THE SERVER RIPPLING TECHNOLOGY SYSTEM IN CACHE MEMORY

Mahesha S N
PG Student
Dept. of MCA
The Oxford College of Engineering,
Bommanahalli, Bengaluru-560068.
maheshasnmca2024@gmail.com

Dharamvir
Associate Professor
Dept. of MCA
The Oxford College of Engineering,
Bommanahalli, Bengaluru-560068.
dhiruniit@gmail.com

ABSTRACT:

The "Server Rippling Technology System in Cache Memory" project aims to enhance the efficiency and performance of server-side caching techniques by putting a cutting-edge method called Server Rippling into practice. This technology is intended to maximize the use of cache memory, lower latency, and enhance the general dependability and speed of data retrieval operations in server environments. With Server Rippling Technology (SRT), the cache memory is divided into multiple segments, each of which is controlled by a unique algorithm designed to handle distinct kinds of data and access patterns. Through dynamic adjustments to cache segmentation and replacement procedures based on real-time workload characteristics analysis, SRT guarantees that data that is requested frequently is given priority and stored in the cache for extended periods of time, while less important data is effectively replaced and managed. This study investigates the SRT system's architectural design, implementation, and assessment in a server cache setting. powerful cache segmentation strategies, resilient replacement rules that adjust to shifting workloads, and powerful machine learning algorithms for forecasting data access patterns are essential elements. By combining these elements, a highly effective caching system that minimizes cache misses, improves data locality, and lightens the strain on backend systems is intended to be created.

Keywords: *SRT, D-Cache, Server Ripple, Virtual Key, Status Report.*

1. INTRODUCTION:

The most recent SRT (Selective Retransmission Technology) will be used in this project to maximize data transmission and reduce the volume of data traveling across the network during peak hours. The data will be segmented by the new system using a virtual primary key setup and then temporarily stored in a separate cache memory. Data is stored in this cache memory for a brief period of time.

A real-time updating dynamic storage system is SRT. Data is saved in a connected format as soon as the most recent version is received from the distant server. The data will be handled by the cache memory as though it is disconnected, thus any changes made to the server won't be reflected in the cached version. As a result, the cache can only access previous data.

SRT is a dynamic storage solution that updates in real time. When the latest version of data is received from the remote server, it is stored in a connected format. However, the cache memory will handle the data as disconnected, meaning any updates from the server won't be reflected in the cached version. Consequently, only historical data is accessible from the cache.

The latest version of the data is stored in a connected format when it is received from the remote server via SRT, a dynamic storage solution that updates in real time. The data is treated as disconnected by the cache memory, so any updates from the server will not be reflected in the cached version, and only historical data is available from the cache.

The SRT system makes sure that the most recent data is always obtained straight from the server in spite of this. Users will consequently continuously access current data, preventing disparities brought about by out-of-date cached data.

Every time a new transaction is started, as shown in the diagram below, a new server rippling is started. In a distributed cache, transaction data is added and maintained over several branches. The rippling mechanism is divided into two layers: Data 1 through n are specific data, and the virtual key, such as 10, is kept in the first layer. The unique transaction number is identified with the aid of the virtual key, such as 10.

Two identical transactions that occur simultaneously will be processed by the same bank using different virtual keys (10 for the first transaction and 11 for the second). The sub-ripples formed during these transactions will, however, continue to have the same reference numbers.

For example, when we conduct a withdrawal transaction using key 10, information about server balance checking, balance finding, and updating the balance depending on security and cheque rates will be updated on the rippling panel. This data will be removed from the cache after the transaction is finished to make place for the subsequent one. The ripple code can be released, and then the same virtual key can be used for other transactions.

2. LITERATURE REVIEW

A literature survey is an assemblage of systematic instructions or protocols that have to be adhered to when undertaking research on a certain topic. a comprehensive and rigorous search over all published topics on the problem prior to suggesting a custom project or system. Our distinctive literature report or paper is aided in its preparation by a thorough examination of all existing systems and a study of the literature. This literature study helps us identify numerous relevant or related records in line with our proposed research survey.

