MOOCS RECOMMENDER BASED ON LEARNING STYLES

SERVICE ORCHESTRATOR FOR PARALLEL VIDEO CLASSIFICATION

Project ID: 19-089

Research Thesis

Liyanage A.Y.K.

B.Sc. (Hons) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

April 2019

SERVICE ORCHESTRATOR FOR PARALLEL VIDEO CLASSIFICATION

Project ID: 19-089

Research Thesis

B.Sc. (Hons) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

April 2019

DECLARATION

I declare that this is my own work and this dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology the non-exclusive right to reproduce and distribute my dissertation in whole or part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as article or books).

Student Name	Student ID	Signature
Liyanage A.Y.K.	IT16032798	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature: Date:

Signature of the supervisor: Date:

Signature of the co-supervisor: Date:

ABSTRACT

Videos are increasing in quality with the advancement of videography and software editing tools, and it has become increasingly time-consuming to classify videos. While GPU based classification provides a solution packed with speed, it comes with a high cost. At the same time, CPU based classification has been deemed slow. This approach is to find a solution that is better than running a classifier on a CPU based execution environment which also cost less or equal. Parallelization comes with the unique problem of having to deduce the outputs of broken-down workloads, into a single output. Furthermore, loosing the output of even a single node in a parallelized workflow can greatly affect the accuracy of classification. Therefore, this research aims to find a middle ground between drawbacks of GPU based classification and CPU based classification of videos and images, while introducing the benefits of distributed computing to the workflow.

Keywords: Containerization, Docker, Parallel Video Classification, Distributed Computing, Persistent Workloads

ACKNOWLEDGEMENT

This academic paper and work described in it were conducted as a part of 4th year research module (Comprehensive Design Analysis Project) under the degree program B.Sc. (Hons) in Software Engineering in Sri Lanka Institute of Information Technology. The fruition of this research and the associated software project are results of dedication of the 4 group members and the immense support given by a supervising panel.

Thus, we would like to express our gratitude towards Dr. Nuwan Kodagoda and Ms. Kushanara Suriyawansa, who are our supervisors as well as lecturers of Sri Lanka Institute of Information Technology. Completion of this academic work would not have been possible without their guidance and insight. We are extremely grateful to Dr. Malitha Wijesundara and Dr. Dasuni Nawinna, who reviewed and approved our research topic and provided valuable suggestions. Furthermore, we wish to express our gratitude to Mr. Jayantha Amaraarachchi, the Head of SLIIT Centre for Research who provided us with immense knowledge and guidance throughout.

Also, we would like to express our gratitude to all our colleagues and companions who helped us in testing and in various other ways to make our research come to fruition. At the same time, we would like to thank everyone who might not have been mentioned but has given their support and encouragement to us.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

Table 0.1:List of Abbreviation

MOOC	Massive Open Online Course
SIMD	Single Instruction Multiple Data-stream
VM	Virtual Machine
AWS	Amazon Web Services
ML	Machine Learning

1. INTRODUCTION

1.1. Background Literature

Video classification is carried out by classifying the individual frames of a video rather than the video itself as a whole. This involves thousands of repeated operations and using a neural network, this process can be parallelized. However, as Kyoung-Su Oh and Keechul Jung implies, utilizing a neural network built on top of FPGA based hardware is both fast and expensive [1].

Furthermore, they also suggest that GPUs can achieve better performance compared to CPUs in an image classification workload [2]. Even though GPUs can achieve better performance, they are expensive compared to traditional CPUs and requires more power to run.

However, both GPU and CPU based approaches that involve a neural network rely on parallelism within the same device, but parallelism can be achieved by distributing the workload across multiple processing nodes, be it GPU based, or CPU based. In general, there are 4 types of parallelisms[3].

- Single Instruction Single Data Stream **(SISD)**
- Multiple Instruction Single Data Stream **(MISD)**
- Single Instruction Multiple Data Stream **(SIMD)**
- Multiple Instruction Multiple Data Stream **(MIMD)**

Image classification relies on SIMD approach where the same operation set is applied to a large dataset. Therefore, by feeding different set of images of a video to a distributed set of processing nodes, a SIMD based classification process can be achieved on a higher level.

