Proceedings of the 9th International Conference on Applied Informatics
Eger, Hungary, January 29–February 1, 2014. Vol. 1. pp. 93–101
doi: 10.14794/ICAI.9.2014.1.93

Constant time median filtering of extra large images using Hadoop

Mohammadreza Azodinia, Vahid Farrokhi, Andras Hajdu

University of Debrecen
m.r.azodinia@inf.unideb.hu
vahid.farrokhi@inf.unideb.hu
hajdu.andras@inf.unideb.hu

Abstract

The spatial resolution of remote sensing and medical images such as MRI, CT and PET are constantly increasing and analyzing these images in real time is a challenging task. But this limits the efficiency of many image processing algorithms. Among different efficient image processing algorithms, median filtering is a principal element in many image processing situations which manages to reduce the noise while preserving the edges. Median Filtering in Constant Time (MFCT) is a simple yet fastest median filtering algorithm which can handle N-dimensional data in fields like medical imaging and astronomy. With trend toward the median filtering of large images and proportionally large kernels, Hadoop MapReduce (a popular big data processing engine) can be applied and utilized. MapReduce provides the simplicity of defining the map and reduce functions while the framework takes care of parallelization and failover automatically.

Hence, in this paper we discuss on possibility of the incorporation of MFCT algorithm with Hadoop MapReduce framework to improve the performance of processing of extra large images.

Keywords: median filtering, MFCT, MapReduce, Hadoop, parallelization

MSC: 68-06, 68U10

1. Introduction

The daily growth of the amount and size of data generated by modern acquisition devices is in a challenge with the processing capabilities of single devices. Considering the fact that processor architectures are reaching their physical limitations, using distributed computing technologies would be the best and safest choice for solving those problems that do not fit into the capability of a single machine. Therefore, relating large-scale data processing, as a solution many approaches turned

towards distributed computing by combining many single computers and let them interact through different interfaces. An example is the MapReduce framework.

Generally, the term large data refers to two types of data. In the first case it refers to very large image e.g. a microscopic image which is roughly hundred thousand pixels into hundred thousand pixels. What makes them challenging is the fact that they do not fit into the memory. The second scenario happens when we are dealing with very large data sets of regular images which cannot be processed on a single personal computer fast enough. The need for an advance computing approach such as MapReduce emerges as the CPUs' architecture are facing their physical limitation caused by excessive growing size of data and limited capacity of a single machine.

In this paper we discuss about our motivations and new approach for filtering a large image using commodity computers for distributed processing. The challenge is to process a large image with a constant time median filter algorithm utilizing MapReduce framework and its open source implementation, Hadoop.

The rest of the paper is organized as follows. In section 2 we discuss briefly the background and section 3 presents our approach for filtering a large image using Hadoop in a constant time. Section 4 concludes this work.

2. Background

2.1. Median Filtering

The median filter is a fundamental image processing algorithm which provides a mechanism to remove image's salt and pepper noise while preserves the edges of the image. More advanced image filters like morphological operations, rank-order processing, and unsharp masking as well as many higher-level applications like speech and writing recognition, object segmentation, and medical imaging are different variations on the median filter [1]. However, the usefulness of the median filter has long been hampered by its high computational cost and its non-linearity.

The median filtering algorithms are distinguished one from each other with the sorting algorithm that each of them use. During median filter implementation all pixel values are sorted and the value which is placed at $(n^2/2)$-th position is selected as the median, where n is the filter size (i.e. the size of one side of the square filter window). If we use Quicksort, the computational complexity of filter is $O(n^2 \log n)$ operations per pixel. However, if Quickselect is used, instead of full sort, it results in computational complexity of $O(n^2)$. The necessity of recalculating everything from scratch for each window position is considered as the major drawback of this approach. In order to address this problem, search trees can be used accordingly where we reuse previous window position's sorting result. Therefore just n obsolete pixels are removed and n new pixels are added to it as shown in Figure 1 where self-balancing binary search trees' addition/subtraction is done in average $O(\log n)$ and selecting the median in constant $O(1)$ time and the overall average complexity would be $O(n \log n)$ per pixel.

In [2] Gil and Werman improved the aforementioned approach by using an Implicit Offline Balanced Binary Search Tree Based data structure and reduced the computational complexity to $O(\log^2 n)$. Moreover completely different approach popped up where Huang et al [3] achieved the computational complexity $O(n)$. They proposed counting of gray values in the kernel utilizing a 256-bin histogram. Median calculation is then done by successive summing of individual bins.

