Journal of Engineering and Sustainable Development
Vol. 28, No. 04, July 2024
https://jeasd.uomustansiriyah.edu.iq/index.php/jeasd
ISSN 2520-0917
Research Article
https://doi.org/10.31272/jeasd.28.4.5

Enhancing Big Data Performance Through Graph Coloring-Based Locality of Reference

Methaq Kadhum[1*], Mohammad Malkawi[2], Enas Rawashdeh[3]

[1]Electrical Engineering Department, College of Engineering. Mustansiriyah University, Baghdad, Iraq
[2]Software Engineering Department, Information and Computing College, Jordan University of Science and Technology Irbid, Jordan
[3]Management information system, Amman College, AlBalqa Applied University, Amman, Jordan

[1]https://orcid.org/0009-0003-3096-3000
[2]https://orcid.org/0000-0003-0109-8196
[3]https://orcid.org/0000-0002-4810-543X
[*]Email: methaq@uomustansiriyah.edu.iq

Article Info

Received	19/09/2023
Revised	14/04/2024
Accepted	21/04/2024

Abstract

Efficiency is a crucial factor when handling the retrieval and storage of data from vast amounts of records in a Big Data repository. These systems require a subset of data that can be accommodated within the combined physical memory of a cluster of servers. It becomes impractical to analyze all of the data if its size exceeds the available memory capacity. Retrieving data from virtual storage, primarily hard disks, is significantly slower compared to accessing data from main memory, resulting in increased access time and diminished performance. To address this, a proposed model aims to enhance performance by identifying the most suitable data locality structure within a big data set and reorganizing the data schema accordingly; by locality, it has been referred to as a particular access pattern. This allows transactions to be executed on data residing in the fastest memory layer, such as cache, main memory, or disk cache.

Keywords: Big Data; Graph Representation; Locality of References; Performance; Synthetic Memory

1. Introduction

Recently, massive collections of large datasets, often referred to as big data, changed the imperatives of data management across industries[1]-[4]. This paradigm shift redefined how information is used and used, becoming the standard method in science, technology, education, health, and safety. These cost reductions have democratized access to data storage and improved computing capabilities, enabling organizations of all sizes to engage in comprehensive data-driven efforts for the purpose, Web forms like Facebook attest to a relentless diversity of data text, images, and videos that contribute to an ever-expanding repository of information that is constantly processed and exchanged, and highlights the real-time nature of modern data in the 19th century [5]-[6]. Fig. 1 [1] shows how gigabyte prices have changed over the last 30 years. According to HIS market analysis[1], more than 1 billion surveillance cameras are installed worldwide as of 2021, which stream petabytes worth of data continuously

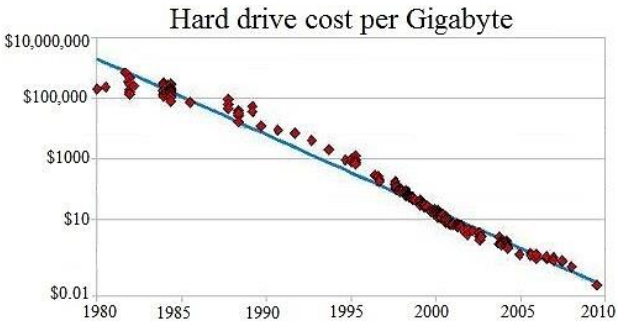

Figure. 1: hardware cost over the last 30 years **[1]**

Certainly, the collected data, whether in health, social networks, or security is extremely valuable for all types of purposes. However, the usefulness of data is as good as its readiness and availability at the time when it is needed for

processing, rather than its being there at some storage network.

Researchers have made significant strides in enhancing the different components of big data through a variety of techniques. These include efficient indexing, caching, and filtering [7]-[8] as well as advancements in query execution plans and effective data partitioning [9]-[10]. These approaches are designed to optimize various aspects of big data processing and yield improved overall performance. By leveraging these techniques, researchers and practitioners can achieve enhanced data analysis, more efficient data retrieval, and meaningful insights extraction from large-scale datasets [11]-[12]. These advancements, combined with the integration of cutting-edge technologies like text mining [13]-[14] and machine learning [15]-[16], have revolutionized the field of big data and enabled informed decision-making across diverse domains. However, one critical challenge faced in big data processing is the performance degradation caused by memory constraints. Memory constraints arise when the amount of memory required to store the data being referenced is significantly smaller than the overall size of the dataset. For instance, a database with a size of 100 GB may not fit entirely into a computer system with only 16 GB of RAM. Accessing data from the database involves transferring it from massive storage, such as networked disk space, to the computer's RAM at the time of data retrieval. As the program requires more data, the system is forced to swap data between the memory and the disk, resulting in expensive data transfer operations. These operations introduce delays as the CPU waits for the new data to be loaded into memory.

