DOI: https://doi.org/10.26565/2304-6201-2025-66-05

УДК (UDC) 004.65

Pugach Mykyta *PhD student, Department of Theoretical and Applied Computer Sciences Karazin Kharkiv National University, Svobody Sq 4, Kharkiv, Ukraine,61022*
e-mail: mykyta.pugach@karazin.ua
https://orcid.org/0009-0004-8923-6489

A Systematic Review on Workload Change Detection in Distributed Databases

Distributed Databases became essential part of a large part of nowadays software. It has numerous of advantages including scalability, fault tolerance, high availability, and improved performance. It solves a lot of problems of centralized databases but can also suffer with challenges. One of them is skewed access. Workload in distributed DBMS often changes, such fluctuations can cause ineffective operation of the system. Imagine access to one row of database became 10 times more frequent, or complex requests start operating with the data highly distributed geographically. Such behavior shows that initial data distribution cannot be always efficient enough. And to address this problem adoptive design technics were invented. In this article we review the common steps of adoptive technics and concentrate attention at workload detection and hot data identification.
The purpose of the article is to introduce adoptive design approach of distributed database management systems, review and analyze existing technics and theirs steps, especially workload change detection and hot data identification. The final goal is to compare theses technics and lead out their main concerns.
As a result of this work some existing approaches were analyzed and highlighted their common parts alongside with differences, presented their main issues.
After reviewing all technics, we can see that current solutions cannot give precise results without creating much overhead to the system. Also, there is no approach to giving up-to-date information about hot data without creating overhead. Overhead in such situations is a major issue. In skewed access patterns distributed nodes can become very busy with processing queries and additional computations can lead to worse overall system performance then without adoptive design or even to node outage. So, search for solutions, that give precise and up-to-date results without significant overhead is a big field of future researches.

Key words: *distributed databases, adoptive design technics, hot data identification, workload change detection.*

How to quote: M. Pugach, “A Systematic Review on Workload Change Detection in Distributed Databases”, *Bulletin of V. N. Karazin Kharkiv National University, series Mathematical modelling. Information technology. Automated control systems*, vol. 66, pp. 56-62, 2025. https://doi.org/10.26565/2304-6201-2025-66-05

Як цитувати: Pugach M. A Systematic Review on Workload Change Detection in Distributed Databases. *Вісник Харківського національного університету імені В. Н. Каразіна, серія Математичне моделювання. Інформаційні технології. Автоматизовані системи управління*. 2025. вип.. 66. С.56-62. https://doi.org/10.26565/2304-6201-2025-66-05

Introduction

Workload of distributed database management systems with online transaction processing (OLTP) is not static. It can significantly change due to daily, weekly, or seasonal fluctuations in demand, or because of rapid growth in demand due to a company’s business success. This often causes imbalances in the load on the nodes of a distributed DBMS. Such situations may happen when database holds “hot” tuples or range of tuples, which means this data is much more in demand. For example, music streaming platforms have trendy songs or albums. According to 2023-year statistics [1] there were 463,000 tracks streamed at least a million times and 45.6 million tracks, which had zero streams. But trends come and go, which causes changes in database workloads. Fluctuations may provoke increases in distributed query execution time or even node fails because of excessive load.
To address this issue, some modern distributed DBMSs apply adaptive design approaches. Key point is to perform incremental redesign, which means that data may be dynamically repartitioned during the system's runtime. There are three interrelated issues that need to be tackled in adaptive distribution design [2][3]:

4.

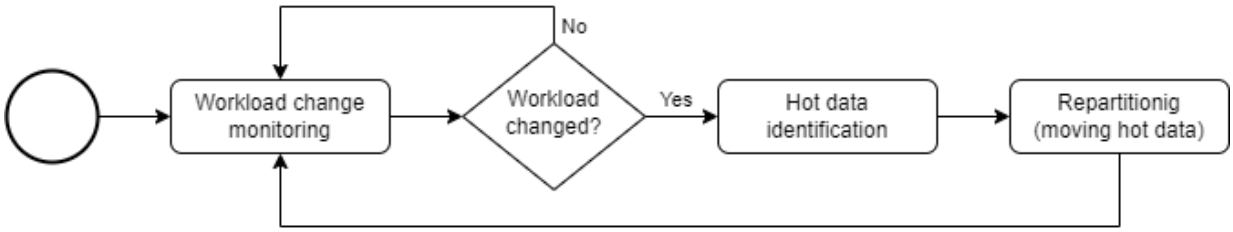

Fig. 1 Adoptive design approach flowchart.
Рис. 1 Схема підходу до проектування на основі адаптивного дизайну

For the last decades, adaptive design has been a topic for active research and quite a few adaptive techniques have been proposed. But most of them assume that the change in the workload is detected and simply focus on the migration problem. Nevertheless, some of these studies presented solutions for the first two problems, while others even addressed all of them. This article investigates existing approaches for workload changes detection and hot processed data identification. Highlighted their main challenges and ways they solve them. And provided analysis of drawbacks of each solution.