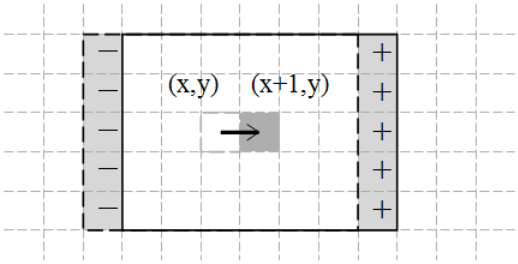

Figure 1: $2r + 1$ pixels must be added to and subtracted when moving from one pixel to the next. In this figure, $r = 2$. [4]

Weiss proposed an improvement [5] for Huang's algorithm by utilizing the distributive property of histograms in which he proposed processing several image rows simultaneously and on the other hand maintaining a set of partial histograms instead of using only one. Consequently for all rows which are processed at the same time, same set of partial histograms is used where the overall computation complexity becomes $O(\log n)$. Although the achieved computation complexity could be considered as close to constant time approach but still lacked of being independent of kernel radius size.

Finally in 2007 the first roughly $O(1)$ average case complexity was proposed by Perreault and Hébert [6] in which for each column of the horizontally moving kernel they used separate histograms. The column histograms are saved and updated recursively. The window histogram is created by summing of corresponding bins of column histograms. The subtraction of the out of scope column histogram of the running window and addition of the histogram for the newly added column will update the window histogram as shown in Figure 1. The proposed algorithm is bit-depth dependant and is independent of the kernel radius n. It achieves $O(1)$ average case complexity of images of 8-bit utilizing SIMD extensions of modern CPUs.

Alekseychuk in 2012 proposed another constant time median filtering algorithm [4] by combining the ideas of search tree with histogram based approaches of [5] and [6]. This algorithm utilizes a hierarchical data structure in order to store value occurrences in specific value intervals. Being capable of processing higher precision data (high bit-depth data) as well as the platforms that are lacking hardware SIMD extensions, motivated us to find the latter algorithm the fastest median filter which can be easily extended to N-dimensional images and supports usual higher-level parallelization of the algorithm. Based on the mentioned reasons, selecting this

algorithm in fact helps us during the processing of the large images in parallel environment.

2.2. MapReduce and Hadoop frameworks

Considering large image, it is always critical utilizing a model of distributed computing to solve image processing tasks. One might be asking what the alternatives are. On a single PC Batch processing is feasible for only small amounts of image, and since our image is very large, the fact is that only a part of this image fits into memory. On the other hand due to slow hard drive access the speed of computation will drastically decrease. Therefore an alternative is a must.

One approach is treating the problem like a traditional large-scale computing task e.g. cluster computers built on GPUs which requires high level parallel programming along with specialized hardware and complex parallel programming. It is interesting to know whether a computer cluster as a solution for many kinds of problems can be tackled with simpler and cheaper systems without much decrease in efficiency. This issue is inspired us to the utilization of both Google MapReduce and Apache Hadoop.

Tackling this issue, assigning the batch process to several computers at the same time is a solution, but of course at the cost of creating a need for job monitoring, mechanisms for data distribution. Besides there should exist a means to ensure that the processing completes even when some computers experience failures during the task execution. This is exactly the problem that both Google MapReduce and Apache Hadoop were designed to solve; scalability and fault tolerance.

MapReduce is a programming model developed by Google for processing and generating large datasets used in practice for many real-world tasks [7] as shown in Figure 2. In MapReduce concept the developer's task is to define only two functions, Map and Reduce functions, while everything else is handled by the implementation of the model. Extra parameters and functionalities of MapReduce are provided to fine-tune the system in order to help the model to conform to the assigned task better. The MapReduce computation consists of reading the input from disk and converts it to Key-Value pairs. Then map function processes each pair separately and outputs the results as any number of Key-Value pairs. For each distinct key, the Reduce function processes all Key-Value pairs with that a Key and returns any number of Key-Value pairs. Once all input pairs have been processed, the output of the Reduce function is then written to disk as Key-Value pairs.