Out of above approaches to parallel processing, given that we have a large amount of image frames that belong to different videos and the fact that we only train one image classifier to identify a range of video styles, the ideal approach is to use SIMD where we have multiple nodes, each running the same classifier(thus Single Instruction) while each node is fed different image data (thus Multiple Data stream). While this

seem ideal in theory, the major concern involving this approach is analyzing the output given for each frame analyzed.

In order to achieve the consistency across all processing nodes, a containerization solution such as Docker can be used. Docker containers are faster than VMs(Virtual Machines) and Kubernetes has made it convenient and robust to run and manage a cluster of Docker containers [5].

1.2. **Research Gap**

With the rise of machine learning and deep learning, GPU based processing has become the norm for such workloads. Due to its inherent nature of having considerably more computing units compared to the traditional CPUs, GPUs excel at parallel processing. However, GPU based parallel processing solutions using convolution neural networks are more expensive compared to their CPU counterpart. Even with such a high price penalty in GPU based parallel computing, it is shown to provide 25 to 49 times more performance compared to CPU based solutions [6].

Video classification involves splitting a video into individual image frames, classifying each image frame and finally deducing the classification of the original video by analyzing the classifications given to each individual image frame. While Apache Hadoop is used as another solution for parallel processing workloads, it involves a considerable amount of modifications to existing libraries to be compatible with image and video processing and classification using OpenCV [4] to aid in "classifying individual image frame" phase. Therefore, even though solutions such as GPU based parallel processing and Apache Hadoop Cluster based processing exists with thorough researches done on them, the research gap that involves distributing both CPU and memory to achieve better performance compared to centralized CPU and memory is high. Furthermore, associating containerization to aid in distributed parallel computing that involves machine learning is an area that has been sparsely researched. Hence, this research aims to dive deeper into using Docker as a containerization medium for a video classifier that will be distributed across multiple servers with their own memory and CPU to determine how much of a performance

improvement can be achieved compared to a similarly powerful centralized processing unit.

1.3. **Research Problem**

FPGA based, GPU based video classification and CPU based video classification represent two extreme ends of the spectrum, one being expensive and fast while the other being relatively cheap, yet slow. While it is possible be achieve parallelization within CPU based workloads by introducing a thread-based implementation, such a solution would face multiple bottlenecks such as shared memory and CPU as well as a low number of available Disk IOPS when multiple image files are read from the disk by multiple threads at the same time.

At the same time, it is crucial to understand that every chunk of the bigger problem is paramount in a parallelized workload in order to arrive at an accurate conclusion at the end. What this implies is that if a dataset is divided into multiple smaller datasets, the processing done on every smaller dataset must be preserved until all such smaller datasets are processed. While use of ECC (Error Correcting Memory) and constantly writing to persistent media such as databases have become the norm, a simpler framework for ensuring the persistency of all smaller workloads of a bigger problem while providing a full-fledged interface to divide the workload and analyze the output of each such divided workload to derive a single output for the bigger problem only exists in the form of Apache Hadoop which is not fully compatible with an video based machine learning classification workload.

Therefore, this research is conducted to address above issues associated with distributed parallel classification of video files using CPUs and devise a framework that provides orchestrating features of Apache Hadoop by marking smaller chunks of a large video file where each of such chunk will be classified by a different instance of the same classifier, which creates a parallel and distributed workforce. Given that the processing is distributed across multiple nodes, the feasibility of using a message queue as opposed to using shared memory will be assessed.

1.4. **Research Objectives**

The research's primary focus is to identify the performance improvement of distributed and parallelized classification of videos using multiple CPUs with its own memory and storage as opposed to CPU based video classification using a single powerful computing unit.