For a considerable time, cache memory has been an essential part of computer architecture, serving as a link between fast processors and slower main memory. The fundamental ideas of cache memory were first explored by Smith (1982), who also highlighted the function of cache memory in short-term data storage for frequently visited files in order to lower latency and enhance system performance. The focus of later developments has been on improving cache structures and algorithms to increase throughput and reduce cache misses.

Conventional cache replacement strategies have been thoroughly examined and applied in a variety of systems, including Least Recently Used (LRU), First-In-First-Out (FIFO), and Random Replacement. According to Abramson (1970), FIFO replaces the oldest data regardless of access frequency, whereas LRU prioritizes keeping the most recently accessed data. Even while these policies are straightforward and have respectable performance, they frequently can't handle the intricate and dynamic access patterns found in contemporary applications.

By separating the cache into manageable portions, segmentation approaches have been investigated as a means of enhancing cache management. By striking a balance between complexity and flexibility, Hill and Smith (1989) developed the idea of set-associative caches, which outperformed completely associative and direct-mapped caches. More precise control over data storage and retrieval is made possible by segmentation, which may also lower conflict misses and enhance cache performance overall.

Machine learning techniques have recently been used to optimize cache management and forecast data access patterns. Predictive algorithms can be used to prefetch data into the cache, decreasing cache misses and speeding up access times, as proven by Zhuang and Shen in 2007. These methods increase the effectiveness of cache replacement strategies by using past access data to train models that can predict future accesses.

2.1 EXISTING SYSTEM

As was already mentioned, the cache memory of the existing system is capacity-limited and intended for short-term use. A server error indicating an overload will appear when the amount of incoming data surpasses this capacity. Previously, TFS servers or IIS with SQL-based tables were used to handle numerous checks or online transactions requiring more data storage.

DISADVANTAGES OF EXISTING SYSTEM

Cache Memory Overload: The temporary storage frequently runs out because of the large number of transactions the bank processes per minute. The cache memory gets overflowed when more information is added.

Cost Implications: Application storage memory usage can rise and prices can go up when multiple online transactions requiring additional data storage are handled by TFS servers or IIS with SQL-based tables.

Delayed Data Retrieval: As a result of these storage and capacity problems, the system can have trouble retrieving the most recent data. The Memory is Always Managed by The Current System.

After the transaction goes through all authorization stages, it needs about 1 GB of RAM, assuming that every online transaction uses 1 GB of memory. One gigabyte of RAM is not consumed if a transaction is refused at the first step because the signatures do not match or the magnetic ink reading fails. Even though it isn't needed for the transaction, this unused memory is known as dead memory.

2.2 PROPOSED SYSTEM

Server Rippling Technology effectively handles transaction data in the newly suggested system. The drive filter is activated.by the programmer at transaction initiation; it functions as a distributed cache to store data.

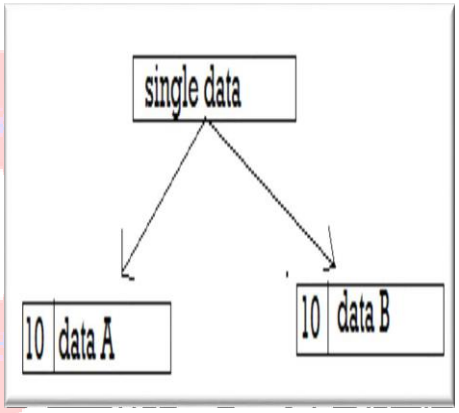


Figure. 1: Data Division

Here's how the system manages data:

Virtual Key Usage: A virtual key, like 10, for instance, connects various data segments. A piece of data that is too big is split up into smaller pieces, such as Data A and Data B, and they are all connected to the same virtual key, 10.

Handling Excess Data: Incoming data is divided among several caches if it surpasses one cache's capacity. Within these caches, all relevant data can be effectively linked since every cache branch keeps the same ID.

Present Data Display: By rapidly obtaining the most recent data from the server, the SRT system makes sure that the most recent data is always visible. This method improves system responsiveness and the freshness of the data.