Hadoop is an open-source framework for distributed computing, written in Java and developed by the Apache Foundation and inspired by Google's MapReduce [8]. On one hand being used by many companies such as Facebook and Yahoo in order to analyze large-scale data tasks and on the other hand being easily adapted for use with any kind of hardware ranging from a single computer to large data center, Hadoop is the best candidate for image processing on the MapReduce model.

There are many approaches where Hadoop framework shows its distinction rather than other distributed computing frameworks. Lv Zhenhua et al. [9] de-

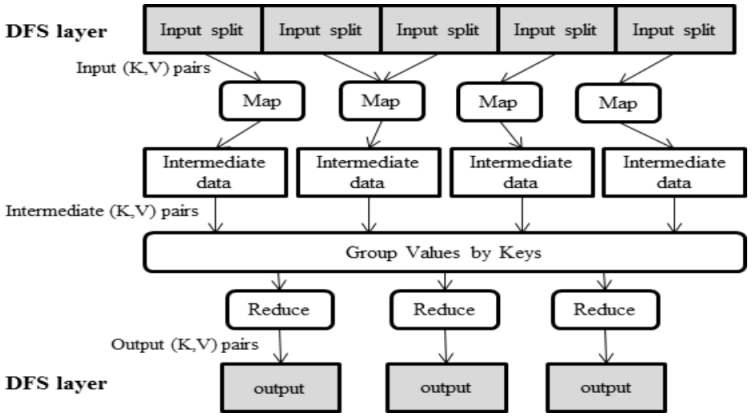

Figure 2: MapReduce architecture

scribed how to find out different elements of satellite images based on their color using MapReduce approach aiming at separating trees from buildings. They claimed that non-parallel approach could not process those images more than 1000×1000 pixels. So they aimed to process those images utilizing MapReduce. Kocakulak and Temizel [10] compared a large dataset of images against an unknown image by using Hadoop and MapReduce. They proposed a correlation method to compare the library of images with the unknown image. Li et al. [11] proposed a parallel clustering algorithm with Hadoop and MapReduce in order to compensate the high running time taken in the execution of clustering algorithms of a large number of satellite images. They achieved the results by clustering each pixel with its nearest cluster and in then based on every pixel in one cluster set they calculated all the new cluster centers. Golpayegani and Halem [12] implemented a gridding of Aqua satellite images collected by AIRS instruments by utilizing a Hadoop MapReduce framework parallel approach to process these images. Mohamed H. Almeer [13] tested eight different image processing algorithms on top of the Hadoop environment of 112-core high-performance cloud computing system. The results show that using a Hadoop MapReduce approach provides us with a scalable and efficient possibility in processing multiple large images used for large-scale image processing applications. He also justifies his approach by showing the difference between the single PC runtime and the Hadoop runtime.

All aforementioned approaches motivated us to use MapReduce and Hadoop in which the parallel algorithm conducted by MapReduce yields superiority rather than an implementation on a single machine. However, it is noticeable that by utilizing higher performance hardware the superiority of the MapReduce algorithm would be reflected better. So using MapReduce approach as a parallel environment and higher performance hardware is strongly suggested in large scale image processing tasks.

3. Proposed approach

In this section we propose a novel approach which enables us to process large images using median filter with Hadoop utilization. One challenge is to choosing an appropriate median filtering algorithm with the aim towards achieving the fastest one. Another challenge is to adapt the images which are expected to be processes using median filter on top of Hadoop. Finally adapting the chosen median filter along with the input data to be run in parallel on Hadoop is the novel proposed method.

There are two different nodes in Hadoop MapReduce phase to be responsible for Map and Reduce functions' execution: JobNode and TaskNode. The JobNode is the manager in Hadoop MapReduce and takes care of task designation to the computing nodes. Task node is responsible for task execution process. While the task execution is processing JobNode keeps updating the JobNode about the operation status so in case the TaskNode is damaged, the JobNode will be informed and will reassign the task to another idle TaskNode to process the task.

Hadoop cluster owns a distributed file system which is called Hadoop Distributed File System (HDFS). It is inspired by the Google File System and it provides a fault-tolerant storage structure capable of holding large amounts of data and allow for fast information retrieval [14]. In the same scenario, the HDFS phase consists of NameNode and DataNode. Name node plays the manager role and is responsible for managing and keeping track of the data in the system. It does not save the data itself but notifies the MapReduce of the storage location of the data. The responsibility of the DataNode is to store the data and keeps the NameNode updated of the data's location.