To address these challenges, our proposed solution leverages the concept of the locality of references. Locality of references refers to the tendency of a processor to access the same set of memory locations repeatedly over a short period [9],[17]. One key characteristic that reduces CPU waiting time and data transfer between memory and disk is the quality of special and temporal [18]-[19] locality of reference. This means that once data is loaded into memory, it remains there for a relatively long time before being sent back to the disk. Furthermore, data that is removed from memory is unlikely to be reloaded shortly. By reducing data transfer between memory and disk and minimizing CPU waiting time, the performance of big data processing can be significantly enhanced [20].

The main contribution of our proposed method is its focus on improving the performance of big data through ontology analysis to discover the locality of reference within the dataset. We will construct a synthetic memory reference trace based on the behavior of database queries. This trace will capture the temporal and spatial relations between data references. To represent these relations effectively, we will utilize graph representation techniques.

The paper is organized as follows: Section 2 discusses a review of methods to improve big data performance using the locality of references from previous related works. Section 3 presents a detailed description of the proposed method. Finally, Section 4 presents the conclusions drawn from this study and makes recommendations for possible future research directions.

2. Literature Review

Several prior works have proposed different approaches to improve the performance of big data, generally, these approaches have been proposed falling into two main categories: system-level optimizations and reference locality exploitation. Our work builds upon these two approaches, for improving the performance of big data processing systems by enhancing system-level optimizations and exploiting reference locality more effectively.

2.1. Methods That Improve Big Data Performance

The most successful approach to improving the performance of big data is to store the primary dataset in the main memory of the computer instead of relying on hard disks for storage It is called memory database systems [21], this approach takes advantage of the speed advantage of memory storage. The concept of performance in memory (PIM) has received considerable attention in previous research, both in previous work such as [22]-[24] and more recent studies such as (e.g., [25]-[27], respectively. PIM revolves around performing computations in front of data, reducing the limitations imposed by off-chip bandwidth limitations. This strategy aims to increase the performance of the computer system in terms of performance and energy consumption. However, with big data, simply storing the entire dataset in memory is not practical due to its enormous size. As a result, methods for optimizing primary memory and disk storage emerged. Researchers have introduced various approaches ([28]- [30]) aiming to overcome this challenge, to ensure efficient use of memory and disk resources. Using memory database management along with performance in memory, using intelligent techniques to utilize primary memory and disk storage, organizations can dramatically increase big data processing performance this approach makes it easily strike a balance between speed and storage capacity, greatly improving the use of large amounts of data.

Furthermore, the columnar layout offers increased potential for parallel processing. Since the data is vertically partitioned, operations on different columns can easily be processed concurrently. For instance, when a query requires searching or aggregating multiple columns, it can be divided into smaller tasks, and each task can be assigned to a separate processor core for parallel execution. In contrast, row-based database

systems store complete records or rows contiguously in the storage media. Row-based layouts are more suitable for transactional workloads that involve accessing multiple attributes of individual records and require fast write operations. Employing a hybrid strategy that incorporates both columnar and row-based layouts, particularly for hot and cold data, can significantly enhance read/write operations and improve query performance [31]. By leveraging the strengths of each approach based on the specific data access patterns, organizations can optimize their big data processing and achieve more efficient performance.

2.2. Methods that leverage the locality of references

Methods aimed at improving the localities of references primarily focus on organizing data in a manner that facilitates easier access. This can be achieved through the utilization of efficient data structures like hash tables or B-trees. Caches, due to their limited size, are only capable of holding copies of recently used data or code. Typically, when new data is loaded into the cache, it replaces other existing data. Caches contribute to performance improvements solely if the previously loaded data is reused before being replaced. The significant reduction in program execution time achieved by caches can be attributed to the principle of locality of references [32]. This principle asserts that most programs do not uniformly access all code or data. Instead, recently used data and data located in proximity to the currently referenced data are highly likely to be accessed soon. Consequently, locality can be categorized into temporal locality (locality by time) and spatial locality (locality by space) [33]. It is worth noting that code optimization techniques [34] and data restructuring techniques [35] both aim to enhance both temporal and spatial locality.