However, the ultimate result of this research would be to present a framework that consist of following aspects,

- A medium to orchestrator a larger workload into smaller workloads where image or video classification is involved.
- A medium to reduce the output/classification of each such smaller workload to derive a single classification on the larger workload.
- A medium to enforce and provide high persistency for all workloads and outputs to aid in deriving a highly accurate output/classification, so that the smaller workloads are not lost even if a node crash. This is paramount to the accuracy of the final classification since the classifications of all the small workloads must be present to have better accuracy.

2. METHODOLOGY

2.1. **Methodology**

2.1.1. **Orchestrating a workload into smaller workloads.**

The main focus of this research is classification of videos parallelly. Therefore, a single video file that has to be classified can be represented as a typical workload. Hence, breaking down a single workload involves logically marking smaller chapters of the video file. This is referred to has chunking henceforth. Chunking of a single video file is achieved as follows.

ΔT_c : $Duration\ of\ a\ chunk\ in\ seconds\ (Pre-determined)$.

ΔT : $Duration\ of\ original\ video\ file$.

f : *Frames per second metric of original video file*

$\sum F$: *Total number of frames of original video file*

$\sum F_c$: *Total number of frames in a single chunk*

$$f = \frac{\sum F}{\Delta T} \tag{1}$$

$$\sum F_c = f * \Delta T_c \tag{2}$$

Once number of frames that should be in a single video chunk is calculated by using equation (1) and (2), this calculated value can be used as the step value by the actual classifier when tasked with classifying a specific chunk. The figure 2.1 depicts how $\sum F_c$ value can be used to identify where each logical chunk starts and ends on the actual video file.

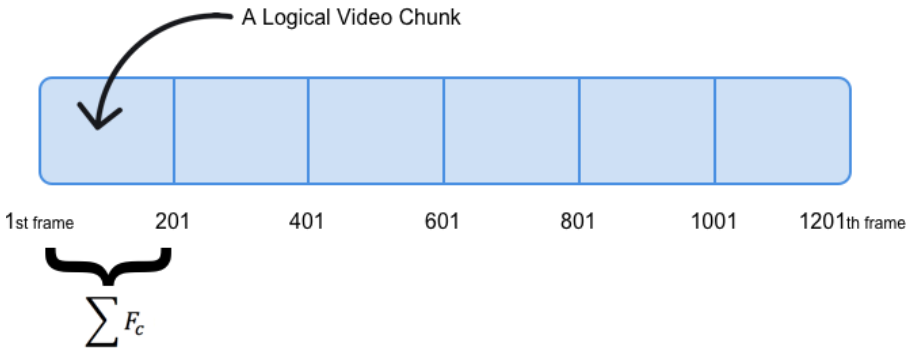

Figure 2.1: How a video file is logically chunked

Furthermore, it is important to note that the video file is not physically split since that will introduce another overhead to the overall process. Instead, $\sum F_c$ value is used as a step value for the classifier to identify the start frame and the end frame of a chunk as shown in equation (3).

$$frame\_start_i : Starting\ frame\ of\ the\ i^{th}\ chunk$$

$$frame\_end_i \ : Ending\ frame\ of\ the\ i^{th}\ chunk$$

Using equation (2):

$$frame\_end_i = frame\_start_{i+\sum F_c} \quad (3)$$

2.1.2. Message queue as the workload coordinator

By using a message queue to communicate the smaller workloads to service workers, a persistent communication medium is introduced. For this particular implementation, RabbitMQ is used. By its nature, a message queue like RabbitMQ will keep the messages preserved until they are acknowledged. By configuring the service workers to acknowledge a message only after processing the workload mentioned in the message, all workloads will be processed even if a service worker crashes since the next service worker will pick up the task.

2.1.3. Containerizing a video classifier to act as a service worker

Docker is used as the containerization technology since it supports mounts which makes it possible to access the files within the host server with ease. Furthermore, the containerized service worker also acts as a subscriber to the message queue in order to receive workloads in the form of messages. Each message contains information about a logical chunk of a video file that must be classified.

For this implementation, a pre-trained model provided by VGG16 is used and the Docker container is based on jjanzic/docker-python3-opencv docker-image [7] which is pre-configured with Python 3 as well as OpenCV. However, following Python modules must be installed for the classifier to work properly.

