Ain Shams University
Faculty of Computer and Information Sciences
Information Systems Department

Efficient Processing of Continuous Queries based on Cloud Computing

A thesis submitted in partial fulfillment of the requirements for
the degree of MSc in Computer and Information Sciences
To
Information Systems Department,
Faculty of Computer and Information Sciences,
Ain Shams University

By

Fatma Mohamed Mahmoud Najib
Teaching Assistant at Information Systems Departments
Faculty of Computer and Information Sciences, Ain Shams University

Under Supervision of

Prof. Dr. Mohamed Fahmy Tolba
Scientific Computing Department
Faculty of Computer and Information Sciences, Ain Shams University

Prof. Dr. Nagwa Lotfy Badr
Information Systems Department
Faculty of Computer and Information Sciences, Ain Shams University

Assoc. Prof. Rasha Mohamed Ismail
Information Systems Department
Faculty of Computer and Information Sciences, Ain Shams University

September – 2016

Acknowledgement

First of all thanks to Allah for giving me the opportunity and the capability to finish this thesis. Great thanks to everyone who assists and encourages me to complete this work.

I would like to deeply thank Prof. Dr. Mohamed Fahmy Tolba for his leadership, support and encouragement all time.

Also, I would like to thank Prof. Dr. Nagwa Badr for her supervision, instructions and help throughout this thesis.

I would also like to thank Assoc. Prof. Rasha Ismail for her guidance, assistance and support through the work.

Finally, all thanks to my sweetie mother for being the reason to any success in my life. A special gratitude and love to my father for his great and unfailing support and love. I would also like to thank every member of my family for their encouragement.

Abstract

Many recent applications in several domains such as sensor networks and financial applications generate continuous, rapid, and time varying datasets which are called data streams. Data streams require real time processing. In most database systems, the query optimizers select a single plan to process all streams tuples which is not efficient with the streams changeable nature. Also there is a little research effort has been made towards the multiple data streams queries' simultaneous execution. In addition applying data streams' multi-directional optimization over an optimized and elastic environment has not been much considered. Thus in this thesis we proposed combined frameworks and different optimization algorithms to solve these problems.

First, we proposed the optimized query mesh for data stream (OQMDS) framework. In which data streams are processed over multiple query plans. Each plan is used to process a cluster of data that have nearest properties. We proposed the Optimized Iterative Improvement Query Mesh (OII-QM) and Non-Search based Query Mesh (NS-QM) algorithms, to efficiently generate the multiple plans. The proposed algorithms improves the optimization time by 70.3%, the execution time by 21.8%, the execution overheads by 80% and the memory usage by 96% over the II-QM algorithm.

Then in this thesis the Continuous Query Optimization based on Multiple Plans framework for data streams over the cloud environment (CQOMP) was proposed. CQOMP provides an optimized streams processing over the cloud. The Optimized Multiple plans (OMP) and the Auto Scaling Cloud Query Mesh (AS-CQM) algorithms were proposed for streams processing over multiple query plans on cloud computing. The proposed OMP improves the performance in terms of the execution time by 83.5%, 47.7%, and the throughput by 69.7%, 40% over the operator-set-cloud methodology and the NS-QM algorithm. The elastic configurations of the proposed AS-CQM increases utilizing cloud processing resources by 33.8% and reduces the costs by 50% over the static configuration.

Finally in this thesis the multiple queries optimization based on partitioning (MQOP) framework was proposed to efficiently execute multiple queries simultaneously on the cloud environment. The optimized global plan (OGP) and the optimized global plan based on partitioning (OGPP) algorithms were proposed for jointly executing multiple continuous queries over an optimized global plan to each cluster of data on the cloud. The proposed OGP improves the execution time by 80% and the throughput by 76.8% over the operator tree technique. The proposed OGPP algorithm improves the performance in terms of execution time by 61.1%, 72.6%, and the throughput by 55.5%, 66.5% over the compile time optimization method and the operator tree technique.