In a study conducted by [18], the phenomenon of locality was investigated over time and across different applications, with a specific emphasis on its application to computer systems and networks.

Finally, previous research has shown that there is a strong tendency to improve big data performance. However, the best method for a particular application will depend on the specific characteristics of the application. This article focuses on enhancing big data performance by enhancing the locality characteristics of the big data and dynamic data structure selection to provide a more adaptive and efficient approach to exploiting reference locality.

3. Proposed Model:

Big data processing faces performance challenges due to memory constraints [36]. When the memory required to store data is much smaller than the total data size, accessing the data involves transferring it from massive storage to limited memory, commonly known as paging. This leads to expensive data transfers between memory and disk space, causing CPU waiting time. To mitigate this, the quality of the locality of reference plays a key role. If data remains in memory for a longer duration and is less likely to be reloaded, data transfer and CPU waiting time are reduced, resulting in enhanced performance for big data processing.

The proposed model focuses on conducting ontology analysis to identify the locality of reference in big data and to enhance the quality of the locality of reference by restructuring the data. To achieve this, we will create a synthetic memory reference trace based on the behavior of database queries. Using a graph representation, we will capture the temporal and spatial relationships between data references. A clique, which consists of completely connected nodes, will indicate a group of data items forming a data locality. This group of data will be included in its structure, such that when a reference is made to any of its members, the entire block will be loaded into memory. The quality of big data will be evaluated based on the discovered characteristics of the locality of reference within the data.

The procedure of the proposed model is described in the following steps:

1. Generate a synthetic memory reference trace by analyzing the behavior of database queries. This log provides important insights into the access and reference methods used in the dataset. The reference string specifies the order in which data is accessed into the memory space, labeling each data item as D1, D2, D3, and so on, to determine how these data elements are presented in the database context as

$$DataItem\ (D) = DB(d):Table(t):Record(r):Field(f) \quad (1)$$

2. We will utilize a graph representation to capture the temporal and spatial relationships between data references. This graph will allow us to visually analyze and understand the connections between different data items. The concept of locality of references can be defined within this graph, denoted as G={V, L}, where V is the set of nodes and the L is the set of links in the graph G. The graph G is based on the referencing behavior of the data in the application. In the graph, each data item (page) corresponds to a node in V, while an edge in L is established between two pages, Di and Dj if Dj is referenced within a certain number of hops or time units from Di and the number of times Dj is referenced after Di is recorded as n. The lowest degree of locality, indicating weaker data referencing patterns, is indicated when the ratio of

hops (k) to frequency of references (n) approaches infinity.

$$k/n \rightarrow (infinite) \quad (2)$$

In other words, the larger the distance between pages and the lower the frequency of being referenced together, the lower the locality of reference. Furthermore, a clique in the graph, representing a set of completely connected nodes, signifies a group of data items that exhibit strong locality characteristics of data.

3. The quality of the big data can be assessed by examining the characteristics of the identified locality of reference. One measure used to quantify the level of locality in a big data system is the ratio of L (locality measure index) to S (data size) [37]. L is calculated as the sum of individual locality measures (Li) multiplied by a weighting factor (q), according to equation (3).

$$L = (\Sigma Li) * q \quad (3)$$

To demonstrate the process of our approach, we will use a practical case. Let's imagine we have a program that interacts with a substantial database containing tables, records, and fields. To analyze the program's memory behavior, we monitor its memory references and generate a reference string that represents the order in which data items are accessed. This reference string captures the sequence of data item references made by the program.

4. Use-Case:

In the following case sequence, we refer to a data item labeled as Di, assuming it represents a virtual page:

D1,D1,D1,D2,D3,D1,D1,D2,D3,D4,D1,D5,D2,D3,D3,D5,D6,D5,D6

We define the distance between two items as the number of hops between Di and Dj. For example, the distance d1,2 between D1 and D2 = 1; and d1,3 = 2; Table 1 is built to show the distances between the data items (pages) as follows.