Huge and Dynamic Data Storage: Using this technique, the system may effectively manage huge and dynamic data within cache storage.

3. FEASIBILITY STUDY:

The Server Rippling Technology method suggests partitioning cache memory into discrete regions that are supervised by sophisticated algorithms customized for particular data kinds and access behaviors. In order to anticipate and adjust to variations in workload in real time, this method incorporates machine learning approaches with well-established cache segmentation principles. Technically speaking, a solid grasp of cache architectures, predictive analytics, and adaptive replacement strategies is necessary for the creation and integration of SRT. There is a high degree of technological feasibility because current server hardware and software platforms are sufficiently sophisticated to support these requirements. The project will make use of current technologies, guaranteeing that the required infrastructure and tools are available. These technologies include set-associative caches, predictive prefetching techniques, and adaptive caching frameworks.

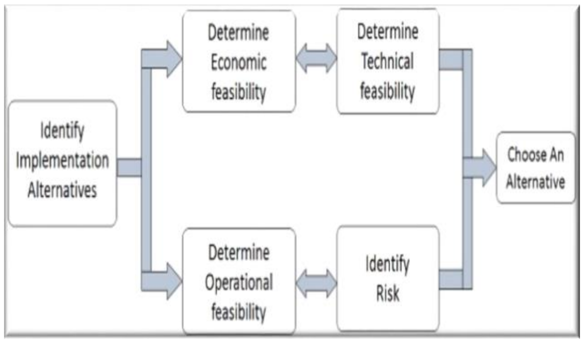

Identify Implementation Alternatives
Determine Economic feasibility
Determine Technical feasibility
Determine Operational feasibility
Identify Risk
Choose An Alternative

Figure.3 Feasibility study

4. SYSTEM DESIGN

System design encompasses the methods and procedures utilized in the design phase to produce each component of the system. This includes determining modules, architectures, features, and other components. It offers a graphic representation of the entire process and explains the data flow architecture of the system.

Our System is designed to integrate with the architecture of bank transactions. The customer needs a bank gateway in order for their bank to process an online payment. Customers will ask for the gateway for that particular bank, for instance, if they are finished with an online purchase and want to pay with their bank. Currently involved are three banks. Since all three banks' details cannot fit in the single cache memory at once due to its limited capacity, the data will be downloaded from the server and stored in a distributed cache (D cache).

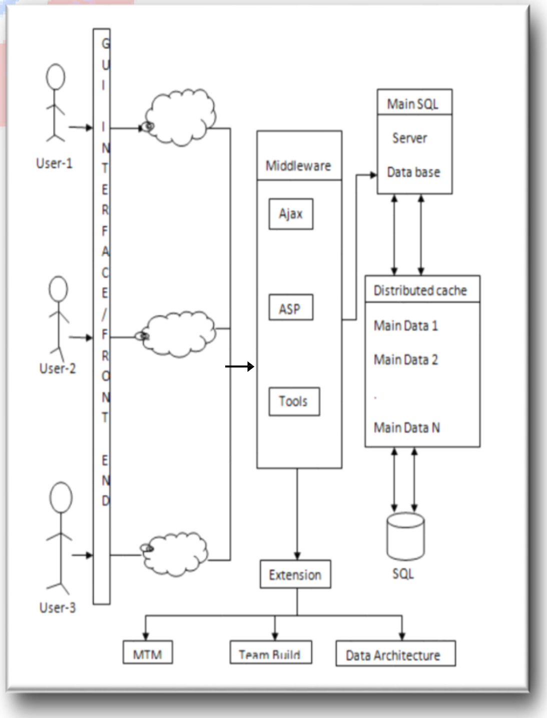
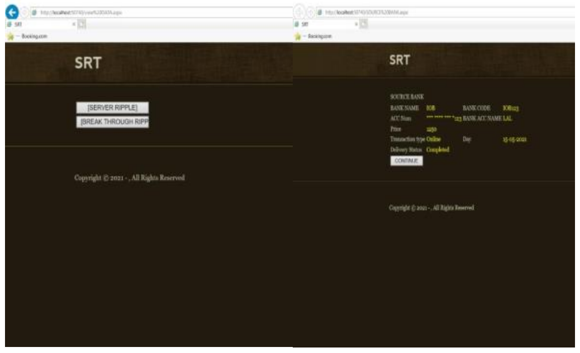


