International Journal of Engineering Applied Sciences and Technology, 2024
Vol. 9, Issue 05, ISSN No. 2455-2143, Pages 11-20
Published Online September 2024 in IJEAST (http://www.ijeast.com)

TOOLS AND TECHNIQUES FOR PARALLEL IMAGE PROCESSING

Mateeh Ullah, Sameer Naveed, Abdullah Asim
Department of Computer Science
FAST-NUCES, Lahore, Punjab, Pakistan

Abstract— **The main aim is to improvement in the quality of image, perform other operation, extraction of information and to classify the image while doing image processing. It is effectively used in computer, medical and other related fields. The main problem is that it is generally a time-consuming process; Parallel computing (parallelism) provides an efficient and convenient way to address this issue. There are many challenges in Image processing like Filtering, Restoration and classification etc. In addition, it is also a time-consuming process. The solution of these challenges is to use a parallel computing technique known as parallel image processing. The main focus of this paper is to review and to provide the comparative study of the existing contributions of tools and techniques of parallel image processing and analysis between different technique which are MATLAB, CUDA, BIONIC, Hadoop and GPU (graphic Processing Unit) along with limitation and advantage of these tools and techniques. In this review, we also tried to discuss the architecture of these parallel image processing techniques.**

Keywords— **time-consuming, parallel image processing, MATLAB, CUDA, BIONIC, Hadoop, GPU**

I. INTRODUCTION

Image combination of small pixels. It may be thought of as a function of two real variables, such as g(x, y), where the amplitude (e.g. brightness) of the picture at the x- and y-axes, i.e.(x,y), is indicated by g. Image processing is used to extract useful and important information from an image by applying operations on it. Image processing plays important roles in many fields like meteorology, medical imaging, computer vision, astronomy, remote sensing, machine/robot vision and other related fields. There are many issues in image processing like Filtering, Restoration, Registration, Fusion, Segmentation and Classification along this it is also a time-consuming process for example gray scale image having size of 1024 pixels on both x-axis and y-axis requires more than 1 million operations for processing of image, now if we have image which is color one then it is product of number of channels and in the processing of images with high resolution.

Now-a-days parallel processing is a very useful technique to achieve maximum utilization of CPU. By using the technique of parallel processing, we can do executions of multiple processing concurrently. We can break tasks/problems into multiple parts and execute them concurrently. Having multiple processes in a computer or having a multi-core processor in a computer is the main requirement of parallel processing.

The solution of time-consuming image processing can be achieved by using parallel techniques which is known as parallel image processing. So, parallel image processing is the distributed execution of an image processing algorithmic program on multiple processors. We can use these techniques in MATLAB, CUDA, Hadoop or in BIONIC (use in mobile devices for parallel image processing). We can also use parallel image processing techniques in GPUs (Graphic Processing Units).

II. LITERATURE REVIEW

In this paper, we are discussing the Tools and techniques of parallel image processing. Previously, a lot of research papers and journal articles have been published, explaining everything about parallel image processing. Some of these articles will be reviewed here.

An article written on the topic "parallel computing in digital image processing". According to researchers, if the application is based on a sequential algorithm then it is difficult to enhance their performance. Image processing needs a high degree of implementation of algorithm in parallel environment. So the main focus of parallel digital image processing is to get better utilization of resources and high performance. In this paper they also discussed the operator which is used in image processing which are point, global and neighborhood operator [1].

Another article "Digital image processing using parallel computing based on CUDA technology". According to researchers, CUDA is widely used in different areas i.e. image noise and noise removal algorithm. It also provides the performance comparison using GPU and without using Graphic Processing Unit along with using different percentage of CPU and Graphic Processing Unit [11] [8].

Table -1 Digital image processing using parallel computing based on CUDA technology

Time(sec)		
Proportion	**160x128 pixels**	**210x182 pixels**
1 CPU thread	184.091343	627.95773
½ CPU + ½ GPU	98.867030	336.006175
1/8 CPU+7/8 GPU	39.236646	139.992048
GPU	29.888206	111.724455

The above table illustrates that how time varies by using different quantity of threads of Graphic processing unit and CPU.

A paper published in 2018 discussed the "Parallel Processing of Images in Mobile Devices using BOINC". In this paper, researchers completely discussed the procedure of parallel processing of images using BOINC. They also discuss the challenges and their solutions which they face while using BOINC. The challenges are modification of code to add function-calls to the BOINC API in mobile devices program, and the division and merging of the image among the mobile devices. Along this they also discuss the life cycle of the BOINC work unit [10] [15].

The details about tools and techniques of parallel image processing can be found anywhere i.e. in papers, thesis or websites. However, the combined analysis of these tools and techniques are not available anywhere. Therefore, we will be discussing the tools and techniques of parallel image processing by giving references to different research papers and thesis in detail.