- pika
- keras
- tensorflow

- pillow

Furthermore, Docker-Compose is used to build the Docker image along with environment variables that contains following information.

- Message queue endpoint
- Message queue username and password
- Message queue name to subscribe

Another use of Docker-Compose is to expose the video files directory in the host server to the Docker container. This video files directory contains video files that are yet to be classified.

2.1.4. Reduce the classifications on smaller workloads into a single classification

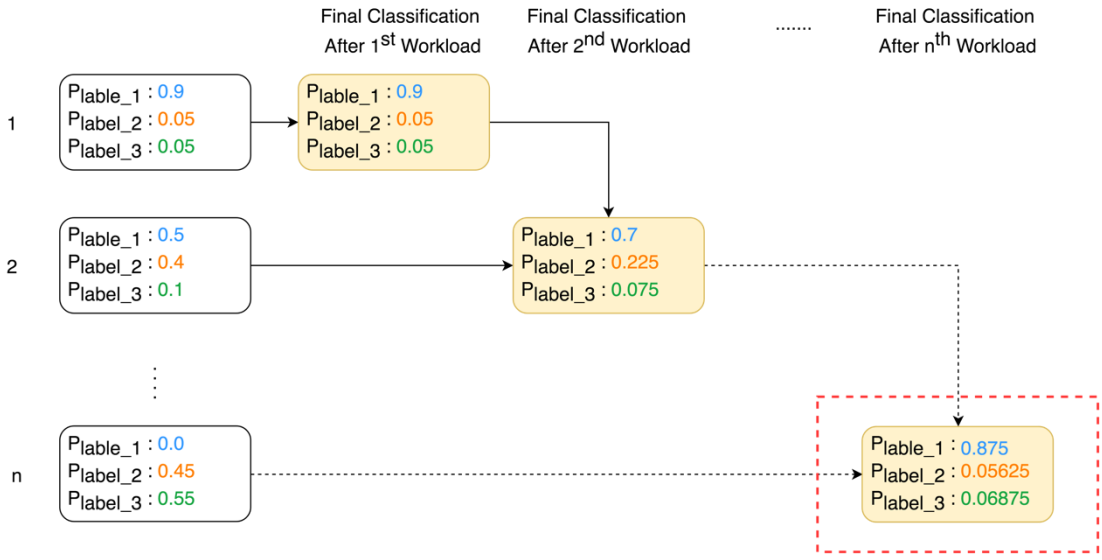

Figure 2.2: Output reduction of video chunks that belong to a single video file

All video chunks (smaller workloads) of a single video file (bigger workload) will not be classified at the same time. Therefore, final classification is re-evaluated every time a chunk that belongs to the initial video file is classified. When the first chunk is classified, its classification is taken as the classification of the video file. However, when a subsequent chunk is classified, its classification and the final classification are

averaged-out and the averaged-out value becomes the new final classification. This process is continued until all the chunks that belong to a single video file are classified.

At a higher level, a similar mathematical calculation is done on a typical video classification implementation where number of frames that were given a specific label is divided by the total number of frames classified. This is done since the output of a video classification is the likeliness of the video belonging to a certain label as percentages.

However, by re-evaluating the final classification every time a chunk is classified as opposed to waiting till all chunks are classified, it is possible to reduce the waiting time and high processing overhead which can be caused by a large number of un-evaluated outputs of chunks if the video file is lengthy or has a higher frame rate. Even though final classification is re-evaluated every time a chunk is classified during the implementation to reduce the processing overhead, the corresponding mathematical formula to obtain the final classification for a video file (C_F) after all chunks are classified can be expressed as show in equation (4),

$$C_i : Classification\ of\ i^{th}\ chunk(smaller\ section\ of\ the\ video)$$

$$C_F : \ Final\ classification$$

$$C_F = \frac{\sum_i^n C_i}{n} \tag{4}$$

2.2. **Testing & Implementation**

The implementation is fully run on AWS (Amazon Web Services), using their EC2 (Cloud based servers) platform. To emulate a parallelized and distributed compute platform, 2 equally equipped servers were used. Centralized compute platform was emulated by using a single server which was twice as powerful compared to a single node of the distributed compute platform [Table 2.1].

