VG-Prefetcher Cache: Towards Edge-Based Time Series Data Management Using Visibility Graph Prefetching

Akram Bensalem, Laurent d'Orazio, Julien Lallet, Andrea Enrici

VG-Prefetcher Cache: Towards Edge-Based Time Series Data Management Using Visibility Graph Prefetching

Akram Bensalem*
akram.bensalem@imt-atlantique.fr
Plouzané, France

Laurent d'Orazio
laurent.dorazio@irisa.fr
Lannion, France

Andrea Enrici
andrea.enrici@nokia-bell-labs.com
Massy, France

Julien Lallet
julien.lallet@nokia-bell-labs.com
Lannion, France

ABSTRACT

The demand for efficient and reliable cloud computing systems is increasing. However, effectively managing data workloads in edge cloud systems, especially for connected cars, can be challenging. To address this issue, we have developed a new cache management technique named VG-Prefetcher Cache that uses visibility graphs to handle time series data more effectively. Our approach involves predicting future data and prefetching it into the cache, which reduces retrieval time and improves system performance. VG-Prefetcher Cache presents a promising approach for overcoming challenges in managing data workloads, thus paving the way for a more efficient and reliable cloud computing system.

CCS CONCEPTS

• **Computer systems organization** → **Real-time system architecture**; *Cloud computing*; • **Information systems** → *Key-value stores.*

KEYWORDS

Data Management, Time Series, Visibility Graph, Prefetching

ACM Reference Format:
Akram Bensalem, Laurent d'Orazio, Andrea Enrici, and Julien Lallet. 2024. VG-Prefetcher Cache: Towards Edge-Based Time Series Data Management Using Visibility Graph Prefetching. In *36th International Conference on Scientific and Statistical Database Management (SSDBM 2024), July 10–12, 2024, Rennes, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3676288.3676304

1 INTRODUCTION

The transportation industry has recently experienced a significant transformation with the emergence of Intelligent Transportation Systems (ITS), replacing traditional systems [25]. This is mainly due to the development of wireless communication technologies, which have enabled vehicles to become more connected and intelligent[11, 14]. However, with the increasing demand for heavy applications and services, challenges arise as we move towards an autonomous future. The growing number of compute-intensive applications requires more resource capacity, leading to increased energy consumption and decreased autonomy for connected vehicles [17]. This is a significant obstacle to developing connected vehicles and ITS. We can look forward to a more intelligent, efficient, and reliable transportation system, improving the safety of connected cars, robots, and other vehicles [15]. Connected and autonomous cars generate a significant amount of data that needs to be processed in real-time to ensure the smooth running of the vehicles [13, 26].
To handle the continuous flow of data, various technologies such as edge computing [8, 10], cloud computing [23], and big data processing are utilized [19, 22]. Edge computing processes the data sensors capture at the edge, which is then transmitted to the cloud for further analysis. The cloud provides the necessary scalability and flexibility to store and manage the vast data generated [20]. While VG-Prefetcher Cache primarily targets the ITS industry, it is a versatile solution designed to cater to any real-time system that relies on real-time time series data. Its adaptability extends to fields such as medicine, robotics, and more, promising many applications and benefits.

Ensuring system reliability is a critical task, and one effective solution is to load data into cache memory in the Edge Cloud, which is near clients. This cache memory stores frequently accessed data, significantly reducing the time required to transfer extensive data. The selection of data to be loaded into cache memory quickly and the location and design of the cache memory are crucial aspects to consider.We aim to revolutionise the performance of connected cars and similar systems by reducing response latency to client queries and increasing overall throughput, explicitly focusing on time series databases. We are venturing into a quick and efficient non-traditional forecasting method to predict client queries as fast as possible to achieve this. Unlike traditional AI methods, this innovative approach holds great promise, sparking curiosity and excitement.

To address these, we propose a novel solution called VG-Prefetcher Cache, an in-memory time series cache management based on Visibility Graph techniques for its adaptive prefetcher.

In this paper, We will review the latest research in the field and then develop a preliminary concept before diving into our solution. This will involve considering global design and architecture and the complexities of caching management and prefetcher functions. Finally, we will conclude by exploring opportunities for further advancements.

*All authors contributed equally to this research.

SSDBM 2024, July 10–12, 2024, Rennes, France

ACM ISBN 979-8-4007-1020-9/24/07
https://doi.org/10.1145/3676288.3676304

2 RELATED WORK

Time series data caching has become increasingly popular in recent years. This paper will discuss the relevant works most applicable to our research. Most previous works have focused on handling time series data in databases created for various purposes and with different focus areas.