III. EXPERIMENT AND RESULT

The tools and technique for parallel image processing are following:

- Parallel image processing using MATLAB
- Parallel image processing using CUDA
- Parallel image processing using Hadoop
- Parallel image processing using BOINC

A. Parallel image processing using MATLAB

The main reason of using MATLAB is that it provides user friendly interface and so popular. It is used for parallel processing of images. Parallel processing in MATLAB can be done by using following technologies:

- pMATLAB with bcMPI
- PCT which is known as "Parallel Computing Toolbox" with Computing Server which is distributed of MATLAB and Star-P.

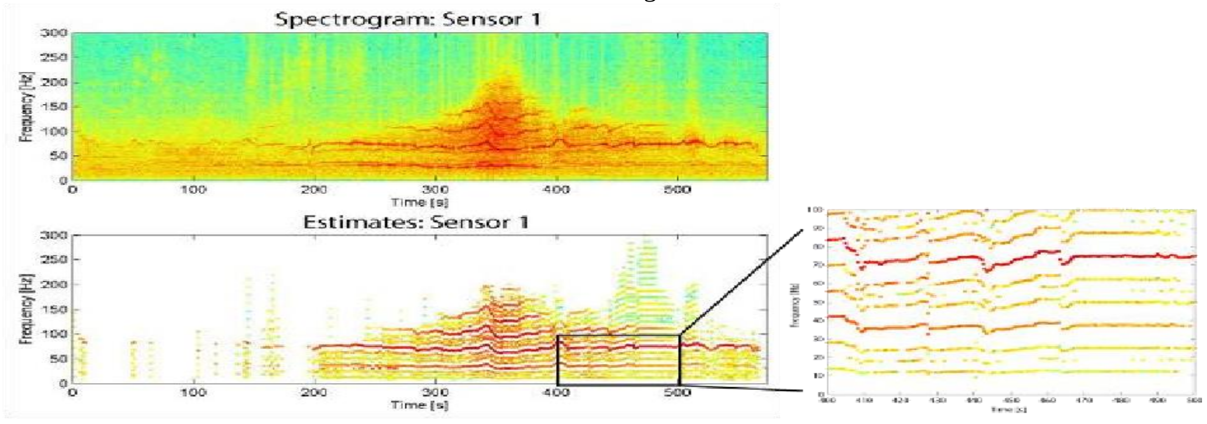
1) Problem:

Processing of Acoustic signal on a field of battle is use to detect and classify vehicles which are present on ground. By using active sensors in array, passing input symbol can be gathered for observation of target on fields, tracking of objects, and to identify the object etc. Environmental factors like as topography, network of sensor, wind speed, and so on can have an impact on self-localization.

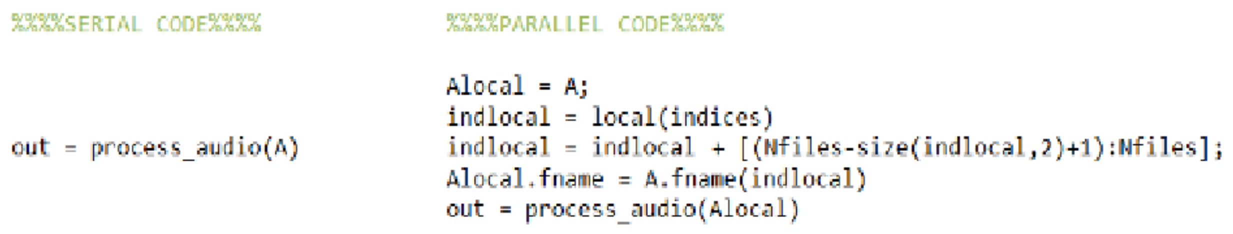
GRAPE is a tool in United State Research lab that consists of a GUI for performing acoustic signal operations [13].

Data having size of terabyte captured in 3-minute and on the other hand each file take 1 or more than 1 minute to process. For the calculation of the time of arrival of the acoustic impulses, a variety of techniques are utilized. We use parallel MATLAB for processing of data for the attainment of real-time. Because each data file may be handled individually, the strategy which use while doing parallelism is to distribute induvial files over many processors [6] [15].

This application was determined to be parallelizable utilizing task parallel (Embarrassingly Parallel) approaches. The process audio() take too much time to execute therefore most of the time was spent to calculate this function, according to the MATLAB profiler. It was also discovered that the information produced by this functionality was not used somewhere else. As a result, task paralleling is employed. A data structure is sent to the method process audio(). The nFiles field in this structure defines How many # of files processed by the process audio() method. These indices were dispersed across numerous processors using a distributed array. Vehicle signature identified is shown below:



(a)



```
%%%%SERIAL CODE%%%%                 %%%%PARALLEL CODE%%%%

                                     Alocal = A;
                                     indlocal = local(indices)
out = process_audio(A)               indlocal = indlocal + [(Nfiles-size(indlocal,2)+1):Nfiles];
                                     Alocal.fname = A.fname(indlocal)
                                     out = process_audio(Alocal)
```

(b)

Fig. 1. (a) Vehicle Signature (b) code before and after parallelization.

B. Parallel Image Processing using CUDA

Point and neighborhood processing techniques of image processing usually become costly operation for those images which are large in size. It also becomes pricier having color in images. Now-a-days graphic processing unit provide flexibility to process large size of data. NVIDIA created the CUDA architecture, which unites GPU cores into a vector that can be configured to minimize time of processing across huge volumes of data [11].

