



PARALLELIZATION OF SEQUENTIAL COMPUTER VISION ALGORITHMS ON BIG-DATA USING DISTRIBUTED CHUNK-BASED FRAMEWORK

By

Norhan Magdi Sayed Osman

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2018

PARALLELIZATION OF SEQUENTIAL COMPUTER VISION ALGORITHMS ON BIG-DATA USING DISTRIBUTED CHUNK-BASED FRAMEWORK

By

Norhan Magdi Sayed Osman

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Under the Supervision of

Prof. Dr. Hossam Aly Hassan Fahmy Dr. Mohamed Mohamed Rehan

Professor

Chief Technology Officer

Electronics and Communications Engineering Department

Avidbeam Technologies

Faculty of Engineering, Cairo University

**FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT**

2018

PARALLELIZATION OF SEQUENTIAL COMPUTER VISION ALGORITHMS ON BIG-DATA USING DISTRIBUTED CHUNK-BASED FRAMEWORK

By

Norhan Magdi Sayed Osman

A Thesis Submitted to the
Faculty of Engineering at Cairo University
in Partial Fulfillment of the
Requirements for the Degree of
MASTER OF SCIENCE
in
Electronics and Communications Engineering

Approved by the Examining Committee:

Prof. Dr. Hossam Aly Hassan Fahmy, Thesis Main Advisor

Dr. Mohamed Mohamed Rehan, Advisor
Chief Technical Officer, AvidBeam Technologies

Prof. Dr. Elsayed Eissa Abdo Hamyed, Internal Examiner

Prof. Dr. Khaled Mostafa Elsayed, External Examiner
Faculty of Computers and Information, Cairo University

FACULTY OF ENGINEERING, CAIRO UNIVERSITY
GIZA, EGYPT
2018

Engineer's Name: Norhan Magdi Sayed Osman
Date of Birth: 27/5/1990
Nationality: Egyptian
E-mail: nmbuckla@live.com
Phone: 01224209955
Address: 12D, Maadi Gardens Compound, Daary
Maadi, Katamya
Registration Date: 1/3/2014
Awarding Date: / /2018
Degree: Master of Science
Department: Electronics and Communications Engineering



Supervisors:

Prof. Dr. Hossam Aly Hassan Fahmy
Dr. Mohamed Mohamed Rehan

Examiners:

Prof. Dr. Hossam Aly Hassan Fahmy (Thesis Main advisor)
Dr. Mohamed Mohamed Rehan (Advisor)
Chief Technology Officer, AvidBeam
Prof. Dr. Elsayed Eissa Abdo Hamyed (Internal examiner)
Prof. Dr. Khaled Mostafa Elsayed (External examiner)
Faculty of Computers and Information,
Cairo University

Title of Thesis:

PARALLELIZATION OF SEQUENTIAL COMPUTER VISION
ALGORITHMS ON BIG-DATA USING DISTRIBUTED CHUNK-BASED
FRAMEWORK

Key Words:

BiG-Data; Computer Vision; Parallel Computing; Video Processing; Video-Chunks

Summary:

In this thesis, we propose a complete framework that enables big-data frameworks to run sequential computer vision algorithms in a scalable and parallel way. Our idea is to divide the input video files into small chunks that can be processed in parallel without affecting the quality of the resulting output. We developed an intelligent data grouping that enables chunk-based framework to distribute these data chunks among the available resources and gather the results out of each chunk faster than the standalone sequential application.

Acknowledgements

I would like to thank my father, my husband and the rest of my family for being patient and supportive during my MSc. period. They believed in me and encouraged me to continue my work with endless love and understanding. I would also like to thank my supervisors, Dr. Hossam Fahmy and Dr. Mohamed Rehan for their continuous support, patience and guidance to me in order to enhance my work and correct my mistakes during this thesis. I really want to thank all my colleagues at AvidBeam who stood beside me to understand several computer vision basics and helped me to mature my developed work. I would like to thank Dina Helal, Menna Ghoneim, Rana Morsi, Ahmed Hegazy, Eslam Ahmed, Mickael Sedrak, Raghda Hassan and Afnan Hassan as well. Special thanks goes to Dr. Hani El Gebaly, Hossam Sami and AvidBeam for providing support, time and hardware capabilities to develop and test my proposed work and reach the final results.

Dedication

To the soul of my *mother*, I know your are proud of me.