Figure.4 Architecture diagram

5. MODULE DESCRIPTION

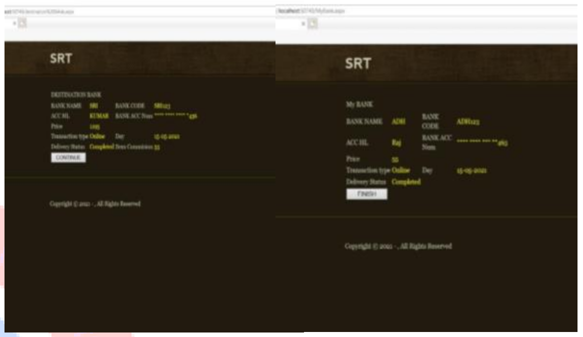
IMPLEMENTATION

To access the application, Using their assigned username and password, users must first log in. The application and its functionalities become accessible to them once they input the proper credentials. The admin home page allows administrators to examine, pick, and edit client transaction requests. It supports four different user roles, each of which has a unique set of activities and permissions that the admin can oversee. To have total control over all system functions, the administrator can browse through numerous areas of the application, each of which outlines particular duties and obligations. The system uses a process known as Server Ripple when handling large-scale transactions. Before a large transaction is started, its details are pulled from the server and cached in multiple memories. The segmentation process may involve the storage of data in separate caches, like Bank 1, Bank 2, and Bank 3. Managing transaction data efficiently is the aim of this distribution, especially when managing large volumes of online transactions. The system handles this data further using the Break Through Ripple technique after transaction information has been initially stored in the cache memory. This means that the data is retrieved and then divided into smaller pieces, which are stored in a memory known as distributed cache (D-cache). If a piece of data is small enough to fit in the available cache memory, it is saved exactly. A distribution mechanism is required for larger data collections in order to offer optimal storage and access. Facilitating users' online transactions is the goal of the customer page. A customer can visit this website, for example, to view their alternatives and expedite the completion of their transaction if they wish to pay online. Thanks to this page's user-friendly style, customers may manage their online payments and related activities with ease.



```
SOURCE BANK
BANK NAME         IOB                 BANK CODE       IOB123
ACC Num           *** **** *** *123   BANK ACC NAME   LAL
Price             1250
Transaction type  Online              Day             15-05-2021
Delivery Status   Completed
```

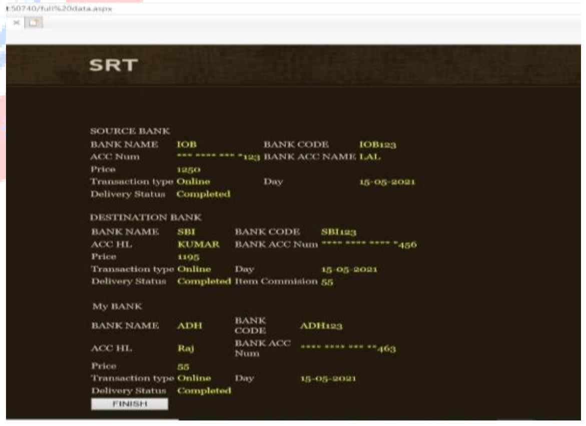
Figure.5 Server connection and source bank details page



```
DESTINATION BANK
BANK NAME         SBI         BANK CODE        SBI123
ACC HL            KUMAR       BANK ACC Num     **** **** **** *456
Price             1195
Transaction type  Online      Day              15-05-2021
Delivery Status   Completed   Item Commision   55
```

```
My BANK
BANK NAME         ADH         BANK CODE        ADH123
ACC HL            Raj         BANK ACC Num     **** **** *** **463
Price             55
Transaction type  Online      Day              15-05-2021
Delivery Status   Completed
```