Table 2.1: On-demand Pricing for EC2 Instances (AWS) in North Virginia Region

Instance Type	**Operating System**	**Number of vCPU**	**RAM (GB)**	**Price Per Hour (USD)**
t2.micro	Amazon Linux 2	1	1	0.013
t2.medium	Amazon Linux 2	2	4	0.052

Source: AWS Pricing [8].

Time taken to classify 5000 image frames by the centralized compute platform and distributed compute platform was assessed separately. Same workload was repeated 3 times on each platform and the average was taken for the final consideration. For determining the time taken to classify 5000 images frames by 2 nodes in the distributed compute platforms, the time taken by each node to classify 2500 frames were added together as the total time taken for that run.

3. RESULTS & DISCUSSION

3.1. **Results**

Time taken for classifying 5000 image frames of a video file was considered as the workload for testing. This workload was processed by

- By a single AWS EC2 instance that had 2 virtual CPUs.
- By two AWS EC2 instances each having half the processing power of the instance mentioned in the former point.

The idea here was to distribute the processing power of a centralized server across two servers. Similarly, for the distributed workload, each instance was tasked with classifying 2500 frames. After repeating the said scenarios 3 times, the centralized server completed classifying 5000 frames in 3962 seconds. The 2 distributed servers took a total of 2122 seconds collectively, to classify a total of 5000 frames [Table 3.1].

Table 3.1: Compute Time Taken for Classifying 5000 Frames - Centralized vs Distributed

Workload Type	**Instance Type**	**vCPUs per Instance**	**Memory per Instance**	**Number of Instances**	**Total Compute Time Taken (Seconds)**			
					Run #1	Run #2	Run #3	Average
Centralized	t2.medium	2	4GB	1	3545	4157	4183	3962
Distributed	t2.micro	1	1GB	2	2168	2107	2090	**2122**

The average cost of computing was calculated by using general purpose compute platform provided by AWS [8]. Based on the compute time taken by centralized and distributed workloads, the calculated cost shows ~86% less cost involved with distributing the workload [Table 3.2].

Table 3.2: Cost Benefit - Centralized vs Distributed

Instance Type	**Specifications**	**Instance Count**	**Cost Per Hour for 1 Instance (USD)**	**Total Compute Time (Seconds)**	**Total Compute Cost (USD)**
t2.medium	2 vCPU 4GB RAM	1	0.052	3,962	0.057224074
t2.micro	1 vCPU 1GB RAM	2	0.013	2,122	0.007661574

