Incremental Techniques for Large-Scale Dynamic Query Processing

Iman Elghandour  
Université Libre de Bruxelles  
ieelghand@ulb.ac.be

Dan Olteanu  
University of Oxford  
dan.olteanu@cs.ox.ac.uk

Ahmet Kara  
University of Oxford  
ahmet.kara@cs.ox.ac.uk

Stijn Vansummeren  
Université Libre de Bruxelles  
svsummer@ulb.ac.be

ABSTRACT
Many applications from various disciplines are now required to analyze fast evolving big data in real time. Various approaches for incremental processing of queries have been proposed over the years. Traditional approaches rely on updating the results of a query when updates are streamed rather than re-computing these queries, and therefore, higher execution performance is expected. However, they do not perform well for large databases that are updated at high frequencies. Therefore, new algorithms and approaches have been proposed in the literature to address these challenges by, for instance, reducing the complexity of processing updates. Moreover, many of these algorithms are now leveraging distributed streaming platforms such as Spark Streaming and Flink. In this tutorial, we briefly discuss legacy approaches for incremental query processing, and then give an overview of the new challenges introduced due to processing big data streams. We then discuss in detail the recently proposed algorithms that address some of these challenges. We emphasize the characteristics and algorithmic analysis of various proposed approaches and conclude by discussing future research directions.

CCS CONCEPTS
• Theory of computation → Lower bounds and information complexity; Database query processing and optimization (theory); Distributed computing models;

KEYWORDS
Dynamic query processing; Incremental View Maintenance; Big Data; Data Streams;

ACM Reference Format:

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

1 INTRODUCTION

In a broad range of domains, such as Real Time Business Intelligence and Complex Event Processing, contemporary applications require the timely dynamic processing of complex analytical queries on continuously arriving data. Here, dynamic processing refers to updating the query result, preferably in real-time, when the underlying data is updated. Implementing such applications remains a difficult task, and involves resolving two orthogonal challenges:

• Designing a suitable dynamic query processing algorithm that determines how the application’s query results are to be updated upon data changes, taking into account that previous results are already available and re-computation should be avoided to ensure timeliness.

• Designing an implementation and deployment of the selected dynamic query processing algorithm that accounts for desiderata such as high throughput, low latency, and the ability to process large data sets. Current approaches mostly rely on distributed computing frameworks such as MapReduce, Flink, Spark, or Storm, to achieve this.

Fortunately, in recent years, there has been a flurry of research on both challenges that provide novel insights in how to resolve them. We briefly survey these next.

Algorithmical insights. Avoiding the re-computation whenever an update is received has long been approached using Incremental View Maintenance (IVM) techniques [6, 9]. IVM materializes the output of a query and then maintains that output under updates. Unfortunately, traditional IVM is not efficient for large databases that are updated at a high frequency. Therefore, new approaches have recently been proposed, whose objective is to reduce the complexity of processing updates and/or to reduce the required memory footprint.

Specifically, research in dynamic query processing has recently received a big boost with: (1) the introduction of Higher-Order IVM (HIVM) [13, 14, 19]; (2) the identification of lower bounds and worst-case optimal algorithms for processing updates [3, 4, 12]; (3) the practical formulations of worst-case-optimal IVM that implement and extend these algorithms [10, 11, 20]; and (4) the introduction of the notion of differential dataflow for computations that require recursive or iterative processing [16, 17]. These approaches often rely on materializing a succinct representation of a query’s output to maintain it more efficiently, and therefore present a fundamental breakthrough with traditional IVM techniques.
Big Data Frameworks support for dynamic query processing. Big data frameworks such as MapReduce [7] and Spark [24] are inherently batch-oriented. Early approaches for implementing dynamic query processing in these frameworks has focused on incremental processing of MapReduce tasks [5, 21, 23]. These are, however, based on traditional IVM techniques and suffer from high latency of MapReduce and its open source implementation Hadoop. More recent versions of distributed compute frameworks such as Apache Spark [25], Apache Flink [1], and Twitter Storm [22] / Heron [15] allow stream-based computations instead of batch-based computations. Out of the box, these frameworks mostly provide primitives for avoiding re-computation over sliding windows, based on traditional IVM. In addition, they present low-level programming primitives by which developers can express their own dynamic query processing algorithms. More recently, there are proposals to automatically incrementalize queries on distributed big data frameworks. Examples of these approaches include the distributed implementation of HIVM [18] and differential dataflow [17], as well as Spark Structured Streaming [2].

2 TUTORIAL STRUCTURE

The tutorial runs for 3 hours and is divided into the following four parts:

Part I: Introduction, desiderata, and traditional IVM

We start the tutorial by giving an introduction to dynamic query processing and show examples that motivate the need for efficient incremental query processing. We give a high-level historical overview of traditional approaches (known as First Order IVM) that have been employed by conventional database systems to maintain query outputs. We present the strong and weak points of traditional approaches and then discuss new challenges introduced by streaming large data at high frequencies.

Part II: Recent Algorithmic Advances in Dynamic Query Processing

In the second part of the tutorial, we survey new efficient approaches and algorithms for dynamic query processing. We discuss the following research works: (1) Higher-Order IVM [13, 14, 19]; (2) Complexity lower bounds for dynamic query processing [3, 4, 12]; (3) Dynamic Yannakakis [10, 11]; (4) Factorized IVM [20]; (5) Spacetime tradeoffs [12]; and (6) Beyond conjunctive queries: relations over application-dependent rings [8, 13, 20].

Part III: Dynamic Query Processing in Big Data Frameworks

Incremental processing of queries has been studied for queries executed by MapReduce [5, 21, 23] and by other distributed streaming platforms such as Spark Streaming [2, 25], Flink [1], and Storm [22] / Heron [15]. However, these systems rely on their users to specify how the queries are maintained or employ traditional incremental view maintenance approaches. Additionally, new parallel approaches that are executed in distributed environments [18] or that extend incremental processing [17] are introduced. We discuss all the mentioned approaches and platforms while highlighting the contributions that each one of them has made.

Part IV: Outlook

Finally we conclude by summarizing the existing research solutions and highlighting the open problems that are yet to be studied.

3 ACKNOWLEDGMENT

The authors are graciously supported by the Wiener-Anspach foundation. The work has received funding from the European Union’s Horizon 2020 research and innovation programme under grant agreement 682588.

REFERENCES