Table of Contents

Acknowledgements	i
Dedication	ii
Table of Contents	iii
List of Tables	v
List of Figures	vi
List of Abbreviations	ix
List of Terms	x
Abstract	xi
1 INTRODUCTION	1
1.1 Motivation and Overview	1
1.2 Introduction to Big-Data	2
1.2.1 Definition	2
1.2.2 Applications	4
1.3 Problem Description	6
1.4 Challenges	7
1.5 Thesis Organization	7
2 BIG-DATA TECHNOLOGIES	9
2.1 Introduction	9
2.2 Apache Hadoop	10
2.2.1 Overview	10
2.2.2 Components and Cluster Architecture	11
2.2.3 Scheduler	15
2.2.4 Limitations	16
2.3 Apache Spark	16
2.3.1 Overview	16
2.3.2 Components and Cluster Architecture	17
2.3.3 Scheduler	20
2.3.4 Limitations	21
2.4 Apache Storm	22
2.4.1 Overview	22
2.4.2 Components and Cluster Architecture	22
2.4.3 Scheduler	25
2.4.4 Storm Stream Grouping	26
2.5 Comparison between Big-Data Technologies	28
2.6 Conclusion	29

3	LITERATURE REVIEW	31
3.1	Introduction	31
3.2	Computer Vision and Video Processing Overview	31
3.3	Previous Work	33
3.4	Conclusion	39
4	DISTRIBUTED CHUNK-BASED BIG-DATA PROCESSING FRAMEWORK	41
4.1	Introduction	41
4.2	Basic Concept	41
4.3	Architecture	45
4.3.1	Resources Calculation Unit (RCU)	47
4.3.2	Data Splitting and Feed unit (DSFU)	54
4.3.3	Decision Making and Resources Mapping Unit (DMRMU)	56
4.4	Proposed Framework Key Competencies	61
4.5	Conclusion	61
5	EXPERIMENTAL RESULTS	63
5.1	Introduction	63
5.2	Evaluation Process Setup	63
5.2.1	Hardware Setup	63
5.2.2	Storm Topology Structure	63
5.2.3	Evaluation Metrics	66
5.3	Evaluation Test Cases	66
5.3.1	Video Summarization	67
5.3.2	Face Detection	70
5.3.3	License Plate Recognition	73
5.3.4	Heatmaps	75
5.3.5	Testing multiple video files	78
5.4	Results Observations	81
5.5	Conclusion	82
6	CONCLUSION AND FUTURE WORK	85
6.1	Conclusion	85
6.2	Future Work	86
	References	87

List of Tables

2.1	Comparison between big-data technologies [17].	28
5.1	Storm configuration for automatic back pressure.	65
5.2	Testing video specifications for each VP algorithm.	66
5.3	Video summarization algorithm evaluation metrics.	68
5.4	Face detection algorithm evaluation metrics.	71
5.5	License plate recognition algorithm evaluation metrics.	74
5.6	Heatmaps algorithm evaluation metrics.	77
5.7	Multiple files evaluation metrics.	79
5.8	Multiple files FPS metrics.	79

List of Figures

1.1	Number of hours of uploaded videos to YouTube till July 2015. [4]	1
1.2	Triple Vs of big-data [9]	3
1.3	Wikibons Big-data market forecast [12]	5
2.1	Big-data processing steps [11].	9
2.2	HDFS architecture with data replication [19].	11
2.3	HDFS abstract architecture of master and slave nodes.	12
2.4	Hadoop mapreduce.	13
2.5	Hadoop simple working flow [24].	14
2.6	Hadoop YARN architecture [26].	15
2.7	Spark DAG.	18
2.8	Spark different deployment schemes [30].	18
2.9	Spark components.	19
2.10	Spark cluster architecture.	20
2.11	Representation of Storm topology containing spouts and bolts.	23
2.12	Storm abstract architecture.	24
2.13	Storm worker node internals.	25
2.14	Storm different stream grouping types [52].	27
3.1	Computer vision steps sequence [55].	32
3.2	Distributed stream processing [60].	34
3.3	Distributed video transcoding abstract architecture [60].	35
3.4	Video transcoding using mapreduce-based cloud computing [61].	35
3.5	Big-data video monitoring system based on Hadoop [64].	37
3.6	Low-delay Video Transcoding initial topology [65].	38
3.7	Storm topology for scalable and intelligent real-time video surveillance framework [67].	39
4.1	Storm topology for CV processing using DCB-Framework.	43
4.2	DCB-Framework architecture in Storm topology.	46
4.3	The RCU block diagram.	48
4.4	The RCU steps to calculate number of frames and free resources.	49
4.5	The RCU working steps to calculate actual chunk size.	51
4.6	The RCU calculation of actual needed number of bolt instances.	53
4.7	Splitting video files in DSFU.	54
4.8	Illustration of video frames sequence consisting of I, P and B frames [70].	55
4.9	Sending chunks frames to spouts queue using parallel chunk-based splitting.	55
4.10	DCB-Framework architecture in Storm topology.	56
4.11	Redis sample view of Assignment Map and Tasks Queue.	57
4.12	Selecting Task _{ID} for each video frame.	60
5.1	Storm slave machine components for VP.	65
5.2	Video Summarization algorithm output sample.	68
5.3	Total processing time of testing video for video summarization.	69