3.2. **Research Findings**

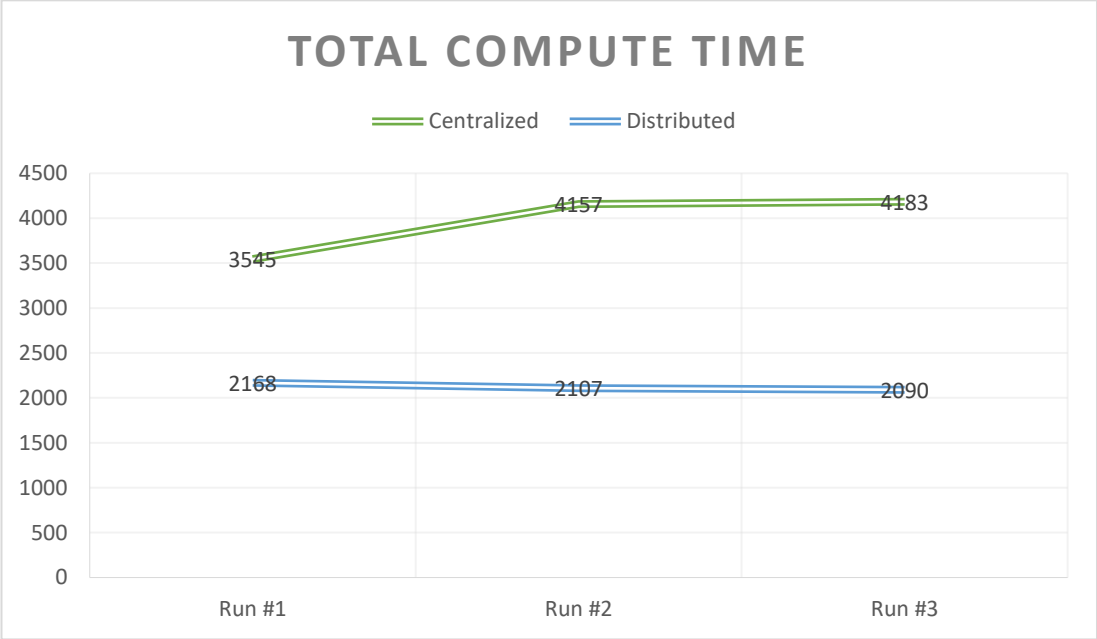

Figure 3.1: Total Compute Time Taken by Distributed vs Centralized Compute Platforms

Give that the distributed platforms collective processing power is equal to that of the centralized platform, ideally, the time taken by two platforms must be equal or close to each with within the margin of error. However, it was found out that this is simply not the case. One reason for that is the CPUs in both platforms have equal cache memories. Therefore, the distributed platform does have an advantage over its centralized counterpart.

Furthermore, it was found out that running the classifier twice at the same time with half the workload assigned to each run in the centralized server does not achieve the same efficiency as the 2 distributed servers since this time the disk of the centralized server is accessed by 2 processes which reduce its available IOPS (Input Output Operations per Second).

3.3. **Discussion**

Based on the testing that was done to determine the performance gain that can be achieved by parallelizing an image classification workload across a distributed computing platform can be said to have yielded positive results. While this method of

classifying videos and images clearly does not yield performance gains that are comparable to GPU based parallel processing, it does however yield more than 75% performance improvements for a lesser cost as well.

4. CONCLUSION

Findings of this research show how distributed computing can be utilized to overcome the drawbacks associated with parallelizing workloads within the same node such as low cache, low disk IOPS. Furthermore, it depicts how containerization can be used to deploy multiple nodes across servers with consistency and ease. While it is true that GPU based parallel processing is still superior, distributed computing for parallelized workflows propose a viable, cost effective solution compared to traditional CPU based processing.

5. REFERENCES

[1] K.-S. a. J. K. Oh, "GPU implementation of neural networks," *Pattern Recognition,* vol. 37, no. 6, p. 1311, 2004.

[2] K.-S. a. J. K. Oh, "GPU implementation of neural networks," *Pattern Recognition,* vol. 37, no. 6, p. 1312, 2004.

[3] A. Macfarlane, S.E. Robertson and J.A. Mccann, *Parallel computing in information retrieval – an updated review,* Emerald insight, 1997.

[4] V. Anton, C.́ . R.́ .-C.́ es, J. Ejarque and R. M. Badia, "Transparent execution of task-based parallel applications in Docker with COMPSuperscalar".

[5] N. a. C. Y.-s. a. W. J.-l. Zhang, "Image parallel processing based on GPU," *2010 2nd International Conference on Advanced Computer Control,* vol. 3, pp. 367--370, 2010.

[6] H. Tan and L. Chen, "AN APPROACH FOR FAST AND PARALLEL VIDEO PROCESSING ON APACHE HADOOP CLUSTERS".

[7] jjanzic, "jjanzic/docker-python3-opencv - Docker Hub," June 2019. [Online]. Available: https://hub.docker.com/r/jjanzic/docker-python3-opencv/. [Accessed 12 August 2019].

[8] AWS, "Amazon EC2 T2 Instances," Amazon Web Services, 2019. [Online]. Available: https://aws.amazon.com/ec2/instance-types/t2/. [Accessed 12 August 2019].