs to reduce tasks. Typically, both of input and output of a MapReduce job are stored in the HDFS. The job is finished when all map tasks and reduce tasks are completed.

2.3. Image Processing and Hadoop

In this section, we present how to apply Hadoop/MapReduce for image processing through an example in which the image histogram calculation is performed in parallel and distributed manner. In this example, we consider to 5x5 pixel of a grayscale image as shown in Figure 1.

The original image has three *intensity levels* in histogram including *i1, i2, i3*. To calculate the histogram for this image using MapReduce, the original image is spliced into sub-images and store on HDFS. In this case, we have five sub-image from FileSplit1 to FileSplit5. Then, we perform three steps as shown in Figure 2.

i3	i1	i1	i1	i3
i1	i3	i2	i3	i1
i1	i2	i3	i2	i1
i1	i3	i2	i3	i1
i3	i1	i1	i1	i3

Figure 1. A 5x5 pixel grayscale image

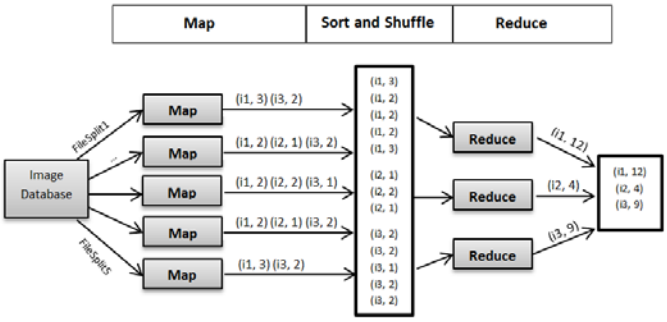

Figure 2. Image histogram calculation using MapReduce

Step 1: Each file is read, then calculate the intensities by Map Task. The input for each Map Task is a pair *(key, value)*, in which key is identified by file name and value is the content of sub-image. The output of Map function is the list of *(key',value')*, in which *key'* is the intensity level and *value'* is the number of pixels corresponds to the intensity level *key'*.

Step 2: It collects all of (key',value') pairs of Map tasks, then sort and shuffle by *key'* value. The pairs with the same *key'* usually are gathered belong to a group which will be processed by the same Reduce Tasks.

Step 3: In the reduce phase, the input is the output of the combiner in *Step 2*, each Reduce Task has a different key. In this example, the key for each reduce task is chosen corresponds to each intensity level in the image. This phase performs a calculation total of pixels for each intensity level. By assembling the output of each reduce task, we can get the final result that is a histogram of the original image.

3. Design and Implementation of IPABigData Platform

3.1. The Architecture of IPABigData platform

Our goal is to build a Big Data platform for plant pest detecting. The platform is designed to run on Hadoop environment. The IPABigData's overall architecture includes three mains layers as shown in Figure 3. The lowest layer as the infrastructure layer that includes multi-cluster (nodes) to build a Hadoop-based system for storing and processing data. The second layer, Image Processing Algorithms, provides the algorithms that apply well-known techniques for image processing such as histogram calculation, and image segmentations. This layer includes three components: Low-

level Image Processing, High-level Image Processing, and Image Data Analysis. The third layer is a Decision Making Support System, which provides some suggestions to user for find out some plant pests or fruit grading based the information of Image Data Analysis in the second layer.

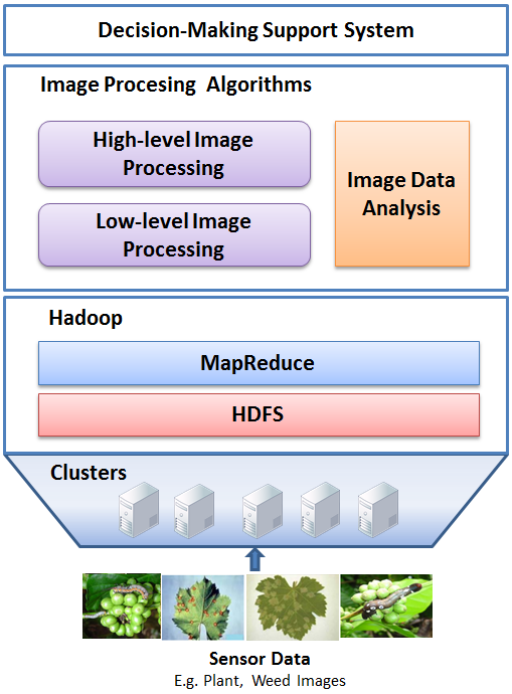

Figure 3. The architecture of IPABigData platform.