5.4	Processing time speed up of testing video for video summarization. . . .	69
5.5	Complete latency of testing video for video summarization.	70
5.6	Face detection algorithm output sample [81].	71
5.7	Total processing time of testing video for face detection.	72
5.8	Processing time speed up of testing video for face detection.	72
5.9	Complete latency of testing video for face detection.	73
5.10	LPR algorithm output sample.	73
5.11	Total processing time of testing video for license plate recognition. . . .	74
5.12	Processing time speed up of testing video for license plate recognition. . .	75
5.13	Complete latency of testing video for license plate recognition.	75
5.14	Sample output of heatmaps algorithm.	76
5.15	Total processing time of testing video for heatmaps.	77
5.16	Processing time speed up of testing video for heatmaps.	78
5.17	Complete latency of testing video for heatmaps.	78
5.18	Total processing time for multiple files.	80
5.19	Processing time speed up for multiple files.	80
5.20	Complete latency for multiple files.	81

List of Abbreviations

AM	Application Master
API	Application programming interface
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CV	Computer Vision
DAG	Direct Acyclic Graph
DCB-Framework	Distributed Chunk-Based Framework
DBMS	Database Management System
DSFU	Data Splitting and Feed Unit
DMRMU	Decision Making and Resources Mapping Unit
FIFO	First In First Out
FPS	Frames Per Second
GFS	Googles File System
GPU	Graphical Procoessing Unit
GOP	Group of pictures
HDFS	Hadoop Distributed File system
JAR	Java Archive
JVM	Java Virtual Machine
LPR	License Plate Recognition
MPEG	Moving Picture Experts Group
NM	Node Manager
NFS	Network File Systems
OCR	Optical Character Recognition
OpenCV	Open Source Computer Vision Library
OpenMP	Open Multi-Processing
OS	Operating System
PC	Personal Computer
QoS	Quality of Service
RCU	Resources Calculation Unit

RDD	Resilient Distributed Dataset
ROI	Region Of Interest
RTSP	Real Time Streaming Protocol
SIMR	Spark In MapReduce
SQL	Structured Query Language
RM	Resource Manager
UI	User Interface
VM	Virtual Machine
VP	Video Processing
YARN	Yet Another Resource Negotiator

List of Terms

Bolts	Data processing entities at Storm.
Executor	A thread runs at Storm worker process.
Kafka	An open source stream processing tool.
Kestrel	A simple, distributed message queue system
MapReduce	Parallel data processing scheme with two stages: Map and Reduce
Nimbus	A Java daemon runs at Storm master node.
Scheduler	An algorithm used to distribute processing resources on big-data tools
Storm	An open source big-data tool
Spouts	Storm data sources.
Supervisor	A Java daemon runs at Storm worker node
Thrift	A software framework used for scalable cross-language services development
Topology	A directed graph that connects Storm spouts and bolts together.
Zookeeper	An entity that governs the communication between Storm nimbus and supervisors

Abstract

The research presented in this thesis addresses the parallelization of sequential computer vision algorithms, such as motion detection, tracking, etc., on big-data tools. Computer vision sequential algorithms have restrictions on how the video frames should be processed. In these algorithms, the inter-relation between successive frames is important part of the algorithm sequence as the result of processing one video frame depends on the result of the previous processed frame(s).

Most of the present big-data processing frameworks distribute the input data randomly across the available processing units to utilize them efficiently and preserve working load fairness. Therefore, the current big-data frameworks are not suitable for processing video data with inter-frame dependency. When processing these sequential algorithms on big-data tools, splitting the video frames and distributing them on the available cores will not yield the correct output. Consequently, the advantage of the processing sequential algorithms on big-data framework becomes limited only to certain cases where video streams are coming from different input sources.

In this thesis, we propose a complete framework that enables big-data tools to execute sequential computer vision algorithms in a scalable and parallel way with limited modifications. Our main objective is to parallelize the processing operation in order to speed up the required processing time. The main idea is to divide the input big-data video files into small chunks that can be processed in parallel without affecting the quality of the resulting output. We have developed an intelligent data grouping algorithm that distributes these data chunks among the available processing resources and gather the results out of each chunk. A parallelized chunk-based data splitter was used to provide the input data chunks concurrently for parallel processing. Then, our grouping algorithm makes sure that all frames that belong to the same chunk are distributed in order to the associated processing cores.

To evaluate the performance of the developed chunk-based framework, we conducted several experimental tests. We used Apache Storm as our big-data framework for its real-time performance. Storm framework was modified to support input video frame splitting and parallel execution. We examined the behavior of our proposed framework against different number of chunks over one hour testing videos. In our evaluation, we used several sequential computer vision algorithms including face detection, video summarization, license plate recognition (LPR), and heatmaps. Those algorithms were integrated into