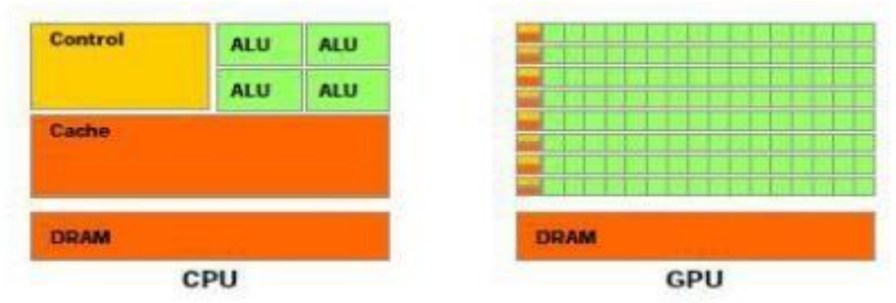


Fig. 2. Difference between CPU and GPU

1) **Architecture of CUDA:**

The GPU's design is responsible for CUDA's significant speed improvement. It has hundreds of ALUs. As a result, the emphasis is on computation. CUDA allows the newest NVIDIA GPUs to be used for computing in the same way that CPUs are. Unlike CPUs, GPUs feature a parallel flow architecture that prefers the execution of multiple concurrent threads slowly above the execution of a single thread quickly. GPGPU refers to this method of tackling general-purpose issues using GPUs (General Purpose Graphics Processing Unit). Following image shows the CUDA workflow diagram [8].

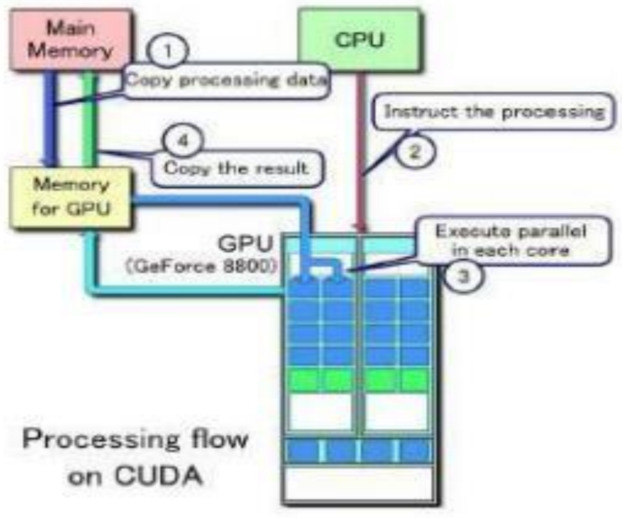


Fig. 3. Workflow of CUDA

To start, the compiled code for CUDA operates in the same manner as some other app. Its core execution occurs place in the Processor. The app continues to use the CPU for non-kernel tasks after a kernel call is sent. The kernel function is being run on the GPU at the same time. As a consequence, parallel processing between the CPU and GPU happens. This is termed as heterogeneous programming. The fundamental bottleneck in application execution is memory movement between the host and the device. Both are paused until this procedure is completed [14] [20].

2) Functions of CUDA

This technique was tested on following image processing functions.

- RGB to Grayscale
- Negative Filter
- Darkening Filter
- Brightening Filter
- Low Pass Filter (Filter having low pass)
- High Pass Filter (Filter having high pass)
- Sobel Filter for Edge Detection
- Recursive Ray Tracing Algorithm

3) Analysis of parallel image processing functions

Applying parallel processing we get following results for each function:

a) RGB to Grayscale

Table -2 Experiment Result of RGB to Grayscale Image processing function

No of blocks	No of Threads	Time(ms)
27000000	1	11.08
54000	500	21.61
27000	1000	42.86

b) Negative Filter

Table -3 Experiment Result of Negative Filter Image processing function

No of blocks	No of Thread	Time(ms)
27000000	1	21.03
54000	500	29.02
27000	1000	44.32

c) Brightening Filter

Table -4 Experiment Result of Brightening Filter Image processing function

No of blocks	No of Thread	Time(ms)
27000000	1	27.36
54000	500	46.14
27000	1000	99.39

d) Darkening Filter

Table -5 Experiment Result of Darkening Filter Image processing function

No of blocks	No of Thread	Time(ms)
27000000	1	36.71
54000	500	53.78
27000	1000	56.39

e) High and Low Pass Filter

- **High Pass Filter**

By development of filer having high pass on the CPU for image which is greyscale having a resolution of 3,000×3,000, we discovered that the execution time was 468 milliseconds. It takes 39.3 milliseconds for execution when it is implemented on GPU. As a result, GPU is performing 12x better.