Table of Contents

Chapter **Page**

Acknowledgement... I
..
Abstract.. II
.
Table of Contents... III
List of Figures.. VII
List of Tables.. X
List of Algorithms.. XI
List of Abbreviations.. XII

1- Introduction 1

1.1 Overview.. 2
1.2 Problem Definition.. 3
1.3 Objective... 4
1.4 Contributions.. 5
.
1.5 Thesis Organization.. 6

2- Background 8

2.1 Data Streams Overview.. 9
2.2 Data Streams Processing and Queries.. 11

3- Related Works 14

3.1 Data Streams Continuous Queries Optimization......................... 15
3.1.1 Data Streams Processing using Multiple Query Plans.. 16
3.1.2 Continuous Query with Probabilistic Guarantees......... 19
3.1.3 Continuous Top-k Query Processing.............................. 20
3.1.4 Continuous Nearest Neighbor Query Processing……… 20
3.1.5 Multiple Continuous Queries Processing………………. 22
3.2 Data Streams processing Environments...................................... 25

3.2.1 Parallel and Distributed Stream Processing.................... 25

A. Data Streams Processing based on Map Reduce Framework……………………… 25

B. Data Streams Processing in Traffic Distributed Networks……….. 26

C. Controlling Streams Processing on Overload Conditions…………………………………… 27

3.2.2 Data Streams Processing based on Cloud Environment... 29

4- The Proposed Data Streams Continuous Query Optimization 31

4.1 Introduction 32

4.2 The Proposed Optimized Query Mesh for Data Stream Framework (OQMDS).. 33

4.2.1 The Initial Query Mesh Preparation Sub-System………. 34

4.2.2 The Continuous Query Optimization Sub-System……… 34

4.2.3 The Pre-Execution Sub-System……………………… 35

4.2.4 The Execution Sub-System……………………………… 35

4.3 The Proposed Continuous Query Optimization Algorithms........ 36

4.3.1 The Proposed Optimized Iterative Improvement Query Mesh (OII-QM) Algorithm................................... 36

A. Partitions Lattice.. 37

B. Search Strategy 38

C. QM Solution... 38

D. Running Example…………………………….. 39

4.3.2 The Proposed Non-Search Based Query Mesh (NS-QM) Algorithm.. 40

A. QM Solution………………………………….. 40

B. Running Example 41

4.4 Experimental Results and Analysis of the Proposed OII-QM and NS-QM Algorithms... 42

4.4.1 The Average Optimization Time……………………. 43

4.4.2 The Average Execution Time……………………….. 44

4.4.3 The Average Execution-Runtime Overhead………… 45

4.4.4 The Optimization Memory Usage……………………… 46

4.5 Discussion.. 48

5- The Proposed Data Streams Processing on Cloud Computing 49

5.1 Introduction 50

5.2 The Proposed CQOMP Framework.................................. 51

5.2.1 The Initial Partitioning Sub-System............................... 52

5.2.2 The Continuous Query Optimization Sub-System......... 53

5.2.3 The Streams' Tuples Pre-Processing Sub-System.......... 53

5.2.4 The Elastic Cloud based Streams Execution Sub-System…………… 54

5.3 The Proposed Optimized Multiple Plans (OMP) Algorithm......... 54

5.3.1 Running example……………………………………… 57

5.4 Experimental Results and Analysis of the Proposed OMP Algorithm........................ 57

5.4.1 Cloud based Algorithms.. 59

A. The Average Overall Execution Time............. 59

B. The Average Throughput................................ 60

5.4.2 Algorithms based on Multiple Query Plans............. 62

A. The Average Overall Execution Time............ 62

B. The Average Throughput............................. 63

5.4.3 The Average Improvement Percentages of Execution Times for the Proposed OMP Algorithm..................... 64

5.4.4 Results for Real Amazon Cloud Environments…… 65

A. The Average Overall Execution Time……… 66

B. The Average Overall Execution Time over Different Cloud Environments…………… 66

5.5 The Proposed Auto Scaling Cloud Query Mesh (AS-CQM) Algorithm…………………………………………………….. 67

5.5.1 Running Example………………………………….. 70

5.6 Experimental Results and Analysis of the Proposed AS-CQM Algorithm…………………………………………………… 71

5.6.1 The Average Overall Execution Time…………………. 74