In our Hadoop execution approach, as shown in Figure 3, the InputFormat interface [8] is responsible to define how and where from the map function reads the input data. When user uploads the large image, InputFormat interface defines InputSplits which aims at splitting the input into smaller chunks (the size of each chunk varies between 16 MB to 64 MB and depends on the programming design) where HDFS distributes and stores them on the computing nodes.

In order to manage these smaller input chunks, Hadoop benefits from RecordReader class in order to provide data access mechanisms for map function. It reads the input chunk from the HDFS and converts it into key-value pairs. RecordReader is iterated over all input splits as far as all the input splits have been converted and prepared to be consumed by map function [15].

Next step is to perform the constant time median filter's MapReduce job in parallel on the split chunks. Each chunk of input will be assigned to each map function where the constant time median filter task will be applied to each input chunk. According to the discussed and selected median algorithm [4], each node creates interval-occurrence tree (IOT) of the image chunk and the median of each pixel is selected irrespective to the size of the sliding window in constant time. Then the filtered chunk, intermediate results, is stored in the local storage of HDFS for the further access and consequently the TaskTracker which is residing at the

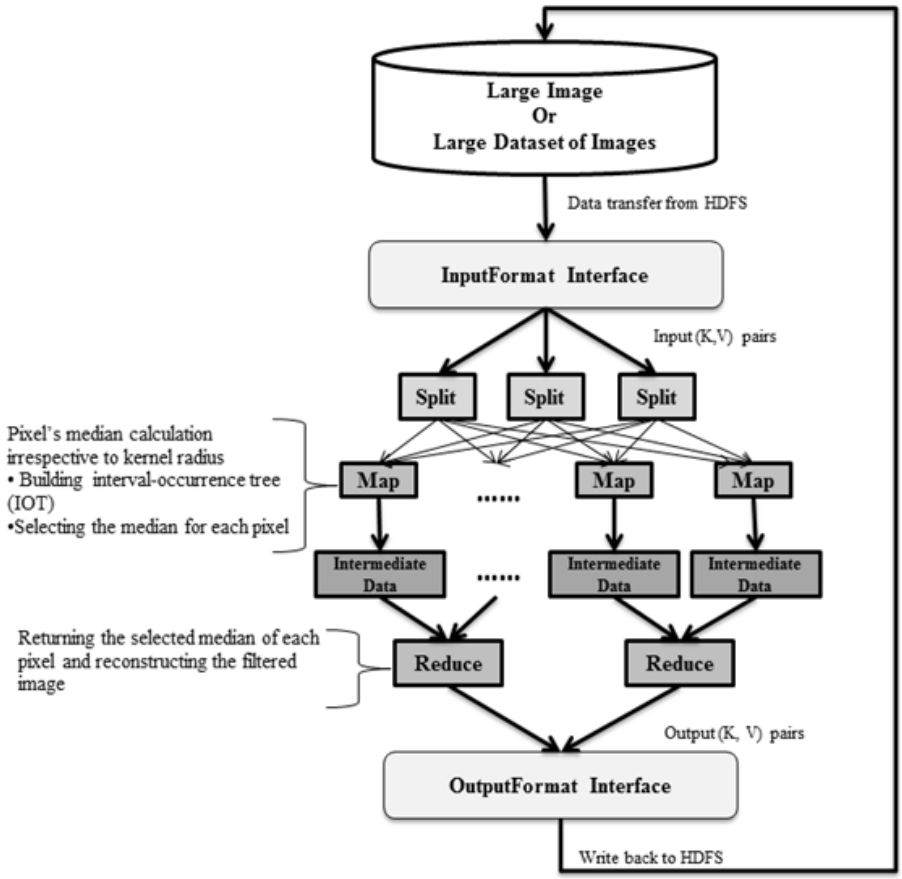

Figure 3: Constant time median filtering of large images

DataNode updates the JobTracker about the completion of the job.

The reducer function of the constant time median filter fetches the output key-value pairs which are having the same key and merges then to construct the final output (filtered image). In order to group the output key-value pairs with the same key, intermediate results must be shuffled over the network. In order to minimize the time consuming shuffling process Hadoop users benefit from an optional optimization option this is called Combiner. As soon as the Reduce phase is finished its output, which are the filtered chunks, would be written back to HDFS which enables the user to retrieve the resulting filtered image. At the final step, all processed chunks will be juxtaposed and construct the large-filtered image.