- **Low Pass Filter:**

By development of filer having high pass on the CPU for image which is greyscale having a resolution of 3,000 X 3,000, we discovered that the execution time was 483 milliseconds. It takes 38.1 milliseconds for execution when it is implemented on GPU. As a result, GPU is performing 12x better.

f) Sobel Filter for Edge Detection

For edge detection, the Sobel mask was utilized both vertically and horizontally. For this purpose, a 3,000 X 3,000 grayscale picture is evaluated. We discovered that the execution time was 297 milliseconds. The implementation on GPU is same and took 28.4 milliseconds to complete. As a result, GPU provides roughly 11 times better performance.

g) Recursive Ray Tracing Algorithm

For 300 passes, a recursive ray tracing technique was constructed. With each pass, the image grows clearer and clearer. A box contains two spheres and a light source. One sphere is translucent, producing light effects such as refraction, absorption, scattering, and so on. Another spherical has a reflective surface. Analysis of recursive ray tracing algorithm on GPU is given below [8]:

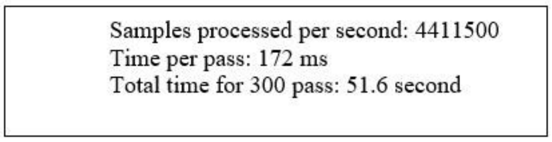

```
Samples processed per second: 4411500
Time per pass: 172 ms
Total time for 300 pass: 51.6 second
```

Fig. 4. Analysis of recursive ray tracing algorithm on GPU

C. Parallel Image Processing using Hadoop

Hadoop based on HDFS. HDFS is abbreviation of "Hadoop Distributed File System". To store files throughout the cluster, Hadoop employs the HDFS. HDFS is a free and open-source project that manages large-scale data collection and parallelism [2].

Small file storage is one of Hadoop's major issues. A tiny file is one that is substantially smaller than the HDFS block size. Large picture datasets are made up of a large number of little image files, which HDFS struggles to handle. This issue may be overcome by providing a container to organize the files in some way. Hadoop provides a few options: HAR File, Sequence File, and Map File [7].
The HDFS parallel processing architecture is built on the MapReduce concept, which was introduced by GFS about 2004. GFS is an abbreviation for "Google File System". MapReduce is a method for dealing with highly distributable problems across big data sets by utilizing a large number of clustered devices (computers). Architecture: Optimization of Hadoop for large image processing

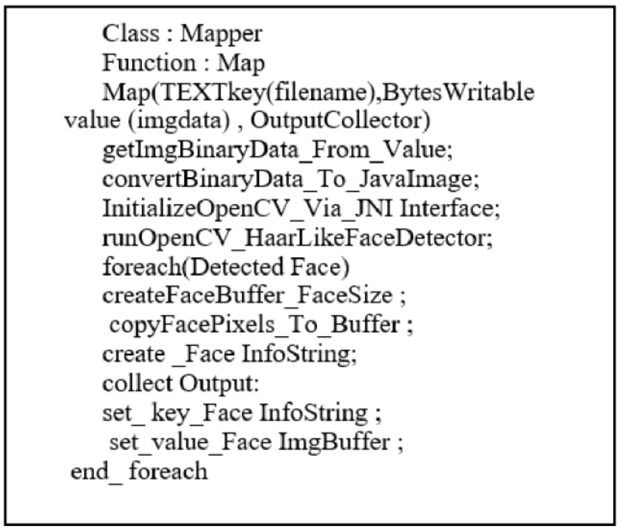
1) Distributed Computing with Hadoop

HDFS is a big and efficient distributed file system made up of several computer nodes. The master node manages NameNode, whereas the worker nodes manage DataNode. DataNodes manage local data storage and offer status updates. In HDFS, there is just one NameNode, but hundreds of DataNodes [2].
In Hadoop, worker nodes function as both file system local storage units and parallel processing nodes. Hadoop uses the MapReduce programming model to do tasks in parallel. This model is divided into two stages: (i)Map and (ii)Reduce, with inputs and store output in pair. Hadoop job execution characteristics is use to develop jobs by user along this implementation of Map and Reducer function help in development of jobs. Jobs are provided and subsequently executed as MapTask or ReduceTask on worker nodes. JobTracker is Hadoop's principal activity management and scheduling tool. JobTracker provides jobs to worker nodes as Mapper or Reducer tasks by initializing TaskTrackers in worker nodes.
TaskTracker executes the Map function or Reduce function job and give information back to JobTracker on its progress. Hadoop provide InputSplits by splitting up input files, with each job processing one InputSplit. If InputSplit has greater size than HDFS block size then InputSplits can be save in 1 or more than 1 block. Due to this reason InputSplits size is carefully assigned. As a result, far blocks must be sent across the network to the MapTask node in order to construct InputSplit. Hadoop's map function generates results that are then delivered to the reduce function. As a result, the map function's output format is the same as the reduce function's input format. Hadoop's FileInputFormat class is the root of all Hadoop-related file input formats. This class contains information about InputSplit. InputSplit is used indirectly by the Mapper class's map method. InputSplits are initially transformed into pairs of input records [19].
The data to be evaluated in distributed systems is frequently not present at the node or device that execute that information, which decreases parallel processing performance. Processing data on the same node where it is stored is 1 of the rules driving the evolution of HDFS. This is known as data locality, and it improves Hadoop's parallel data processing performance [2].