3.2. Implementation of Input Format

In Hadoop, the input data need to be processed by *InputFormat* class at first and then pass to each mapper through the standard input. The *InputFormat* class in Hadoop is used to handle input data for Map/reduce job, which need to be customized for different data formats. The *InputFormat* class describes the input data format and define how to split the input data into *InputSplits* buffer, which will be sent to each mapper. Another class, *RecordReader,* is called by mapper to read data from each *InputSplit*.

For image processing in Hadoop, we implement ***ImageFileInputFormat*** class to extend ***FileInputFormat*** class in Hadoop, which returns 'false' in isSplitable and creates ***ImageFileRecordReader*** class instance in get ***RecordReader***. ImageFileRecordReader will create Key/Value pair for mapper and read the whole content of input image file actually.

3.3. Implementation of Output Format

The OutputFormat interface determines how the results of a MapReduce job are stored. In Hadoop framework, there are several classes and interfaces with different types of formats, and customization is done by extending one of these. The default class of OutputFormat is TextOutputFormat in which the lines are separated and a tab character is used to separate the key/value pair. In this paper, after processing data, the results are image files, because of that we have implemented an ***ImageFileOutputFormat*** class extends ***FileOutputFormat*** class.

3.4. Implementation of Image Processing Algorithms

In our platform, image processing operations can be divided into two levels including low-level image processing and high-level image processing.

1) Low-level Image Processing Algorithms

The low level is also called image pre-processing that operates at the pixel level [8]. The input to low-level image processing operators is an image whereas the output is either image or data. Few examples of low-level image processing operators are contrasted enhancement, noise reduction, and noise removal in an image. They are also used for edge detection and various image transformations or calculation of simple characteristics such as contours histograms.

2) High-level Image Processing Algorithms

The high-level image processing operations operate in order to generate higher abstractions [9]. They work on abstractions derived from intermediate-level image processing operators. They are used to interpret the image content such as classification and object recognition.

4. Evaluation

4.1. Evaluation Settings

The settings for experiments used to execute algorithms in this paper are described as following.

Environment setting. We deploy our framework with five machines: one machine for master node, and four others for compute nodes. Each compute node has 4 physical CPU cores and 8GB of RAM. All algorithms are implemented in Java.

Image Dataset. Our experiment uses the data related to crop/weed images which are collected from the Internet. To show the performance of our platform with big image data, we prepare multiple dataset of images with a different number of images and volume as Table 1.

Table 1. The various size of image dataset

Number of images	Volume (MB)
100	153
200	301
400	590
600	891
800	1229

4.2. Evaluation Results

In this section, we present experimental results of using our platform for image segmentation of agriculture image dataset as described above. For image segmentation, in our platform, we implemented (1) an algorithm for converting the color image to grayscale image and (2) Otsu's algorithm for the binarizing image, which are in Low-level Image Processing module. We also implemented a program in Java which conducts the same algorithms in a single machine, and we named using this program as Local-based approach through this evaluation. Then, we run the experiment and get the results as shown in Figure 4.

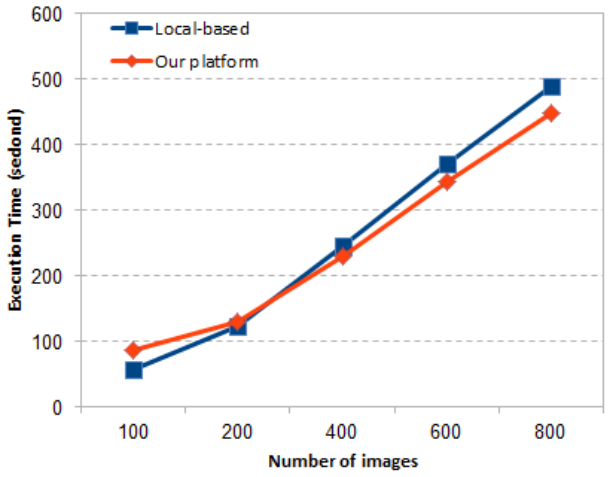

Figure 4. Comparison of execution time between IPABigData platform based on Hadoop (24 CPU cores) and a Local-based approach.