1. Why are adaptive design approaches essential?

In nowadays, when the amount of data increasing exponentially and users demand faster and faster access to it, it is very essential for organizations to have systems which will meet these requirements. More and more companies choose to use distributed databases because they have improved scalability and fault tolerance, as they allow data to be stored across multiple nodes and locations, ensuring that the system can grow efficiently and remain operational even in the case of node failures. Additionally, they provide enhanced availability by distributing data across various sites, minimizing the risk of downtime.

So, modern distributed DBMSs demand high throughput, low latency, and continuous availability. However, these requirements become challenging to maintain when data access is skewed, or workload patterns change. Significant shifts in access patterns can cause some nodes in a DBMS cluster to become overloaded while others remain underutilized, reducing performance despite sufficient total resources.

Experiments demonstrate [3] that as the skew increases, system throughput decreases, and latency rises. This imbalance occurs because heavily loaded partitions accumulate longer queues, resulting in higher latencies, while underutilized partitions remain idle, decreasing overall efficiency. Furthermore, CPU utilization becomes uneven, with highly loaded partitions showing significantly higher usage compared to others, amplifying the imbalance, and impacting system performance.

2. Workload change detection

Detecting the workload change can be called the first step of an incremental redesign process. The goal is to understand whether the system needs to be reconfigured. Naturally all the nodes won't have completely same load. Yet they still have near-uniform distribution. Skews in workload usually could be modeled as Zipfian distribution. So, the higher the value of Zipfian's distribution exponent (α) parameter, the higher skew we have. If it is small enough, then we have a skew which does not affect the system that much, that needs to be reconfigured. So, as a result of this phase a system needs to be aware that there is a significant workload change.

We can highlight two different approaches to achieving this. The first one is based on scanning system metrics and the second is counting the number of distributed transactions for each partition.

2.1. E-Store

The first approach was presented in E-Store, an elastic partitioning framework for distributed OLTP DBMSs [3]. E-Store has an E-Monitor component which addresses workload change detection.

In the initial phase, E-Monitor gathers CPU utilization metrics for each partition on the DBMS's nodes at an OS level (where each partition maps to a core). This high-level, coarse-grained data is easy to collect and offers sufficient insights. CPU usage in a main-memory DBMS is a reliable indicator of overall system performance. When E-Monitor polls a node, it collects the current utilization for all partitions on

that node and calculates the moving average over the last 60 seconds. E-Monitor employs two thresholds – high-watermark (e.g., 90%) and low-watermark (e.g., 50%) to decide whether intervention is required. These thresholds, adjustable by the DBA, reflect a balance between system responsiveness and resource use. If either threshold is surpassed, the E-Monitor initiates a more granular monitoring phase at the tuple level.

2.2. Clay

Also, Clay adaptive partitioning approach [4] uses this technique. It works like E-Monitor but examines SLA (service-level agreements).

2.3. Kairos

The system, named Kairos [5], uses analogous approach. Its resource monitor queries the OS and DBMS running on each machine for statistics about CPU, RAM, and disk I/O, buffer pool utilization, and log flushes.

2.4. SWORD

The second approach was presented in SWORD, a scalable workload-aware data partitioning and placement approach for OLTP workloads [6]. SWORD model the workload as a hypergraph, where each hyperedge corresponds to a transaction or a query, and employ hypergraph partitioning algorithms to guide data placement decisions. Such a graph-based approach was presented in the work of Curino et al. [7].

SWORD offers technique that counts transactions. The system monitors the percentage increase in the number of distributed transactions and determines that the changes are significant enough to require reconfiguration if this percentage increase exceeds a defined threshold. More specifically, SWORD sets a task using *min-cut* term. The *min-cut* problem is a standard concept in graph theory, where the goal is to partition a graph (representing the system's data here) into two parts, such that the number of edges (representing data dependencies or connections) crossing the partition is minimized. In SWORD, the min-cut is used to assess how well the data is distributed across partitions. This allows SWORD to dynamically adapt its configuration to maintain optimal performance when the system’s load changes, particularly when access patterns change or when there is a significant increase in the number of transactions. Here threshold is also a system parameter which can be set depending upon the sensitivity of applications to latency.