2) Interface Design

The map function must take the entire picture contents as a single input record in order to apply a classifier to each image. Classifier is treated as “face detection algorithm”. HDFS generates splits from input files according on the split-size option. These InputSplits are passed to the MapTasks. When splitting files, if the file size exceeds the split-size, the file is split into numerous splits. A collection of files can also be merged into a single InputSplit if the total size of the input files is less than the split size.
The ImageFileInputFormat class in Hadoop is inherited from the FileInputFormat class. ImageFileInputFormat converts each picture file into a FileSplit. Because each picture file is not divided, binary image content is not destroyed. Furthermore, the ImageFileRecordReader class extends Hadoop's RecordReader class in order to create image records from FileSplits for map functions. As a consequence, photo pixel data may be recovered easily from Hadoop input that has been segmented into image processing tasks.

Following that, we may apply any picture processing to the picture material [2] [12].
The map method of the Mapper class is used in the following example. In the following example, the map function of the Mapper class is used to apply the face detection algorithm to image data. The Haar Feature-based Cascade Classifier for Object Detection approach from the OpenCV package is used for face detection. The Java Native Interface is used to incorporate OpenCV into interfaces (JNI). The implementation of the map function is shown below. The variable "FaceInfoString" stores information about detection attributes such as image name and detection positions [12].



```
    Class : Mapper
    Function : Map
    Map(TEXTkey(filename),BytesWritable
value (imgdata) , OutputCollector)
    getImgBinaryData_From_Value;
    convertBinaryData_To_JavaImage;
    InitializeOpenCV_Via_JNI Interface;
    runOpenCV_HaarLikeFaceDetector;
    foreach(Detected Face)
    createFaceBuffer_FaceSize ;
     copyFacePixels_To_Buffer ;
    create _Face InfoString;
    collect Output:
    set_ key_Face InfoString ;
     set_value_Face ImgBuffer ;
 end_ foreach
```

Fig. 5. Hadoop Optimization for Massive Image Processing

Hadoop generates output file names as strings containing job ID numbers. Processing of image interface generate files for output containing the detected face photos after detection of face. The output file names should include the recognized picture name and detected coordinate information to make it easier to recognize these photographs. The ImageFileOutputFormat class was designed to save output files as images with the appropriate name.

Before using the process approach, consolidate small-size files into a single large-size file to reduce the number of jobs. SequenceFile is a Hadoop file type for combining many tiny files. SequenceFile is the most often utilized solution in HDFS for tiny file difficulties. A huge number of little files are compressed into a single big file, which comprises small files as formatted indexed components. The file entry information is represented by the key, while the file contents is represented by the value. This is accomplished by creating a conversion process that accepts small-files as input and outputs SequenceFile. Although using SequenceFile increases overall speed, after merging, the image formats of the input images are lost. Each time a new input photo collection is introduced, preprocessing is necessary [9].

To optimize small-size image processing in HDFS, a series of images is also incorporated as one InputSplit method. Hadoop CombineFileInputFormat may combine several files and generate InputSplits from them. Furthermore, like InputSplit, CombineFileInputFormat selects files to be merged from the same node. As a result, the quantity of data exchanged between nodes lowers while overall performance improves.

Combining Files in Hadoop for large amount of Image Processing: Face Recognition Case Study 669th putFormat is an app system that does not interact with image files directly. To produce CombineFileSplit as a series of photos, we created CombineImageInputFormat, which is derived from CombineFileInputFormat. To construct records from CombineFileSplit, the MultiImageRecordReader class was created. Each picture treated as a separate record to map function by the help of ImageFileRecordReader. Output is generated by the help of ImageFileOutputFormat function, which are then saved to HDFS [2] [7].

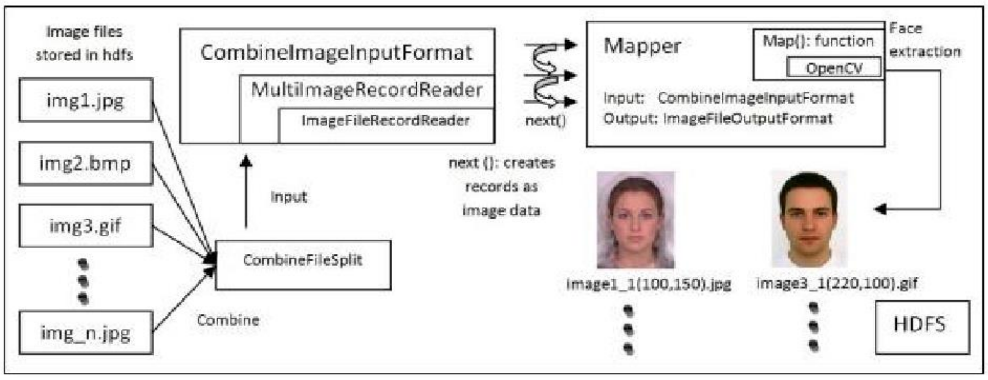


Fig. 6. Combine and process technique

D. Parallel Image Processing using BOINC

BOINC is the abbreviation of "Berkeley Open Infrastructure for Network Computing". BOINC software platform is used to develop volunteer computing or grid computing. BOINC is intended to assist applications with high computational, storage, or both requirements. The basic requirement of the application is that it be divided into a large number of jobs that may be done separately (thousands or millions). This job group also covers data collecting via evaluating camera images [18].

