

# Sharing Aggregate Computation for Distributed Queries

Ryan Huebsch, Minos Garofalakis, Joseph M. Hellerstein, and Ion Stoica  
University of California at Berkeley, Intel Research Berkeley & Yahoo! Research

## ABSTRACT

An emerging challenge in modern distributed querying is to efficiently process multiple continuous aggregation queries simultaneously. Processing each query independently may be infeasible, so multi-query optimizations are critical for sharing work across queries. The challenge is to identify overlapping computations that may not be obvious in the queries themselves.

In this paper, we reveal new opportunities for sharing work in the context of distributed aggregation queries that vary in their selection predicates. We identify settings in which a large set of  $q$  such queries can be answered by executing  $k$   $q$  different queries. The  $k$  queries are revealed by analyzing a boolean matrix capturing the connection between data and the queries that they satisfy, in a manner akin to familiar techniques like Gaussian elimination. Indeed, we identify a class of *linear* aggregate functions (including SUM, COUNT and AVERAGE), and show that the sharing potential for such queries can be optimally recovered using standard matrix decompositions from computational linear algebra. For some other typical aggregation functions (including MIN and MAX) we find that optimal sharing maps to the NP-hard *set basis* problem. However, for those scenarios, we present a family of heuristic algorithms and demonstrate that they perform well for moderate-sized matrices. We also present a dynamic distributed system architecture to exploit sharing opportunities, and experimentally evaluate the benefits of our techniques via a novel, flexible random workload generator we develop for this setting.

**Categories and Subject Descriptors:** H.2.4 [Systems]: Distributed databases

**General Terms:** Algorithms, Design, Measurement

**Keywords:** Multi-query optimization, aggregation, linear algebra, duplicate insensitive

## 1. INTRODUCTION

There is a large and growing body of work on the design of distributed query systems. The focus of much of this work has been on the efficient execution of individual, one-shot queries, through intelligent data-processing algorithms, data/query shipping strate-

gies, etc. Recent years, however, have witnessed the emergence of a new class of *large-scale distributed monitoring* applications – including network-traffic monitors, sensornets, and financial data trackers – that pose novel data-management challenges. First, many monitoring tasks demand support for *continuous queries* instead of ad-hoc requests, to accurately track the current state of the environment being monitored. Second, given the inherently distributed nature of such systems, it is crucial to minimize the *communication overhead* that monitoring imposes on the underlying infrastructure, e.g., to limit the burden on the production network [5] or to maximize sensor battery life [16].

In most monitoring scenarios, the naive “warehousing solution” of simply collecting a large, distributed data set at a centralized site for query processing and result dissemination is prohibitively expensive in terms of both latency and communication cost – and often, simply unnecessary. The amount of data involved can be so large and dynamic in nature that it can easily overwhelm typical users or applications with too much detailed information. Instead, high-level, *continuous aggregation queries* are routinely employed to provide meaningful summary information on the underlying distributed data collection and, at the same time, to allow users to iteratively drill-down to interesting regions of the data. Typical aggregation queries also allow for effective, *in-network processing* that can drastically reduce communication overheads by “pushing” the aggregate function computation down to individual nodes in the network [14].

Another crucial requirement for large-scale distributed monitoring platforms is the ability to *scale* in both the volume of the underlying data streams and the number of simultaneous long-running queries. While there may be many queries in the network, they may have significant computational overlap either due to the intrinsic interest of certain streams, or due to “canned” query forms that keep queries narrowly focused. As an example, consider the Network Operations Center (NOC) for the IP-backbone network of a large ISP (such as Sprint or AT&T). Such NOCs routinely need to track (in real time) hundreds of continuous queries collecting aggregate statistics over thousands of network elements (routers, switches, links, etc.) and extremely high-rate event streams at different layers of the network infrastructure. This requirement emphasizes a new class of multi-query optimization problems that focus on dynamically sharing execution costs across continuous stream queries to optimize overall system performance.

**Our Contributions.** In this paper, we focus on dynamic multi-query optimization techniques for continuous aggregation queries over physically distributed data streams. In a nutshell, we demonstrate opportunities to compute  $q$  aggregation queries that vary in their selection predicates via  $k$   $q$  queries that may, in fact, be *different from the input queries themselves*. This surprising result

Permission to make digital or hard copies of all or part 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. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGMOD'07, June 12–14, 2007, Beijing, China.

Copyright 2007 ACM 978-1-59593-686-8/07/0006 ...\$5.00.