Thus, SWORD observes the rate of increase in load, allowing it to promptly react to increases in transaction volume, which might indicate the need to reconfigure data placement to maintain system efficiency. This adaptive reconfiguration helps maintain balance between the distributed data and the load on various parts of the system, which is crucial for stable operation under changing conditions.

2.5. Summary

Through the evaluation phase, each method validated its capability to solve the assigned problem. However, both have their pros and cons. The primary advantage of E-Monitor is low overhead, which is achieved by periodicity poling nodes’ CPU and CPU metrics themselves, because system metrics are pretty cheap. On the other hand, SWORD tracks all the transactions, which produces much greater overhead. But this approach is supposed to be more accurate than CPU utilization. Clay in its turn uses SLA which produces not much more overhead than checking CPU but is more precise.

Table 1. Workload change detection approaches summary.
Таблиця 1. Підсумок підходів до виявлення змін робочого навантаження.

	Low overhead	**Precision**
E-store	**+**	**+-**
Clay	**+**	**+**
Kairos	**+**	**+-**
SWORD	**-**	**+**

3. Identify which data to move.

Once workload change is detected, adaptive incremental redesign techniques need to select a bunch of data that should be moved to another partition. As the purpose of the entire process is load balancing, data should meet several requirements. The main criterion is hot processing. Skews in workloads most often happen due to increasing demand for a limited number of tuples. Some adoptive design approaches also measure interconnections between tuples. The ideal case occurs when each transaction “matches” a

partition because the transaction has to access that only partition [8]. So, in this phase system needs to choose the data that will be moved.

In general, all existing approaches could be split into two groups: inline and offline. And these approaches have such common characteristics: inline produces overhead to the system but results with up-to-date information, unlike offline techniques processes data separately from transactions and do not produce overhead but result with a delay.

3.1. Siberia

We will start with offline approaches. The first one was presented by Levandoski et al. [9]. Their project was called Siberia; its goal was hot records classification. This approach, in short, uses logging combined with algorithmic operations to detect active or frequently accessed records.

Let us consider it more precisely. Siberia associates each record with a discrete time slice, denoted $[t_n, t_{n+1}]$. The next time interval begins at t_{n+1} and ends at t_{n+2}, and continues similarly. Time is tracked by the number of record accesses, with each "tick" of the clock occurring after each record access. A time slice is identified using its beginning timestamp (t_n for $[t_n, t_{n+1}]$). A time slice denotes a distinct period during which record access is detected, and conceptually, log records (RecordID, TimeSlice) pairs. In practice, the log keeps a sequence of record IDs in the order of access, separated by time markers that indicate the boundaries of each time slice.

Siberia uses exponential smoothing to estimate record access frequencies. Exponential smoothing calculates an access frequency estimate for a record *r* as

$$est_r(t_n) = \alpha * x_{t_n} + (1 - \alpha) * est_r(t_{n-1})$$

Where, t_n denotes the current time slice, while x_{t_n} represents the observed value at t_n. In this model, x_{t_n} is set to 1 if the record r is accessed during t_n; otherwise, it is 0. The term $est_r(t_{n-1})$ refers to the estimate derived from the preceding time slice, t_{n-1}. The parameter α serves as a decay factor, controlling the influence of new observations and the rate at which past estimates lose significance. Typically, α is chosen in the range of 0.01 to 0.05, where higher values prioritize recent observations more heavily.

The authors claim that Siberia does not use every record access for its algorithms, because it may degrade system performance. To minimize system overhead, they adopt a sampling-based approach for logging. Each worker thread determines whether to log its activity by flipping a biased coin, with the bias corresponding to the sampling rate. Depending on the result of the coin flip, the thread either records its data in log buffers or skips the logging process. The authors state that sampling only 10% of the accesses reduces the accuracy by only 2.5%.

There were four algorithms presented, all of them take the same data as input and result the same but there are significant differences between them in performance. They take stored logs, described before, and parameter that signifies the number of records to classify as hot. Authors propose two basic algorithms: forward and backward. The forward algorithm simply scans the log forward from a beginning time slice. It updates *r*'s current access frequency estimate using the exponential smoothing equation. This forward algorithm has two primary drawbacks: it requires a scan of the entire log, and it requires storage proportional to the number of unique record ids in the access log.