***1)* Cross Compilation**

Because contemporary Android OS phones use ARM CPUs but lack a pre-installed native compiler, a compiled wrapper for Android Platform is necessary. As a result, C++ compiler is used for tools and application code for ARM. Hence compiled wrapper is used from (https://github.com/BOINC/boinc/pull/1671) [10].

***2)* ITK Modules**

ITK is used as library that is for registration and segmentation of picture. This library, along with others like as VTK and IGSTK, is the most widely used open-source library for interpretation and processing in medical and other related fields. They have been rigorously tested to ensure their longevity, adaptability, and elongation. All of these characteristics combine to make them standard bearers. The ITK library is developed in C++ and is cross-platform compatible, with the installation procedure handled by the CMake build environment. The library includes a variety of picture segmentation, registration, and filtering algorithms. For parallel implementation it is necessary to:

- Divide the image and allocate pieces to each work unit.
- Collect the outputs to generate the final result [10].

***3)* Distribution and collection of Data**
Workunits are generated by the Work Generator by associating programmes and parameters. The command-line utility for submitting jobs is called create_work.
create_work [arg] infile_1 ... infile_n

***4)* Work divisor**
Divides the image and saves the portions (N) in separate files (N files).

***5)* Adapter**
This script employs create_work to generate N work units and their associated files. Keep track of the job identifiers.

***6)* Work generator**
It creates the work.

***7)* Assimilator**
BOINC deletes all files uploaded by the client when a work is assessed and assimilated by default. As a result, the findings must be processed or saved quickly.
The Assimilator's goal is to integrate the collected data by operating over the separate outcomes provided by each completed work unit.

***8)* Experimental Design**
a) 2k Factorial Design
The purpose of this experimental design is to explore the influence of k variables on a response variable, each of which has two options or levels.

***b)* Base Test**
There is no need of BOINC while running Smoothing Recursive Gaussian ImageFilter. And with that it run on single computer. The whole running duration was 49 minutes.

***c)* Factors**
In this 2k design, four important project elements were chosen, which are processing devices, degree of parallelism, redundancy and location of server [10].

Result of 2k experiment is shown below:

Table -6 2k Experiment Result

		Desktop Computer		**Mobile Devices**	
		450 work units	**1000 work units**	**450 work units**	**1000 work units**
Local	Redundancy 1	15 min	37 min	19 min	42 min
	Redundancy 2	38 min	1 hr 22 min	40 min	1 hr 26 min
Remote	Redundancy 3	48 min	59 min	54 min	1 hr 3 min
	Redundancy 4	1 hr 37 min	1 hr 54 min	1 hr 52 min	2 hr 1 min

IV. ANALYSIS AND RESULTS

In these sections we will discuss the conclusion of experiment done in all technique.

A. Result and conclusion of MATLAB

The results of GRAPE code with parallelism technologies of MATLAB are shown below. The primary (left) vertical axis in the graphs below represents the total time required by the process audio() function to complete analysis on 63 data files [6]. The speedup realized while operating on many processors is shown on the secondary (right) axis. It is also worth noting that the changes necessary to parallelize the code resulted in a little than 1 percent increase in SLOC [17].
The MDCS and bcMPI tests were completed on the Ohio Supercomputer Center's Pentium 4 cluster using an InfiniBand communication network. The Star-P experiments were carried out on the Ohio Supercomputer Center's IBM 1350 Cluster, which has nodes with dual 2.2 GHz Opteron CPUs, 4 GB RAM, and an InfiniBand communication network. Each parallel MATLAB tool is accelerated approximately linearly because the parallelization technique is a task parallel solution with no inter-process communication. It is also clear that parallel MATLAB may significantly aid in giving the consumer with a quick response [3].