The most well-known in-memory solution is Memcached[1]. It is a distributed caching system introduced by Fitzpatrick in 2004 to improve the performance of web applications by reducing the load on databases [9]. Its features include its distributed nature, key-value store approach and in-memory storage.

Meanwhile, it was not designed explicitly for time series. Pelkonen et al. (2015) introduced Gorilla[21] as a high-performance in-memory database for storing large amounts of time-series data. The database employs a unique compression algorithm to maximise memory efficiency and minimise overhead. With its in-memory architecture, Gorilla offers significantly reduced query latencies.

Recent research on semantic caches has revealed their varied applications in different domains[24]. BSCache[28] is a specific semantic caching scheme that caters to the time-series data that Zhang, Wang, and Shao created. It introduces a novel caching mechanism that utilises a semantic approach to account for unique access patterns and update frequencies. This mechanism organises cached data into semantic layers and prioritises them based on relevance, significantly improving query performance. BSCache[28] can also adjust its caching strategy dynamically to optimise utilisation for evolving workloads.

TSCache[16] is a caching time-series data scheme specifically designed for time-series data by leveraging the benefits of flash memory. TSCache[16] efficiently separates hot and cold data, significantly minimising write operations, and employs algorithms that adapt to real-time data access patterns.

Unlike BSCache, TSCache[16] does not directly interact with time-series databases. Instead, it adopts a time-aware mechanism to distinguish hot data, keeping it in the cache longer. Additionally, it utilises a unique two-layered indexing scheme to keep track of queries and data in the cache efficiently. In contrast, BSCache[28] relies on the semantics of time series for cache management.

3 PRELIMINARY

Time series Queries. Time series refers to sets of observations or measurements taken at specific time intervals [2, 3], usually at uniform intervals. The nature and properties of time series queries distinguish it from other data types. and it have unique properties:

- Write-Once and Append-Only
- Dominance of Range Queries
- Unique and Highly Skewed Access Pattern

Edge Caching. Caching is a well-known technique for improving the performance of distributed systems and thereby reducing the latency of data retrieval [5].

Prefetching. Prefetching refers to predicting future requests for data before the actual requests are made and proactively fetching the data from a remote storage location to a local cache in anticipation of the predicted requests [6, 27]. This method aims to enhance the performance of distributed systems by reducing the latency of data retrieval and improving the responsiveness of applications. In edge cloud computing, prefetching can be applied to store frequently accessed data, such as sensor data and other content, at the edge servers (ESs).

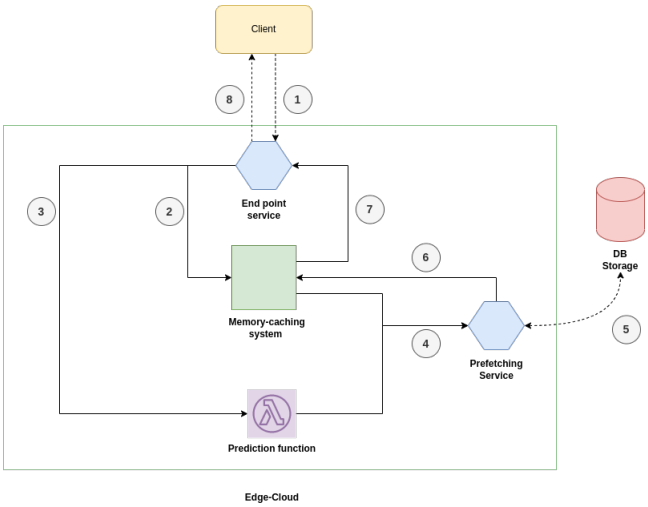

Figure 1: Globale Architecture overview

4 VG-PREFETCHER CACHE

4.1 Design and Architecture

As presented in Figure 1, our proposal system design utilises a time-series system cache mechanism. The client's end-point service will communicate directly with the memory cache instead of the Remote Database. This change will not affect the platform's behaviour; the result will remain the same. The service will first access data from the cache memory. Suppose it does not exist in the cache memory. In that case, another service will handle the situation by retrieving the data from remote sources in the cloud and uploading it into the memory cache. Simultaneously, several serverless functions will run in parallel to predict the future required data by analysing previous queries.