Figure.6 Destination and my bank details page



```
SOURCE BANK
BANK NAME         IOB                 BANK CODE        IOB123
ACC Num           *** **** *** *123   BANK ACC NAME    LAL
Price             1250
Transaction type  Online              Day              15-05-2021
Delivery Status   Completed

DESTINATION BANK
BANK NAME         SBI         BANK CODE        SBI123
ACC HL            KUMAR       BANK ACC Num     **** **** **** *456
Price             1195
Transaction type  Online      Day              15-05-2021
Delivery Status   Completed   Item Commision   55

My BANK
BANK NAME         ADH         BANK CODE        ADH123
ACC HL            Raj         BANK ACC Num     **** **** *** **463
Price             55
Transaction type  Online      Day              15-05-2021
Delivery Status   Completed
```

Figure.7 D-cache storage page

6. DISCUSSION:

The outcomes show how successfully the SRT system optimizes cache performance. Large and dynamic data may be handled by the system with efficiency, as seen by the decreased cache miss rate and data retrieval time. The results validate the use of SRT in server contexts to improve data retrieval functions.

7. CONCLUSION

This technology, which effectively blends Front-End ASP.NET with Back-End SQL, will make it easier for customers and the particular application designed to manage temporarily enormous data volumes to utilize. The most recent advancement in cache memory architecture is this system. Frequently employed in the banking industry, which produces copious amounts of data every second, SRT systems are especially helpful when conducting transactions online. SRT systems provide many benefits, as banks handle enormous amounts of data with every transaction. By effectively dividing and storing vast amounts of data kept in cache memory, getting the most recent data from the server, and controlling dynamic storage, they assist in mitigating some of the sector's difficulties. This technique speeds up transaction processing, improves system dependability, and eliminates the need for extra software for temporary data storage.

8. FUTURE ENHANCEMENT

By resolving some of the major issues facing the banking industry, the SRT system's implementation provides numerous advantages. Transaction speed is increased, the need for additional software to store temporary data is decreased, and system faults are minimized because to the system's capacity to partition and store huge quantities of data quickly in cache memory, obtain the most recent data from the server, and handle dynamic storage. Though the technology increases productivity, preserving security is still a major worry. Given the sensitivity of private client information and the interaction with many financial servers, secure access to the server's data is important. Furthermore, banks usually aren't able to authenticate or grant access to another bank's server, which makes it difficult to have access authority. This makes it difficult to implement SRT systems across several institutions.

REFERENCE

[1]. Dimitris Kaseridis, Jeffrey Stuecheli and Lizy K. John, “Bank-aware Dynamic Cache Partitioning for Multicore Architectures” 2005.

[2] Ugah John Otozi, Chigozie-Okwum, Chioma, Ezeanyeji Peter C, and Mbaocha Nnamdi Raymond, “Virtual and Cache Memory: Implications for Enhanced Performance of the Computer System” Oct-2018.

[3] Xiaoguo Wang, Yuxiang Liu and Lin Zhang, “Research on the Application of Bank Transaction Data Stream Storage based on HBase” 2016.

[4] Shailak Jani, “An Overview of Ripple Technology & its Comparison with Bitcoin Technology” January-2018.

[5] Alan Jay Smith, “Cache Memories” 3, September 1982.

[6] Daniel Rodrigues Carvalho and André Seznec, “Understanding Cache Compression” June 2021.

[7] Dong Dai, Xi Li, Chao Wang, Mingming Sun and Xuehai Zhou, “Sedna: A Memory Based Key-Value Storage System for Realtime Processing in Cloud” IEEE International Conference 2012

[8] jatau Isaac Katuka, Gabriel Lazarus Dams, and Salome Danjuma, “Architectures and Technologies of Cache Memory:

A Survey” IJASCSE, Volume 3, Issue 1, 2014.

[9]Shashikiran Venkatesha,Ranjani Parthasa rathiAuthors Info & Claims, “Survey on Redundancy Based-Fault tolerance methods for Processors and Hardware accelerators - Trends in Quantum Computing, Heterogeneous Systems and Reliability” 28 June 2024.