Continuous Query-based Data Trading

Jin Cheng
School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
jincheng2@link.cuhk.edu.cn

Ningning Ding
Department of Electrical and Computer Engineering
Northwestern University
Evanston, USA
ningning.ding@northwestern.edu

John C.S. Lui
Department of Computer Science and Engineering
The Chinese University of Hong Kong
Hong Kong, China
cslui@cse.cuhk.edu.hk

Jianwei Huang*
School of Science and Engineering, Shenzhen Institute of Artificial Intelligence and Robotics for Society
The Chinese University of Hong Kong, Shenzhen
Shenzhen, China
jianweihuang@cuhk.edu.cn

ABSTRACT

In the era of big data, traditional data trading methods designed for one-time queries on static databases fail to meet the demands of continuous query-based trading on streaming data, often resulting in repeated and inaccurate charges due to neglecting computation sharing in continuous query execution. To address this, we introduce CQTrade, the first trading framework tailored for continuous queries, which incorporates computation sharing across different time windows and queries, enhancing integration with existing trading systems. Our contributions include a theoretical analysis of computation-sharing techniques, the development of a general optimization problem to maximize seller revenue adaptable to various techniques, and the proposal of a branch-and-price algorithm to handle the problem's complexity. Our evaluation shows that the proposed framework improves the success rate of data trading by 12.8% and boosts the seller's revenue by an average of 28.7%, compared to the one-time query-based data trading methods used in the current data market.

ACM Reference Format:
Jin Cheng, Ningning Ding, John C.S. Lui, and Jianwei Huang. 2024. Continuous Query-based Data Trading. In *Abstracts of the 2024 ACM SIGMETRICS/IFIP PERFORMANCE Joint International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS/PERFORMANCE Abstracts '24), June 10–14, 2024, Venice, Italy*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3652963.3655050

1 MOTIVATION

Data has increasingly become a vital asset in data-driven decision-making processes such as machine learning, highlighting the need for robust data trading platforms [3]. The surge in online analytics and online machine learning calls for a shift in data trading methods from traditional one-time query-based to continuous query-based. However, current data trading practices primarily cater to static, one-time queries. If applied directly to continuous queries, existing methods often lead to repeated and imprecise charges that hurt the seller's revenue. This is because existing methods do not consider the computation sharing across different time windows and among different queries, which is crucial for continuous query execution.

To meet the evolving needs of online analytics, it is crucial to transition from traditional static one-time data trading methods to those that support continuous queries (CQs) [1]. CQs continuously collect and update data within specified time windows, enhancing real-time decision-making. In practice, computation sharing boosts the efficiency of CQ execution by allowing computation sharing across different time windows and queries. To fully leverage this capability, an essential step is to develop a CQ-based data trading framework that incorporates computation sharing, optimizing data utilization, and maximizing economic returns.

2 CQTRADE: OVERVIEW

We begin by analyzing computation sharing techniques for continuous queries. From this analysis, we develop CQTrade, a trading framework integrated with a data stream management system. We then formulate a general optimization problem to maximize seller revenue. To address the complexity of the problem, we introduce a branch-and-price algorithm.

2.1 Continuous Query

Customers submit continuous queries (CQs) to the seller with a data stream management system (DSMS) based on their needs.

Definition 2.1. *(Continuous Query) A continuous query $q = (s, r, \pi)$ aggregates streaming data over time windows, each with a specified time span termed as 'range' (r), and updates its results at every 'slide' (s) interval. The query execution involves a 'payment' (π) to the seller per time unit.*

We investigate computation sharing among CQs in multiple CQ execution techniques, a critical aspect for continuous query-based data trading [2]. We introduce and analyze self-sharing, which optimizes data processing within the same query across different

*Jianwei Huang is the corresponding author of this paper.

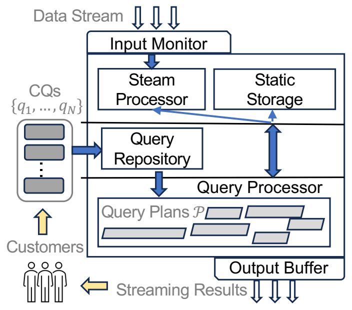

Figure 1: The trading system, CQTrade.

time windows, and mutual-sharing, which extends to sharing computations among different queries. From our theoretical analysis, two key observations emerged: First, self-sharing does not always guarantee revenue maximization, especially when the payment is low, highlighting the constraints on the maximum range and its implications on profitability. Second, mutual-sharing often proves beneficial, especially when individual CQs are unprofitable on their own. By executing these together, mutual-sharing can enhance both the success rate of data trading and the overall revenue, providing a compelling case for its implementation in continuous query-based trading systems.

2.2 System Design

Fig. 1 depicts the trading framework CQTrade, integrated with a data stream management system. Customers submit CQs within the query set $Q = \{q_1, \cdots, q_N\}$ of different window specifications to the query repository. The query processor executes CQs according to the query plans (defined in Definition 2.2) outlined in the trading strategy, interacting with the stream processor and the storage for processing the stream and storing results.