According to Figure 1, eight steps integrate the cache mechanism with prefetching capability in the edge cloud; the client sends queries to the cloud. The edge cloud endpoint handles the request by accessing the local cache memory. Simultaneously with step two, the edge cloud launches a serverless function to predict future data based on previous requests for 5. and 6. In case of any missing values of queries from the client or predicted queries in the cache memory, another service will handle the situation by uploading the missing data from the data source. The data is sent to the designated endpoint for the edge cloud. Then, It streams the response directly to the final destination.

4.2 Cache Management

It is essential to have data management strategies that are both efficient and customised for different data types, particularly time-sensitive data. Leveraging techniques from popular caching systems such as TSCache [16] and BSCache [28], our architecture aims to provide efficient in-memory caching for time series data.
VG-Prefetcher Cache is designed to address the complexities of edge computing. It is adapted from TSCache [16]. This allows it to effectively manage caching in distributed systems, ensuring that frequently accessed data is available to edge devices while minimizing network traffic and latency.
A In-memory cache system as Memcache [1] has been updated to include new and essential features that make it work better.

- **Time-range based Interface:** The time-range-based interface has been designed to allow users to retrieve data within specific intervals. where queries often require data from specific time ranges.
- **Handling Partial Hits:** Instead of retrieving all requested data from the remote database when it is not in the cache, our system identifies that part of data that already existed in cache memory and retrieves whatever data is available in the cache first. It then proceeds to obtain the remaining data from the remote database.
- **Time-aware Caching Policy:** We propose to have a time-aware policy that considers the temporal nature of time-series data. It prefers recent data, which is more relevant in time-series scenarios, and evicts older data. It ensures users get the most pertinent and timely data from the cache.

4.3 Visibility Graph Prefetcher

Data prefetching involves predicting the data that will be required in the future and preloading it into the cache of the edge device. This approach helps reduce latency and improve the overall performance of edge computing systems. With the increasing demand for real-time data processing and analysis. We will explore our proposal to predict data to load in cache memory before being requested.

4.3.1 Prediction Function. We recommend using a serverless approach. Prefetching future query predictions can also reduce the delay in loading a large amount of data from the data source.
Considering various factors, this function utilises an adaptive prefetcher that identifies coming data to be loaded in the memory cache.

Adaptive Prefetcher. When a client submits a request for a read query, the system's specific service performs two simultaneous tasks, as shown in figure 2. The system's specific service is designed to handle many read queries without compromising the system's performance. This means the query results are delivered quickly, even if multiple queries are processed simultaneously.
The first process is to predict the following query, and the prediction function will predict both the time range and any metadata of the query. By analysing the patterns in the start time, end time, and metadata tags, we can determine the necessary information for the query. This process requires various algorithms to predict time series data and determine the most probable outcome.
The cache memory system is designed to hold frequently accessed data, which means that the more often a query is executed, the higher the likelihood that the response data will be found in the cache memory.
To optimize performance, the service will execute the upcoming queries before the client asks for them. This allows part of the following data to be in the cache memory.

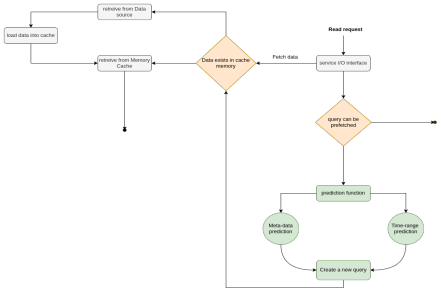

Figure 2: Adaptive prefetche

Query predictability. Certain conditions must be met to predict a client's query accurately. As shown in figure 3, Specific criteria must be met to determine if a query can be repeated. Firstly, we require a sufficient number of previous queries of the exact nature to forecast the current query. we also need to ensure that the values of the tags and fields used in past queries are repeatable. By checking these two conditions, we can determine if a query is repeatable.