5.6.2 The Average CPU Utilization Percentages……………. 76

5.6.3 The Average Throughput……………………………… 77

5.6.4 The Cost Saving Percentage………………………….. 77

5.7 Discussion... 82

6- The Proposed Multiple Queries Optimization for Data Streams 84

on Cloud Environment

6.1 Introduction 85

6.2 The Proposed MQOP Framework.................................. 86

6.2.1 The Multiple Continuous Queries Optimization Sub-System………………………….. 87

6.2.2 Incoming Tuples Pre-Processing Sub-System………… 88

6.2.3 The Multiple Continuous Queries Execution on the Cloud Environment Sub-System……………………… 89

6.3 The Proposed Optimized Global Plan (OGP) Algorithm............ 89

6.4 Experimental Results and Analysis of the Proposed OGP Algorithm…………………... 91

6.4.1 The Optimization Time.. 93

6.4.2 The Average Execution Time………………………. 94

6.4.3 The Average Throughput... 95

6.5 The Proposed Optimized Global Plan based on the Partitioning (OGPP) Algorithm…………………………………………. 96

6.5.1 Running Example……………………………………… 98

6.6 Experimental Results and Analysis of the Proposed OGPP Algorithm…………………………………………………….. 99

6.6.1 The Optimization Time………………………………… 100

6.6.2 The Average Execution Time……………………… 102

6.6.3 The Average Throughput………………………… 104

6.7 Discussion ……………………………………………… 107

7- Conclusion and Future Work 109

Publications…....……………………………………………………. 114

References……………………………………………………… 115

Arabic Summary 126

List of Figures

Fig. 2.1	Example for stream processing over single plan and multiple plans	11
Fig. 4.1	The proposed OQMDS framework	33
Fig. 4.2	The proposed OII-QM algorithm	37
Fig. 4.3	The partitions lattice	38
Fig. 4.4	The proposed NS-QM algorithm	53
Fig. 4.5	The average optimization time of the II-QM, the proposed OII-QM and the proposed NS-QM algorithms	44
Fig. 4.6	The average execution time of the II-QM, the proposed OII-QM and the proposed NS-QM algorithms	45
Fig. 4.7	The average execution-runtime overhead of the II-QM, the proposed OII-QM and the proposed NS-QM algorithms	46
Fig. 4.8	The optimization memory usage of the II-QM, the proposed OII-QM and the proposed NS-QM algorithms	47
Fig. 5.1	The proposed CQOMP framework	52
Fig. 5.2	The proposed OMP algorithm	56
Fig. 5.3	The average overall execution time of the operator-set-cloud methodology and the proposed OMP algorithm	60
Fig. 5.4	The average throughput of the operator-set-cloud methodology and the proposed OMP algorithm	61
Fig. 5.5	The average overall execution time of the proposed OMP	62

algorithm and the NS-QM algorithm

Fig. 5.6 The average throughput of the proposed OMP and the NS-QM algorithms 63

Fig. 5.7 The average improvement percentages of execution times of the proposed OMP algorithm over different numbers of VMs 65

Fig. 5.8 The average overall execution time of the operator-set-cloud methodology and the proposed OMP algorithm on the amazon cloud environment 66

Fig. 5.9 The average overall execution time of the proposed OMP algorithm on different cloud environments 67

Fig. 5.10 The proposed AS-CQM algorithm 69

Fig. 5.11 The Average Overall Execution Time over the proposed elastic configurations provided by the proposed AS-CQM algorithm and the static configuration provided by the model view approach 75

Fig. 5.12 The average CPU utilization percentages over the proposed elastic configurations provided by the proposed AS-CQM algorithm and the static configuration provided by the model view approach 79

Fig. 5.13 The average throughput over the proposed elastic configurations provided by the proposed AS-CQM algorithm and the static configuration provided by the model view approach 80

Fig. 5.14 The cost saving percentage of the proposed elastic configurations provided by the proposed AS-CQM algorithm over the static configuration provided by the model view approach 81

Fig. 6.1 The proposed MQOP framework 87

Fig. 6.2 The proposed OGP algorithm 90

Fig. 6.3 The optimization time of the proposed OGP algorithm and the operator tree technique 94