Table 1: Trace Data Item Matrix

	D1	D2	D3	D4	D5	D6
D1	0	1^2, 2	2^2,3	3	1,5	6
D2		0	1^3,2	2	4,3	4
D3			0	1	1,2	2
D4				0	2	7
D5					0	1^2
D6						0

The index 12 means that page 2 is referenced two times at a distance of 1 from page 1. Using this table, we can extract the locality of reference characteristics for the program. We build a graph for this case as shown in Fig 2.

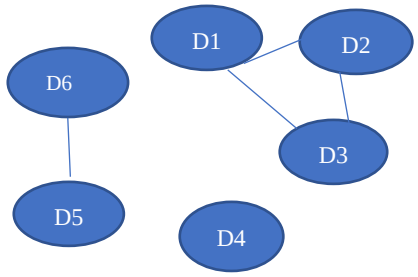

Figure. 2: Data item Reference Behavior Graph

As can be seen from Fig 2, we build an edge between two nodes if the distance is less than 3 and the frequency of the distance is more than one. This means that the two connected pages are referenced within a short time and referenced frequently.

In the graph, pages D1, D2, and D3 are referenced within close distance to each other and referenced more than once within each other. This indicates that nodes D1, D2, and D3 form a locality of reference. The same for pages D5, and D6. Node D4 does not belong to any locality.

The concept of locality of reference is a fundamental aspect of data quality, especially in the context of big data. To identify the locality characteristics of the data, our approach focuses on the ontology part. This process involves creating a memory reference graph by monitoring the program's interactions with advanced data sets. By constructing a graph that reflects the reference pattern and identifying the overlapping groups (called cliques) in the graph, we can accurately characterize the data at the local level.

This research aims to increase temporal and spatial locality by ways to reorganize the data. While many approaches focus only on improving location, such as using in-memory technologies, our approach also addresses temporal regions separately moreover vertical separation and a mixture of vertical ones are used, distinguishing our approach from those typically used for column-based databases. Let's differentiate the number of jobs based on vertical segmentation

5. Conclusion

In summary, the ongoing challenge of increasing big-data performance remains a major focus in the field. This study addresses this challenge by emphasizing the importance of the concept of context as an important factor. Through ontology analysis, we identified specific spatial attributes that are well embedded in big data, which we created by creating simulated memory reference records from database query patterns The use of graph representations allowed us to interpret them visually so we examined temporal and spatial relationships between data references. Creating local groups in a graph and finding overlapping data objects allows us to greatly simplify data processing. This approach enables transactions to be executed on data residing in the fastest memory layer, leading to enhanced performance. In future work, it would be valuable to explore the potential integration of machine learning techniques to automatically detect the most efficient locality structure in big data, further improving performance and scalability.

Acknowledgment

The authors would like to acknowledge and thank Dr. Umaiya Murad –from the Association of Arab Universities, for her efforts in reviewing the paper and providing the authors with valuable comments to improve the paper.

Compliance with Ethical Standards and Conflict of Interest

The authors declare that there is no conflict of interest regarding the publication of this paper, and strictly complied

Author Contribution Statement

Dr. Mohammad Malkawi proposed the research problem, and both Dr. Methaq Alnoori and Enas Rawashdeh researched the topic and validated the research theme. All three authors contributed to the investigation of the research theme and discussed the model and the results. Dr. Methaq and Dr. Enas compiled the reference list and summarized related research.

References

[1] S. Usman, R. Mehmood, I. Katib, and A. Albeshri, "Data Locality in High-Performance Computing, Big Data, and Converged Systems: An Analysis of the Cutting Edge and a Future System Architecture," Electronics, vol. 12, no. 1, p. 53, 2022. DOI: https://doi.org/10.20944/preprints202211.0161.v1

[2] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of things data analysis: a survey," Digit. Commun. Networks, vol. 4, no. 3, pp. 161–175, 2018, doi: https://doi.org/10.1016/j.dcan.2017.10.002.