Definition 2.2. *(Query Plan) A query plan, denoted as $p \subseteq Q$, represents a continuous query set executed in one container, with a computation sharing type $t \in \mathcal{T}$, indicating how to share computation.*

A query plan organizes multiple CQs that the system executes collectively, leveraging computation sharing to optimize operational efficiency. The computation sharing types for plans $\mathcal{T} = \{T_{sm}, T_{sn}, T_{nn}\}$ include T_{sm} for self-sharing and mutual-sharing, T_{sn} for just self-sharing, and T_{nn} for no sharing. Only CQs within a plan can share computations, leading to diverse potential execution and computation-sharing patterns. The cost structure of each plan is a critical component of our trading framework. Through our analysis, we recognize that it comprises two main parts: the *startup cost*, which is shared among all CQs within a plan, and the *query-specific costs*, which are individual to each CQ. The startup cost includes expenses related to setting up the plan, which are incurred once per plan activation. In contrast, query-specific costs are calculated per CQ based on the processing and memory resources they consume during their execution. The dual cost structure identifies a common cost framework for different techniques, allowing us to create a general optimization problem.

2.3 Optimization Problem

$$\begin{aligned}
\max_{\boldsymbol{x},\boldsymbol{y}} \quad & \sum\nolimits_{i\in[K]} \sum\nolimits_{j\in[N]} x_{ij} a_{ij} - \sum\nolimits_{i\in[K]} C_i y_i \\
s.t. \quad & \sum\nolimits_{j\in[N]} x_{ij} \boldsymbol{w}_{ij} \le \boldsymbol{W}_i y_i, \text{ for } i \in [K], \\
& \sum\nolimits_{i\in[K]} x_{ij} = 1, \text{ for } j \in [N], \\
& x_{ij} \in \{0,1\}, \text{ for } i \in [K], j \in [N], \\
& y_i \in \{0,1\}, \text{ for } i \in [K],
\end{aligned} \qquad (\mathbb{P}_1)$$

According to the common cost structure, we formulate a general optimization problem $\mathbb{P}_1$ to maximize seller revenue. The problem incorporates: binary variables y_i for starting up plan p_i and x_{ij} for assigning CQ q_j to plan p_i, with K and N denoting the total numbers of plans and CQs. Parameters a_{ij}, C_i, $\boldsymbol{w}_{ij}$, and $\boldsymbol{W}_i$ represent the net profit of CQ allocation, plan startup cost, specific allocation cost, and plan capacity, respectively. The complexity of the problem stems from combining aspects of knapsack and bin packing problems.

To address the complexity, we propose a branch-and-price algorithm adapted to the problem, achieving global optimality. This solution stands out due to its ability to decompose the problem into manageable sub-problems, which are then solved iteratively. The algorithm's design allows it to adapt dynamically to changes in query characteristics and system demands, ensuring an optimal configuration of query plans that significantly enhances the trading system's efficiency and profitability.

3 EXPERIMENTAL RESULTS

We conduct a comprehensive evaluation of our proposed framework, focusing on its impact on the success rate and the revenue it generates for sellers. Our experiments demonstrate that the trading framework improves the success rate of data trading by 12.8% and boosts the seller's revenue by an average of 28.7%.

We also found that precise adjustments in the query plan settings can significantly influence the success rate and profitability of data trading. Specifically, plans tailored to the dynamic requirements of the data market allow for optimized computation resource usage and cost management. This fine-tuning leads to not only higher revenue margins but also to a more stable trading environment.

ACKNOWLEDGMENTS

This work is supported by the National Natural Science Foundation of China (Project 62271434), Shenzhen Science and Technology Innovation Program (Project JCYJ20210324120011032), Guangdong Basic and Applied Basic Research Foundation (Project 2021B1515120008), Shenzhen Key Lab of Crowd Intelligence Empowered Low-Carbon Energy Network (No. ZDSYS20220606100601002), Shenzhen Stability Science Program 2023, and the Shenzhen Institute of Artificial Intelligence and Robotics for Society.

REFERENCES

[1] Arvind Arasu, Shivnath Babu, and Jennifer Widom. 2006. The CQL continuous query language: semantic foundations and query execution. *The VLDB Journal* 15 (2006), 121–142.
[2] Jin Cheng, Ningning Ding, John C.S. Lui, and Jianwei Huang. 2024. *Continuous Query-based Data Trading.* https://www.dropbox.com/scl/fi/ey2erxx1yr42duint8hhq/CQTrade.pdf?rlkey=srjis9c88b8ll294ua2yozpt1&dl=0
[3] Zicun Cong, Xuan Luo, Jian Pei, Feida Zhu, and Yong Zhang. 2022. Data pricing in machine learning pipelines. *Knowledge and Information Systems* 64, 6 (2022), 1417–1455.