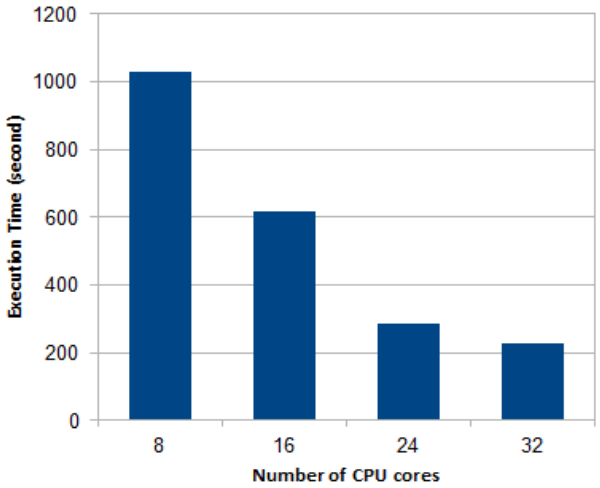

Figure 5. Efficient of increasing number of CPU cores on IPABigData platform based on Hadoop

Figure 4 shows the results of execution time in the experiment. With a small image dataset (100 images), the execution time of Local-based approach is smaller than our platform. Because the system with Hadoop needs to read/write the data at the initial state of the whole process, so it spends more time to setup the process. However, with a large image data set more than 400 images, our IPABigData platform executes faster than the Local-based approach. It shows that our platform is more scalable in aspects of the volume of input datasets.

To evaluate the efficiency and scalability, we varied the number of CPU cores of our system for conducting map tasks and reduce tasks from 8 to 32. In this experiment, we ran our platform for image segmentation algorithm with 400 images. We observed that the execution time decreases dynamically when we increase the number of CPU, as shown in Figure 5. The execution time with 8 CPU cores takes more 1000 seconds; meanwhile, the execution time with 32 CPU cores decreases down to 229 seconds. That is, our platform is much more efficient and scalable than the Local-based approach in the aspects of processing power.

5. Conclusion

The massive quantities of images generated in real-time through multi-devices during monitoring of plant growth in agriculture leads to the challenges of big data. In this paper, we have proposed a Hadoop based IPABigData platform and implemented map/reduce compatible algorithms to support large-scale image processing in agriculture. Our experiment showed that IPABigData platform outperforms traditional local computing based approach in the aspects of execution time of conducting image segmentation of large image data set. In future works, we will focus on improving the performance of our proposed platform, and provide more algorithms for image processing in agriculture.

Acknowledgment

This work was carried out with the support of "Cooperative Research Program for Agriculture Science and Technology Development (Project No. PJ01182302)" Rural Development Administration, Republic of Korea. This research was supported by the MSIP(Ministry of Science, ICT and Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-R2718-16-0011) supervised by the IITP(Institute for Information & communications Technology Promotion).

References

[1] Vibhute, Anup, and S. K. Bodhe. "Applications of image processing in agriculture: a survey." International Journal of Computer Applications 52.2 (2012).

[2] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.

[3] Sweeney, Chris, et al. "HIPI: a Hadoop image processing interface for image-based mapreduce tasks." Chris. University of Virginia (2011).

[4] Hare, Jonathon S., Sina Samangooei, and David P. Dupplaw. "OpenIMAJ and ImageTerrier: Java libraries and tools for scalable multimedia analysis and indexing of images." Proceedings of the 19th ACM international conference on Multimedia. ACM, 2011.

[5] Sozykin, Andrey, and Timofei Epanchintsev. "MIPr-a framework for distributed image processing using Hadoop." Application of Information and Communication Technologies (AICT), 2015 9th International Conference on. IEEE, 2015.

[6] Ahmed, Khidir Elhadi Bala, Rania A. Mokhtar, and Rashid A. Saeed. "A new method for fast image Histogram calculation." Computing, Control, Networking, Electronics and Embedded Systems Engineering (ICCNEEE), 2015 International Conference on. IEEE, 2015.

[7] Otsu, Nobuyuki. "A threshold selection method from gray-level histograms."Automatica 11.285-296 (1975): 23-27.

[8] Nicolescu, Cristina, and Pieter Jonker. "Parallel low-level image processing on a distributed-memory system." International Parallel and Distributed Processing Symposium. Springer Berlin Heidelberg, 2000.

[9] Bräunl, Thomas, et al. Parallel image processing. Springer Science & Business Media, 2013.