Fig. 6.4 The average execution time of the proposed OGP algorithm and the operator tree technique 95

Fig. 6.5 The average throughput of the proposed OGP algorithm and the operator tree technique 96

Fig. 6.6 The proposed OGPP algorithm 97

Fig. 6.7 The optimization time of the proposed OGPP algorithm against the compile-time optimization method and the operator tree technique 102

Fig. 6.8 The average execution time of the proposed OGPP algorithm against the compile-time optimization method and the operator tree technique 103

Fig. 6.9 The average throughput of the proposed OGPP algorithm against the compile-time optimization method and the operator tree technique 105

List of Tables

Table 3.1 Comparison between QM, SQO, Cyclops and AMR 18

Table 3.2 Comparison between A-SEGO, FPGA, compile-time optimization, dynamic programming solution and the greedy algorithm, and operator tree 24

Table 3.3 Comparison between M3 and Map Update 26

Table 5.1 Specifications of the simulated Windows Azure and the real Amazon instances 58

Table 5.2 Amazon instances elastic specifications 73

List of Algorithms

The proposed OII-QM algorithm	36
The proposed NS-QM algorithm	40
The proposed OMP algorithm	54
The proposed AS-CQM algorithm	67
The proposed OGP algorithm	89
The proposed OGPP algorithm	96

List of Abbreviations

AMR	The adaptive multi-route query processing system
AS-CQM	The proposed Auto Scaling Cloud Query Mesh algorithm
A-SEGO	The Adaptive Sharing based Extended Greedy Optimization Approach
CANN query	The continuous aggregate nearest neighbor query
CEP	The Complex Event Processing framework
CQOMP	The proposed Continuous Query Optimization based on Multiple Plans framework for data streams over the cloud environment
CRNN query	The continuous range nearest neighbor query
DSPS	Distributed Stream Processing Systems
EM	The Expectation Maximization algorithm
IEVA	The Improved Event Based Algorithm
II-QM	The Iterative Improvement Query Mesh algorithm
kNN query	The k nearest neighbor query
M3	The Main-Memory MapReduce
MkNN query	The mobile k nearest neighbor query
MQOP	the proposed multiple queries optimization based on partitioning framework
NS-QM	The proposed Non-Search based Query Mesh algorithm
OGP	The proposed optimized global plan algorithm
OGPP	The proposed optimized global plan based on partitioning algorithm
OII-QM	The proposed Optimized Iterative Improvement Query Mesh algorithm
OMP	The proposed Optimized Multiple plans algorithm
OQMDS	The proposed optimized query mesh for data stream framework
PDAMS	The power and deadline-aware multicore scheduling algorithm
QM	The Query Mesh framework
RLD	Robust Load Distribution algorithm
SQO	The semantic query optimization approach
VGQ-Vor	The virtual grid quadtree with Voronoi diagram
Vk NN query	The visible k nearest neighbor query
VM	Virtual machine

Chapter 1

Introduction

Chapter 1 Introduction

1.1 Overview

Most of the recent applications such as sensor networks [1-4], traffic analysis applications [5], and tracking moving objects [6] generate a continuous, rapid, time varying and dynamic data element that are called data streams [7-12]. These applications present new challenges which are not handled by traditional techniques. Thus continuous queries are considered to process data streams which have a continuous and dynamic nature. Continuous queries are evaluated continuously with the continuous arrival of data streams over time [13-16].

Traditional database systems uses optimizers that process all data stream tuples based on a single query plan. This plan is not efficient with data streams that have continuous variations over time. For example, a continuous query with three operators (op1, op2, op3) and the query result is updated every five minutes for the latest data streams tuples over this interval. Because of the continuous nature of data streams over time, it is not efficient to process all tuples over all runs (each five minute) or even in the same run using one query plan. Consider that the best query plan to process the tuples within the first two minutes is (op1 - op2 – op3) but the best plan for the tuples in the next minute is (op2 - op3- op1) and so on. Therefore data streams processing using multiple query plans is a good alternative solution. Each query plan is the most suitable plan to process a cluster of tuples that have the nearest characteristics.