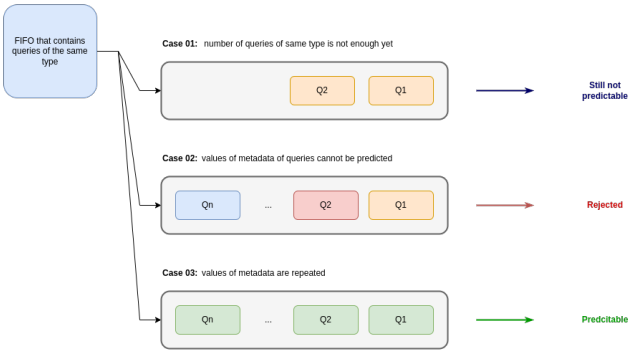

Figure 3: The Cases of Queries that can be predictable

4.3.2 Prediction of Time Range for Query. As shown in figure 4. The proposed method is designed to achieve accurate and efficient predictions. The process is divided into different phases, each carefully designed to produce the desired outcome.
Phase 1 focuses on collecting a list of queries with the same key and placing them in a First-In-First-Out (FIFO) queue for quick predictions. **Phase 2** involves extracting two sets of time series for further examination, providing valuable information about the data's behaviour over time. **In phase 3**, data is transformed into a complex network graph using the Visibility Graph approach to predict future time series data. **Phase 4** uses the visibility graph to predict future values. It finds similarities between nodes and uses these similarities as factors to make accurate predictions.

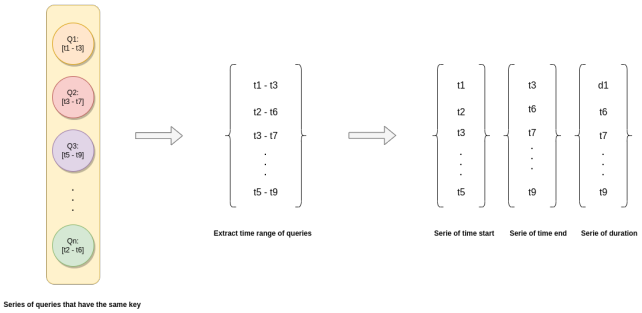

Figure 4: Extract time start, time end, and duration series from a list of queries that have the same key

5 DISCUSSION

The proposed method optimises edge cloud-based time series systems by integrating traditional caching methods with predictive prefetching capabilities. This enables seamless and faster data access, reducing latency and optimising bandwidth utilisation, particularly during high-traffic scenarios. Our solution is tailored for edge cloud environments and is highly effective in time series database applications such as ITS, making it adaptable to various situations, especially when handling massive real-time datasets. This system efficiently manages large amounts of data and users, adapting to real-time incoming data to ensure a smooth user experience.

6 CONCLUSION

This paper introduces VG-Prefetcher Cache which is a new approach to managing time series data on the edge cloud by developing an advanced version of an in-memory caching tool well-suited for real-time applications such as connected cars. This system will predict the required data based on Visibility Graph techniques, making the process faster and more efficient. This reduces waiting time and enhances system performance, especially under heavy usage. As technology develops, we can improve our in-memory caching solution and adaptive prefetcher function by incorporating FPGA accelerators[4, 7, 12]. By leveraging Field-Programmable Gate Arrays (FPGA), we can significantly enhance the performance of VG-Prefetcher Cache, particularly in reducing latency incurred during memory accesses and predicting queries with reduced latency [18]. Using FPGA accelerators can bypass the latencies of traditional CPU-based approaches and achieve a performance boost, leading to significant advancements in performance optimisation and setting a new standard in the field of in-memory database solutions.

ACKNOWLEDGMENTS

The research presented in this paper was supported by Nokia Bell Labs France in collaboration with IRISA Laboratory.