Backward algorithm avoids scanning the entire log from beginning to end in order to improve classification performance. The primary concept involves scanning the log in reverse order and derive successively tighter upper and lower bounds for the estimates of accessed records. Occasionally, the algorithm classifies records based on these bounds, allowing it to potentially stop the scan earlier. It maintains estimates only for records that are still contenders for the hot set, thereby limiting its memory usage to the number of hot records rather than the total record count.

Backward algorithm also uses exponential smoothing:

$$estb_r(t_n) = \alpha(1 - \alpha)^{(t_e - t_n)} + estb_r(t_{last})$$

where $t_{last} > t_n$ since scanning in reverse.

Calculates upper bound and lower bound:

$$upEst_r(t_n) = estb_r(t_n) + (1 - \alpha)^{t_e - t_n + 1}$$
$$loEst_r(t_n) = estb_r(t_n) + (1 - \alpha)^{t_e - t_b + 1}$$

where t_e means end time slice and t_b – the first time slice.

As the backward classification approach continues processing more record accesses, the upper and lower bounds converge toward an exact estimate.

Authors use two optimizations for this algorithm. First, dropping records with upper bound values less than k^{th} record lower bound value, where *k* is parameter, which determines needed size of resulting hot set. This optimization origins from the properties of upper and lower bounds. Record would never have an estimation bigger than upper bound and less than lower bound. Second, algorithm can stop when only *k* records are still in contention for the hot set. So, the backward approach is much more efficient than the forward.

The researchers also propose parallel variants of these two algorithms. These solutions are more efficient, as expected. Backward algorithm split logs into *N* parts and use controller-worker processes scheme to calculate hot records set. As parallel backward algorithm is the most efficient, Siberia uses it.

3.2. SWORD

We have already discussed how SWORD [6] addresses workload detection. And mentioned that it uses hypergraph representation model for workload, where hyperedges are transactions or queries. SWORDs technique of 'data to move' recognition is inextricably connected with its repartitioning method. Its approach is based on efficiently identifying candidate sets of data items whose migration has the potential to reduce the frequency of distributed transactions the most and then performing the migrations during periods of low load.

More specifically, authors use the term *virtual node* which is a logical abstraction of physical node. One physical node can contain few virtual nodes. The initial distributed database design defines such nodes in such a way, that data inside them are highly interconnected and data between these nodes are loosely connected. Using graph-based interpretation system uses the *min-cut*. Data identification step starts once min-cut crosses the threshold. The algorithm manages pairs of candidate virtual node sets that can be exchanged to decrease the min-cut size. It performs some of such swaps per step, aiming to minimize the *min-cut* of the data placement based on the current workload. This process continues until the *min-cut* falls below the specified threshold.

So, here the data identification process united with repartitioning plan. The system selects data according to number of distributed transactions. Such an approach can be less efficient than hot data identification, because the number of hot repartitions is less than others. Thus, the system has to make more moves, which provides significantly more overhead.

3.3 Apollo

The framework called Apollo [10] uses graph-based workload visualization, too. But unlike SWORD it utilizes query patterns instead of queries. So, in place of such a request:

```
SELECT C_ID FROM CUSTOMER WHERE C_UNAME = 'Bob' AND C_PASSWD = 'pwd'
```

The system will save such query pattern:

```
SELECT C_ID FROM CUSTOMER WHERE C_UNAME = ? AND C_PASSWD = ?
```

While this reduces the granularity of determining the exact set of data items that are affected, it may allow the detection of additional data items that might be affected by similar queries and reduce the frequency of changes that are necessary.

3.4. SAHARA

Project, called SAHARA [11], introduces an inline approach of hot data identification. It focuses on two key types of data access: domain access and row access. As SAHARA works with column stores, domain means a set of values for a particular attribute. Domain accesses are recorded to evaluate potential partitioning layouts, while row accesses are captured to estimate the memory footprint of the partitioning. This data is collected over defined time windows to avoid biases from short-term access bursts, ensuring that the statistics reflect long-term patterns relevant to memory management strategies, such as buffer pool eviction policies.

