Master of Information Engineering

Universidad de Los Andes

Master's Thesis

Network Aware Elastic Key-Splitting in Distributed Stream Processing Systems

Diva Mercedes Martínez Laverde

Universidad de los Andes

Master of Information Engineering

Universidad de Los Andes

Master's Thesis

Network Aware Elastic Key-Splitting in Distributed Stream Processing Systems

Author:	Diva Mercedes Martínez Laverde
1st examiner:	ToFind
2nd examiner:	ToFind
Assistant advisor:	Nicolás Cardozo Álvarez Ph.D.
Submission Date:	June 2019

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

June 2019 Diva Mercedes Martínez Laverde

Acknowledgments

If someone helped you or supported you through your studies, this page is a good place to tell them how thankful you are.

"People sometimes ask me if it is a sin in the Church of Emacs to use vi. Using a free version of vi is not a sin; it is a penance. So happy hacking"

-Richard Stallman

Abstract

This document will serve as an example to you, of how to use LaTeXto write your CSE Master's Thesis. It will have examples and recomendations, and hopefully a few laughs. Because this is the abstract, it will have to convince you that this template is something you want to use. It has been proven, that without using this template, writing your thesis will be much more dificult. The template is based on previous work, and has been improved apon and updated. The result of this template is a modern latex template that everyone can contribute to and use for their studies of CSE @ TUM.

Some more great abstract tips can be found here: Great Abstract tips

Contents

1 Introduction

2 Preliminaries

Put down everything I think should be here. Order it, make subsections for each. Start filling the gaps. Pags: 3- 17 min

2.1 Paradigm Shift

Paradigm Shift[1]
As a result, the amount of data to be processed can be unbounded or never ending. At the same time, these applications need processing capabilities for continuously computing and aggregating incoming data for identifying interesting changes or patterns in a timely manner.

These applications are different from traditional DBMS applications with respect to data arrival rates, update frequency, processing requirements, Quality of Service (QoS) needs, and notification support

Queries that are processed by a traditional DBMS are termed ad hoc queries. They are (typically) specified, optimized, and evaluated once over a snapshot of a database. In contrast, queries in a stream processing environment are termed continuous queries

In addition, the snapshot approach for evaluating stream data may not always be appropriate as the values over an interval are important (e.g., temperature changes) for stream processing applications. Furthermore, the inability to specify quality of service requirements (such as latency or response time) to a DBMS makes its usage less acceptable for stream applications.

DBMSs were not designed to manage high-frequency updates (in the form of data streams) and to provide continuous computation and output for queries.

Applications[1]
i) to monitor traffic slowdown or accidents using data sent by each car on the road every few seconds or minutes, (ii) to perform program trading based on changes in the stock price of a particular stock relative to other stock prices using data from multiple feeds, and (iii) to monitor environmental and security applications for water quality, fire spread, etc. based on data received from sensors

Data Stream Characteristics[1]

Data Stream Application Characteristics[1]

Continuous Queries[1]

Window Specification[1]

QoS Metrics[1]

Data Stream Management System Architecture[1]

QoS-Related Challenges[1]

Capacity Planning and QoS Verification[1]

Scheduling Strategies for CQs[1]

Load Shedding and Run-Time Optimization[1]

Complex Event and Rule Processing[1]

Design and Implementation of a DSMS with CEP[1]

Stream Processing Model[2]

Physical Distribution of PEs[2]

Stream Processing Engine Requirements[2]

Fault Tolerance[2]

GAP Recovery[2]

Rollback Recovery[2]

Upstream Backup[2]

Precise Recovery[2]

Operator Graphs[3]

State in Operators[3]

Flavors of Parallelism[3]

Safety and Profitability[3]

3 Motivation

pags 19 - 28 mi

4 Related Work

Pags 23 - 28

5 Solution

Bibliography

[1] Sharma Chakravarthy. *Stream data processing : a quality of service perspective : modeling, scheduling, load shedding, and complex event processing*. Springer, New York, 2009.

[2] Supun Kamburugamuve, Geoffrey Fox, David Leake, and Judy Qiu. Survey of distributed stream processing for large stream sources. *Grids. Ucs. Indiana. Edu*, 2013.

[3] Scott Schneider, Martin Hirzel, and Buğra Gedik. Tutorial: stream processing optimizations. In *Proceedings of the 7th ACM international conference on Distributed event-based systems*, pages 249–258. ACM, 2013.