REFERENCES

[1] [n. d.]. Memcached. https://github.com/memcached/memcached. Accessed: 2023-09-09.
[2] [n. d.]. What Is a time series? https://www.investopedia.com/terms/t/timeseries.asp. Accessed: 2023-09-09.
[3] [n. d.]. What is time series data? https://www.influxdata.com/what-is-time-series-data/. Accessed: 2023-09-09.
[4] Thomas Bollaert. 2018. Fundamentals of FPGA-based Acceleration. https://www.xilinx.com/publications/events/developer-forum/2018-frankfurt/fundamentals-of-fpga-based-acceleration.pdf Accessed: 2023-09-10.
[5] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. 2012. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*. 13–16.
[6] Surendra Byna, Yong Chen, and Xian-He Sun. 2008. A taxonomy of data prefetching mechanisms. In *2008 International Symposium on Parallel Architectures, Algorithms, and Networks (i-span 2008)*. IEEE, 19–24.
[7] Andrew Canis and Ruolong Lian. 2018. Accelerating Memcached on Cloud FPGAs. https://www.xilinx.com/publications/events/developer-forum/2018-frankfurt/accelerating-memcached-on-cloud-fpgas.pdf Accessed: 2023-09-10.
[8] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. 2020. An overview on edge computing research. *IEEE access* 8 (2020), 85714–85728.
[9] Brad Fitzpatrick. 2004. Distributed caching with memcached. *Linux journal* 2004, 124 (2004), 5.
[10] Fabio Giust, Vincenzo Sciancalepore, Dario Sabella, Miltiades C Filippou, Simone Mangiante, Walter Featherstone, and Daniele Munaretto. 2018. Multi-access edge computing: The driver behind the wheel of 5G-connected cars. *IEEE Communications Standards Magazine* 2, 3 (2018), 66–73.
[11] Jianhua He, Kun Yang, and Hsiao-Hwa Chen. 2020. 6G cellular networks and connected autonomous vehicles. *IEEE Network* 35, 4 (2020), 255–261.
[12] Kevin D Hsiue. 2014. *FPGA-based hardware acceleration for a key-value store database*. Ph. D. Dissertation. Massachusetts Institute of Technology.
[13] Rasheed Hussain and Sherali Zeadally. 2018. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys & Tutorials* 21, 2 (2018), 1275–1313.
[14] Damigou Kombate et al. 2016. The Internet of vehicles based on 5G communications. In *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 445–448.
[15] Yangxin Lin, Ping Wang, and Meng Ma. 2017. Intelligent transportation system (ITS): Concept, challenge and opportunity. In *2017 ieee 3rd international conference on big data security on cloud (bigdatasecurity), ieee international conference on high performance and smart computing (hpsc), and ieee international conference on intelligent data and security (ids)*. IEEE, 167–172.
[16] Jian Liu, Kefei Wang, and Feng Chen. 2021. TSCache: an efficient flash-based caching scheme for time-series data workloads. *Proceedings of the VLDB Endowment* 14, 13 (2021), 3253–3266.
[17] Shaoshan Liu, Liangkai Liu, Jie Tang, Bo Yu, Yifan Wang, and Weisong Shi. 2019. Edge computing for autonomous driving: Opportunities and challenges. *Proc. IEEE* 107, 8 (2019), 1697–1716.
[18] Fabio Maschi, Dario Korolija, and Gustavo Alonso. 2023. Serverless FPGA: Work-In-Progress. In *Proceedings of the 1st Workshop on SErverless Systems, Applications and MEthodologies*. 1–4.
[19] Bob McQueen. 2017. *Big data analytics for connected vehicles and smart cities*. Artech House.
[20] Pavan Muralidhara. 2017. IoT applications in cloud computing for smart devices. *INTERNATIONAL JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY* 1, 1 (2017), 1–41.
[21] Tuomas Pelkonen, Scott Franklin, Justin Teller, Paul Cavallaro, Qi Huang, Justin Meza, and Kaushik Veeraraghavan. 2015. Gorilla: A fast, scalable, in-memory time series database. *Proceedings of the VLDB Endowment* 8, 12 (2015), 1816–1827.
[22] Christian Prehofer and Shafqat Mehmood. 2020. Big data architectures for vehicle data analysis. In *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 3404–3412.
[23] Ling Qian, Zhiguo Luo, Yujian Du, and Leitao Guo. 2009. Cloud computing: An overview. In *Cloud Computing: First International Conference, CloudCom 2009, Beijing, China, December 1-4, 2009. Proceedings 1*. Springer, 626–631.
[24] Qun Ren, Margaret H Dunham, and Vijay Kumar. 2003. Semantic caching and query processing. *IEEE transactions on knowledge and data engineering* 15, 1 (2003), 192–210.
[25] Akhtar All Shah and Lee Jong Dal. 2007. Intelligent transportation systems in transitional and developing countries. *IEEE Aerospace and Electronic Systems Magazine* 22, 8 (2007), 27–33.
[26] Aditi Tiwari and KB Akhilesh. 2020. Exploring connected cars. *Smart Technologies: Scope and Applications* (2020), 305–315.
[27] Steven P Vander Wiel and David J Lilja. 1997. When caches aren't enough: Data prefetching techniques. *Computer* 30, 7 (1997), 23–30.
[28] Kai Zhang, Zhiqi Wang, and Zili Shao. 2022. BSCache: A Brisk Semantic Caching Scheme for Cloud-based Performance Monitoring Timeseries Systems. In *Proceedings of the 51st International Conference on Parallel Processing*. 1–10.