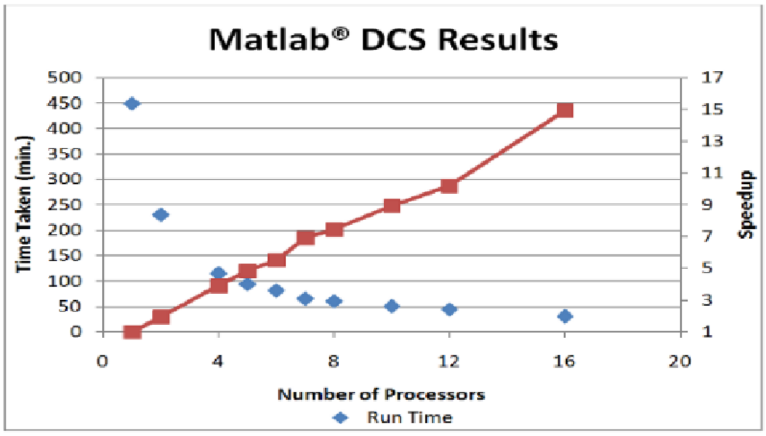


Fig. 7. Matlab DCS Results

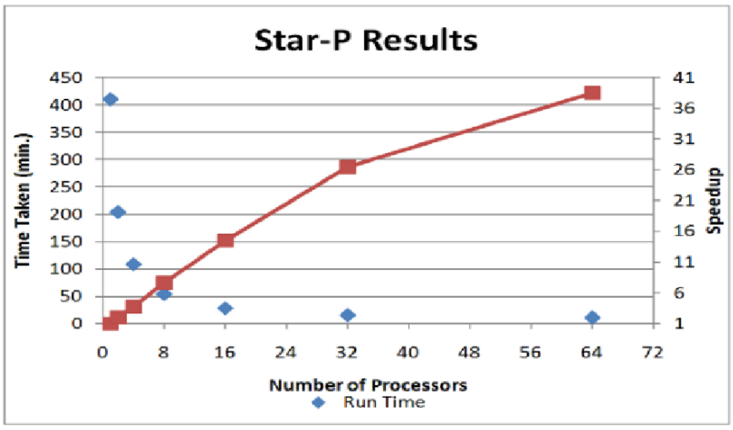


Fig. 8. Star-P Results

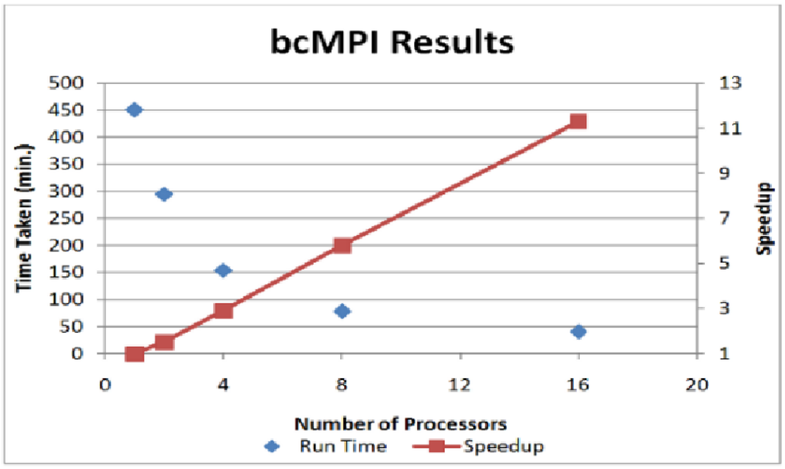


Fig. 9. bcMPI Results

B. Result and conclusion of CUDA

We've discovered that some programmes employ a lot of inter-thread communication, therefore increasing the number of threads per block enables us to reach outcomes faster. Inter-thread contact is not required in parallel programmes.

It was also observed during the development of an image filter that the composite of the # of blocks and threads should be equal to the # of items to be processed.

The algorithm which is use for tracing is Recursive ray. It has too much time complexity i.e. it requires too much computations. So, by processing it on GPU, we gain additional benefit from parallelism [16].

C. Result and conclusion of Hadoop

We built up a 6-node HDFS cluster to test the system and assess the outcomes. Face detection jobs on the cluster are executed on a predefined collection of picture files. A HDFS cluster of six nodes is built up to conduct face detection tasks on picture sets. Framework of Hadoop is deployed on a computer having virtual environment on each node. Because the default size is inadequate, the highest "Java Virtual Machine" size for Hadoop processes has been increased to six hundred megabytes [7] [5].

5 unique little images are in file which is used as input. In the input folders, the photo distribution based on file size is preserved. The HDFS face detection task evaluated the photographs in the source directories using the three methodologies. These are (i) the brute-force strategy with one task per image (just for comparison), (ii) the SequenceFile processing method, and (iii) the combine and process images method. Following Figure shows the results [2]:

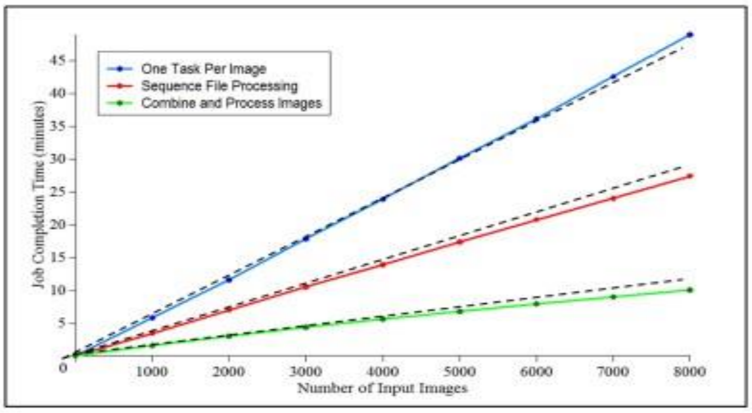


Fig. 10. HDFS face detection task evaluation result

D. Result and Conclusion of BOINC

Execution and processing time of image processing is very close and similar to desktop computer using mobile devices. There is no redundancy in BOINC and along this BOINC system is not intrusive. So due to above results and conclusion we recommend to place BOINC in local server [10].