[3] J. Yu, M. Ai, and Z. Ye, "A review on design inspired subsampling for big data," Stat. Pap., pp. 1–44, 2023. DOI: https://doi.org/10.6339/21-JDS999

[4] A. A. Hamad and M. J. Farhan, "A NEW MIMO SLOT ANTENNA FOR 5G APPLICATIONS," J. Eng. Sustain. Dev., vol. 24, no. 6, pp. 33–41, 2020. DOI: https://doi.org/10.31272/jeasd.24.6.3

[5] J. Ousterhout et al., "The case for RAMClouds: scalable high-performance storage entirely in DRAM," ACM SIGOPS Oper. Syst. Rev., vol. 43, no. 4, pp. 92–105, 2010.DOI: https://doi.org/10.1145/1965724.1965751

[6] A. H. Rashed and M. H. Hamd, "Robust detection and recognition system based on facial extraction and decision tree," J. Eng. Sustain. Dev., vol. 25, no. 4, pp. 40–50, 2021. DOI: https://doi.org/10.31272/jeasd.25.4.4

[7] P. Vagata and K. Wilfong, "Scaling the Facebook data warehouse to 300 PB," Faceb. Code, Faceb., vol. 10, 2014. https://code.facebook.com/posts/229861827208629/ scaling-the-Facebook-data-warehouse-to-300-pb/

[8] H. Herodotou and S. Babu, "Profiling, what-if analysis, and cost-based optimization of MapReduce programs," Proc. VLDB Endow., vol. 4, no. 11, pp. 1111–1122,2011. https://doi.org/doi.org/10.1145/2522968.2522979

[9] S. I. M. Mosharraf and M. A. Adnan, "Hwang, Eunji, et al. 'Exploring memory locality for big data analytics in virtualized clusters.' Proceedings of the 2017 Symposium on Cloud Computing. 2017.," J. Big Data, vol. 9, no. 1, pp. 1–30, 2022. DOI: https://doi.org/10.1109/CCGRID.2018.00017

[10] Y. Lu, "Zhang et al., 2015." Massachusetts Institute of Technology, 2017. https://doi.org/10.1007/s13351-017-7088-0

[11] M. Kadhum, E. Rawashdeh, and M. Alshraideh, "An Efficient Bug Reports Assignment for IoT Application with Auto-Tuning Structure of ELM Using Dragonfly Optimizer," J. Hunan Univ. Nat. Sci., vol. 48, no. 7, 2021.

[12] E. F. Rawashdeh, I. Aljarah, and H. Faris, "A cooperative coevolutionary method for optimizing random weight networks and its application for medical classification problems," J. Ambient Intell. Humaniz. Comput., vol. 12, pp. 321–342, 2021. DOI: https://doi.org/10.1007/s12652-020-01975-3

[13] L. M. Abualigah, A. T. Khader, and M. A. Al-Betar, "Unsupervised feature selection technique based on genetic algorithm for improving the text clustering," in 2016 7th International Conference on computer science and information technology (CSIT), IEEE, 2016, pp. 1–6. DOI: https://doi.org/10.1109/CSIT.2016.7549453

[14] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: a survey. Dig. Commun. Netw." Press, 2017. https://doi.org/10.1016/j.dcan.2017.10.002

[15] M. Kadhum, S. Manaseer, and A. L. Abu Dalhoum, "Cloud-edge network data processing based on user requirements using modify mapreduce algorithm and machine learning techniques," Int. J. Adv. Comput. Sci. Appl., vol. 10, no. 12, 2019, doi: https://doi.org/10.14569/ijacsa.2019.0101242.

[16] M. Kadhum, M. H. Qasem, A. Sleit, and A. Sharieh, Efficient MapReduce matrix multiplication with optimized mapper set, vol. 574. 2017. doi: https://doi.org/10.1007/978-3-319-57264-2_19.

[17] J. Jin et al., "A data-locality-aware task scheduler for distributed social graph queries," Futur. Gener. Comput. Syst., vol. 93, pp. 1010–1022, 2019. https://doi.org/10.1016/j.future.2018.04.086

[18] R. Bunt and C. Williamson, Temporal and spatial locality: A time and a place for everything. na, 2003. https://doi.org/10.1145/301618.301668

[19] Z. Sha, Z. Cai, F. Trahay, J. Liao, and D. Yin, "Unifying temporal and spatial locality for cache management inside SSDs," in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE), IEEE, 2022, pp. 891–896. doi: https://doi.org/10.23919/DATE54114.2022.9774532