4. Conclusion

The aim behind using MapReduce (Hadoop) is to process large amount of data and provide to the user with the results in the minimum time. Many researchers have used Hadoop framework for the applications which requires lower computing complexity and higher speed. In this paper, we proposed a new approach by using Hadoop implementation, where Map function is in charge of constant time median filtering execution and Reduce function is assigned to combine the intermediate data which is processed at the Map phase and creating the final filtered large image. The MapReduce model implemented by Hadoop is a logical option to be utilized to solve these processing issues as it is freely available and provides a reliable platform for parallelizing computation and does not have any requirements with regard to specialized hardware or software. Although Hadoop is able to read wide variety of input formats like text, binary, database and etc through FileInputformat class, it still limits in some cases where there is no way to directly operate some image formats on Hadoop. For handling such cases, one option is to change the form of the image e.g. changing the form of tif format images into text. The proposed approach can play an important role in denoising large remote sensing and medical images such as satellite images and high resolution microscopic images as well as any situation in which denoising a very large image is of a need. Our future aim is to achieve more image processing tasks of large images in the cloud infrastructure utilizing Hadoop framework.

References

[1] P. Maragos and R. W. Schafer, "Morphological Filters - Part 2: Their Relations to Median , Order-Statistic , and Stack Filters," IEEE Trans. onAcoustics Speech Signal Process., vol. 35, no. 8, pp. 1170 – 1184, 1987.

[2] J. Gil and M. Werman, "Computing 2-Dimensional Min , Median and Max Filters," IEEE Trans. image Process. a Publ. IEEE Signal Process. Soc. Anal. Mach. Intell., vol. 15, no. 5, pp. 504 – 507, 1996.

[3] T. Huang, G. Yang, and G. Tang, "A Fast Two-Dimensional Median Filtering Algorithm," IEEE Trans. onAcoustics, Speech Signal Process. Speech Signal Process., vol. 27, no. 1, pp. 13 – 18, 1979.

[4] A. Alekseychuk, "Hierarchical Recursive Running Median," in IEEE International Conference on Image Processing (ICIP), 2012 19th, 2012, pp. 109 – 112.

[5] B. Weiss, "Fast median and bilateral filtering," in ACM SIGGRAPH 2006 Papers (SIGGRAPH '06), 2006, vol. 1, no. 212, pp. 519–526.

[6] S. Perreault and P. Hébert, "Median filtering in constant time," IEEE Trans. Image Process., vol. 16, no. 9, pp. 2389–2394, Sep. 2007.

[7] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Commun. ACM, vol. 51, no. 1, pp. 107–113, 2008.

[8] T. White, Hadoop: The definitive guide. O'Reilly Media, Inc., 2012.

[9] Z. Lv, Y. Hu, H. Zhong, and J. Wu, "Parallel K-Means Clustering of Remote Sensing Images Based on MapReduce," in Proceeding WISM'10 Proceedings of the 2010 international conference on Web information systems and mining, 2010, pp. 162–170.

[10] H. Kocakulak and T. T. Temizel, "A Hadoop solution for ballistic image analysis and recognition.pdf," in High Performance Computing and Simulation (HPCS), 2011 International Conference on, 2011, pp. 836 – 842.

[11] B. Li, H. Zhao, and Z. Lv, "Parallel ISODATA Clustering of Remote Sensing Images Based on MapReduce," in Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2010 International Conference on, 2010, pp. 380 – 383.

[12] N. Golpayegani and M. Halem, "Cloud Computing for Satellite Data Processing on High End Compute Clusters," in Cloud Computing, 2009. CLOUD '09. IEEE International Conference on, 2009, pp. 88 – 92.

[13] M. H. Almeer, "Cloud Hadoop Map Reduce For Remote Sensing Image Analysis," J. Emerg. Trends Comput. Inf. Sci., vol. 3, no. 4, pp. 637–644, 2012.

[14] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," ACM SIGOPS Oper. Syst. Rev., vol. 37, no. 5, pp. 29–43, Dec. 2003.

[15] J. Venner, Pro Hadoop (Expert's Voice in Open Source). Apress, 2009.