V. CONCLUSION

Parallel computing has become norm in the field of computing. Now-a-days people are using parallel computing to achieve their goals because by use of parallel computing we can reduce the computations and divide task into sub task. So due to this reason in the field of image processing people are adapting parallel computing to deal with problem of high computations. In this paper, we reviewed different articles on parallel image processing then discuss the techniques of parallel image processing. After that, we analyze the experimentation of these techniques. We conclude that these techniques must be further worked on and implemented in the parallel image processing.

VI. REFERENCE

[1] Saxena, S., Sharma, S., and Sharma, N. (2016). Parallel Image Processing Techniques, Benefits and Limitations. Research Journal of Applied Sciences, Engineering and Technology, 12, 223-238. doi: 10.19026/rjaset.12.2324.

[2] Akhtar, M. N., Mohamad-Saleh, J., and Grelck, C. (2018). Parallel Processing of Image Segmentation Data Using Hadoop. International Journal of Integrated Engineering, 10. doi: 10.30880/ijie.2018.10.01.012.

[3] Anbarjafari, G. (2000). Digital Image Processing. [Online]. Available: https://sisu.ut.ee/imageprocessing/book/1.

[4] Fisher, R., Perkins, S., Walker, A., and Wolfart, E. (2000). Image Processing Learning Resources. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/HIPR2/hipr_top.htm.

[5] Skirnevskiy, I. P., Pustovit, A., and Abdrashitova, M. (2017). Digital Image Processing Using Parallel Computing Based on CUDA Technology. Journal of Physics: Conference Series, 803, p. 012152. doi: 10.1088/1742-6596/803/1/012152.

[6] Krishnamurthy, A., Samsi, S., and Gadepally, V. (2009). Parallel MATLAB Techniques in Image Processing. IntechOpen, London, UK. doi: 10.5772/7046.

[7] Demir, İ., and Sayar, A. (2014). Hadoop Optimization for Massive Image Processing: Case Study Face Detection. International Journal of Computers Communications & Control, 9.

[8] Saha, D., Darji, K., Patel, N., and Thakore, D. (2016). Implementation of Image Enhancement Algorithms and Recursive Ray Tracing using CUDA. 7th International Conference on Communication, Computing and Virtualization. [Online]. Available: https://www.sciencedirect.com/.

[9] ResearchGate. (2022). Analysis of Different Parallel Implementation of Image Processing Algorithms. [Online]. Available: https://www.researchgate.net/figure/Analysis-of-different-parallel-implementation-of-image-processing-algorithms_tbl1_297629054.

[10] Curiel, M., Calle, D. F., Santamaría, A. S., Suarez, D. F., and Flórez, L. (2018). Parallel Processing of Images in Mobile Devices using BOINC. Open Engineering, 8(1), 87-101. doi: 10.1515/eng-2018-0012.

[11] Skirnevskiy, I., Pustovit, A., and Abdrashitova, M. (2017). Digital Image Processing Using Parallel Computing Based on CUDA Technology. Journal of Physics: Conference Series, 803, 012152. doi: 10.1088/1742-6596/803/1/012152.

[12] Distributed Image Processing - Blackboard System. (n.d.). [Online]. Available: https://www.igi-global.com/dictionary/distributed-image-processing-blackboard-system/21835.

[13] Stamopoulos, C. (1975). Parallel Image Processing. IEEE Transactions on Computers, 24(4), 424-433.

[14] Addison, T., and Vallabh, S. (2002). Controlling Software Project Risks – an Empirical Study of Methods Used by Experienced Project Managers. Proc. SAICSIT, 128-140.

[15] Singh, S., and Kaur, K. (2015). Parallel Computing in Digital Image Processing. IJARCCE, 183-186. doi: 10.17148/IJARCCE.2015.4139.

[16] Babar, M.A., and Paik, H.Y. (2009). Using Scrum in Global Software Development: A Systematic Literature Review. Global Software Engineering, 4th IEEE Int. Conf., 175-184.

[17] Wright, A., and Jones, B. (2013). Efficient Parallel Algorithms for Large-Scale Image Processing. Journal of Parallel and Distributed Computing, 73(9), 123-130. doi: 10.1016/j.jpdc.2013.04.010.

[18] Tanaka, K., and Li, X. (2015). Techniques for Distributed Processing in Image Segmentation. International Conference on Computational Intelligence, 512-518. doi: 10.1109/ICCI.2015.73.

[19] Kumar, P., and Singh, R. (2017). High Performance Computing Techniques for Image Processing. Journal of Computational Science, 5(2), 101-109. doi: 10.1016/j.jocs.2017.01.004.

[20] Sharma, V., and Kapoor, M. (2014). Survey on Parallel Image Processing Using Various Architectures. International Journal of Emerging Technology and Advanced Engineering, 4(5), 412-418.