To manage memory efficiency, accesses are recorded in blocks rather than individually, which reduces the memory overhead but may introduce imprecision in access frequency measurements. The block sizes are adjusted experimentally to balance accuracy with memory usage, with a maximum overhead of 1% of the dataset size. Furthermore, the methodology incorporates row and domain block counters to track accesses to specific data partitions and attribute values within given time frames. This approach allows for a more granular understanding of data access patterns. As statistics is gathered, SAHARA classifies hot data using π-second-rule. It means that data is hot if it is accessed more often than every π-seconds,

where π is a settings parameter. Authors emphasize that the time window length should not be set substantially smaller than π. In addition, the Nyquist–Shannon sampling theorem proves that a sample rate of $\pi/2$ is sufficient to achieve precise statistics. Therefore, the time window length is set to $\pi/2$.

3.5. E-Store

Taft et al. [3] in their research suggested approach based on time windows, too. The system called E-Store makes tuple-level monitoring on the entire cluster for a short period of time. Authors assume all non-replicated tables of an OLTP database form a tree-schema based on foreign key relationships. Although this rules out graph-structured schemas and m-n relationships, it applies to many real-world OLTP applications, they claim. Considering such assumption, monitoring only the root tuples provides a good approximation of system activity and minimizes the overhead of this phase.

E-Store's monitoring system, called E-Monitor classifies hot tuples as the top-k most frequently accessed tuples within a given time window. A tuple is considered "accessed" if it is read, modified, or inserted during a transaction. The system collects two types of data: the total number of accesses to tuples within a partition (*L*) and the subset of top-k most frequently accessed tuples (*TK*). When tuple-level monitoring is activated, the database management system (DBMS) sets up an internal histogram for each partition, which tracks how many times each tuple has been accessed by a transaction. After the time window concludes, the execution engine at each node compiles *L* and *TK* for its local partitions and forwards this data to E-Monitor. E-Monitor consolidates the information from all partitions to create a global top-k list.

3.6. Summary

We reviewed several technics of hot data identification. They use completely different approaches, like gathering query logs or collecting tuple level statistics. All of them has their advantages along with disadvantages. Processing all computations on another CPU, like in Siberia, lowers overhead nicely but with that system receives not up-to-date information. E-Store's tuple level monitoring provides actual information but can produce critical overhead in the busiest nodes. Graph-based solutions frequently are not enough precise and universal.

4. Conclusion

With the rise of amount of information and demand of quick access to it, distributed databases became the essential technology. But often initial data allocation becomes ineffective and reduces all advantages of distributed DBs. Here comes adoptive design approaches. They allow dynamic data reconfiguration to keep the system highly efficient.

We discussed that adoptive design technics mostly contain three steps and reviewed the first two of them: workload change detection and hot data identification. Several DBMSs implemented these steps, and we can see that there are numerous of approaches how to do that. In general, adoptive techniques are balancing between high efficiency and high precision, low overhead and timeliness. Overhead in such situations is a major issue. In skewed access patterns distributed nodes can become very busy with processing queries and additional computations can lead to worse overall system performance then without adoptive design or even to node outage. So, it's crucial to keep developing these approaches to meet all these characteristics at once to create more efficient distributed database management systems.

REFERENCES

1. Luminate Data, LLC, "Year-End Music Industry Report 2023," Luminate Data, LLC, 2023. [Online]. Available: https://luminatedata.com/reports/yearend-music-industry-report-2023/. [Accessed: Nov. 27, 2024]
2. M. T. Özsu and P. Valduriez, *Principles of Distributed Database Systems. 4th edition*. Cham, Switzerland: Springer Nature, 2020.
3. R. Taft et al., "E-Store: Fine-grained elastic partitioning for distributed transaction processing systems", *Proceedings of the VLDB Endowment*, vol. 8, no. 3, pp. 245 – 256, 2014. https://doi.org/10.14778/2735508.2735514.
4. M. Serafini, R. Taft, A. J. Elmore, A. Pavlo, A. Aboulnaga and M. Stonebraker, "Clay: Fine-grained adaptive partitioning for general database schemas", *Proceedings of the VLDB Endowment*, vol. 10, no. 4, pp. 445 – 456, 2016. https://doi.org/10.14778/3025111.3025125.