[20] M. Y. Özkaya, A. Benoit, and Ü. V Çatalyürek, "Improving locality-aware scheduling with acyclic directed graph partitioning," in Parallel Processing and Applied Mathematics: 13th International Conference, PPAM 2019, Bialystok, Poland, September 8–11, 2019, Revised Selected Papers, Part I 13, Springer, 2020, pp. 211–223. https://doi.org/10.3390/s18061676

[21] H. Zhang, G. Chen, B. C. Ooi, K.-L. Tan, and M. Zhang, "In-memory big data management and processing: A survey," IEEE Trans. Knowl. Data Eng., vol. 27, no. 7, pp. 1920–1948, 2015. DOI: https://doi.org/10.1109/TKDE.2015.2427795

[22] H. S. Stone, "A logic-in-memory computer," IEEE Trans. Comput., vol. 100, no. 1, pp. 73–78, 1970. DOI: https://doi.org/10.1109/TC.1970.5008902

[23]"(PDF) Performability analysis of wireless cellular networks." Accessed: Feb. 18, 2022. [Online]. Available: https://www.researchgate.net/publication/255821609_Performability_analysis_of_wireless_cellular_networks. DOI: https://doi.org/10.1002/dac.605

[24] D. Patterson et al., "A case for intelligent RAM," IEEE micro, vol. 17, no. 2, pp. 34–44, 1997. DOI: https://doi.org/10.1109/40.592312

[25] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications," IEEE Internet Things J., vol. 4, no. 5, pp. 1125–1142, 2017, doi: https://doi.org/10.1109/JIOT.2017.2683200.

[26] V. Seshadri, "Fast and Energy-Efficient In-DRAM Bulk Data Copy and Initialization". https://doi.org/10.1145/2540708.2540725

[27] V. Seshadri et al., "Gather-scatter DRAM: In-DRAM address translation to improve the spatial locality of non-unit strided accesses," in Proceedings of the 48th International Symposium on Microarchitecture, 2015, pp. 267–280. https://doi.org/10.1145/2830772.2830820

[28] J. DeBrabant, A. Pavlo, S. Tu, M. Stonebraker, and S. Zdonik, "Anti-caching: A new approach to database management system architecture," Proc. VLDB Endow., vol. 6, no. 14, pp. 1942–1953, 2013. https://doi.org/10.14778/2556549.2556575

[29] R. Stoica and A. Ailamaki, "Enabling efficient OS paging for main-memory OLTP databases," in Proceedings of the Ninth International Workshop on Data Management on New Hardware, 2013, pp. 1–7. https://doi.org/10.1145/2485278.2485285

[30] J. J. Levandoski, P.-Å. Larson, and R. Stoica, "Identifying hot and cold data in main-memory databases," in 2013 IEEE 29th International Conference on Data Engineering (ICDE), IEEE, 2013, pp. 26–37. doi: https://doi.org/10.1109/ICDE.2013.6544811

[31] A. W. J. Lu, A study of an in-memory database system for real-time analytics on semi-structured data streams. University of Toronto (Canada), 2015.

[32] J. L. Hennessy and D. A. Patterson, Computer architecture: a quantitative approach. Elsevier, 2011. ISBN: 9780123838735

[33] S. Manegold, "Memory Locality BT- Encyclopedia of Database Systems," L. Liu and M. T. Özsu, Eds., New York, NY: Springer New York, 2016, pp. 1–2. doi: 10.1007/978-1-4899-7993-3_686-2. https://doi.org/10.1145/3357526.3357571

[34] E. N. Rush, B. Harris, N. Altiparmak, and A. Ş. Tosun, "Dynamic data layout optimization for high performance parallel i/o," in 2016 IEEE 23rd International Conference on High Performance Computing (HiPC), IEEE, 2016, pp. 132–141. doi: https://doi.org/10.1109/ISPASS48437.2020.00025

[35] M. Kaufmann, "Storing and processing temporal data in main memory column stores." ETH Zurich, 2014. https://doi.org/doi.org/10.14778/2536274.2536333

[36] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," Journal of Network and Computer Applications, vol. 98. pp. 27–42, 2017. doi: https://doi.org/10.1016/j.jnca.2017.09.002.

[37] P. J. Denning and J. P. Buzen, "The operational analysis of queueing network models," ACM Comput. Surv., vol. 10, no. 3, pp. 225–261, 1978. https://doi.org/10.1145/356733.356735