5. C. Curino, E. P. C. Jones, S. Madden and H. Balakrishnan, “Workload-aware database monitoring and consolidation”, in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data.* Athens, 2011, pp. 313 – 324. https://doi.org/10.1145/1989323.1989357.
6. A. Quamar, K. A. Kumar and A. Deshpande, “SWORD: Scalable workload-aware data placement for transactional workloads”, in *Proceedings of the 16th International Conference on Extending Database Technology*. Genoa, 2013, pp. 430 – 441. https://doi.org/10.1145/2452376.2452427.
7. C. Curino, E. Jones, Y. Zhang and S. Madden, “Schism: A workload-driven approach to database replication and partitioning”, *Proceedings of the VLDB Endowment*, vol. 3, no. 1-2, pp. 48 – 57, 2010. https://doi.org/10.14778/1920841.1920853.
8. S. Navathe, S. Ceri, G. Wiederhold and J. Dou, “Vertical partitioning algorithms for database design”, *ACM Transactions on Database Systems*, vol. 9, no. 4, pp. 680 – 710, 1984. https://doi.org/10.1145/1994.2209.
9. J. J. Levandoski, P.- Å. Larson and R. Stoica, “Identifying hot and cold data in main-memory databases” in *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE)*. Brisbane, 2013, pp. 26 – 37. https://doi.org/10.1109/ICDE.2013.6544811.
10. B. Glasbergen, M. Abebe, K. Daudjee, S. Foggo and A. Pacaci, “Apollo: Learning query correlations for predictive caching in geo-distributed systems” in *Proceedings of the 21st International Conference on Extending Database Technology (EDBT)*. Vienna, 2018, pp. 253 – 264. https://doi.org/10.5441/002/edbt.2018.23.
11. M. Brendle, N. Weber, M. Valiyev, N. May, R. Schulze, A. Böhm and G. Moerkotte, “SAHARA: Memory footprint reduction of cloud databases with automated table partitioning” in *Proceedings of the 25th International Conference on Extending Database Technology (EDBT)*. Edinburgh, 2022, pp. 13 – 26. https://doi.org/10.5441/002/edbt.2022.02.

Пугач Микита Сергійович *Аспірант, Кафедра теоретичної та прикладної інформатики Харківського національного університету імені В.Н. Каразіна, майдан. Свободи 4, Харків, Україна, 61022*
e-mail: mykyta.pugach@karazin.ua
https://orcid.org/0009-0004-8923-6489

Систематичний огляд на виявлення змін робочого навантаження в розподілених базах даних

Розподілені бази даних стали важливою частиною значної частини сучасного програмного забезпечення. Вони мають численні переваги, включаючи масштабованість, відмовостійкість, високу доступність і покращену продуктивність. Це вирішує багато проблем централізованих баз даних, але також можуть мати проблеми. Одна з них – нерівномірний доступ до даних. Робоче навантаження в розподілених СУБД часто змінюється, такі коливання можуть стати причиною неефективної роботи системи. Уявіть, що доступ до одного рядка бази даних став у 10 разів частішим, або складні запити починають працювати з даними, розподіленими територіально. Така поведінка свідчить про те, що первинний розподіл даних не завжди може бути достатньо ефективним. І для вирішення цієї проблеми були винайдені технології адаптивного дизайну. У цій статті ми розглядаємо загальні кроки адаптивних технологій і зосереджуємо увагу на виявленні робочого навантаження та ідентифікації гарячих даних.

Метою статті є огляд адаптивного підходу до проектування розподілених систем керування базами даних, огляд і аналіз існуючих реалізацій та їхніх кроків, особливо виявлення зміни робочого навантаження та ідентифікації гарячих даних. Кінцева мета полягає в тому, щоб порівняти ці техніки та виявити їх основні проблеми.

У результаті цієї роботи було проаналізовано деякі існуючі підходи та виділено їх спільні сторони та відмінності, представлено їх основні проблеми.

Після перегляду всіх технологій ми можемо побачити, що поточні рішення не можуть дати точних результатів, не створюючи значних накладних витрат на систему. Крім того, немає підходу до надання актуальної інформації про гарячі дані без створення накладних витрат. Накладні витрати в таких ситуаціях є серйозною проблемою. У шаблонах нерівномірного доступу розподілені вузли можуть бути дуже зайняті обробкою запитів, а додаткові обчислення можуть призвести до більшого погіршення загальної продуктивності системи, ніж коли адаптивний підхід не використовується, або навіть до збою вузла. Таким чином, пошук рішень, які дають точні та своєчасні результати без значних накладних витрат, є великим полем для майбутніх досліджень.

Ключові слова: *розподілені бази даних, адаптивна підходи до проектування, ідентифікація гарячих даних, виявлення зміни робочого навантаження.*

Надійшла у першій редакції 02.04.2025, в останній - 14.05.2025.
The first version has been received on 02.04.2025, the final version - on 14.05.2025.