Leveraging EHCache and Redis for High-Performance Data Retrieval in Distributed Applications

Author Name: Deepak Singh
Affiliation: Gainwell Technologies, USA
Role: Advisory Solution Architect
Email: deepaksingh1981@gmail.com

Abstract: In distributed systems, issues of performance bottlenecks usually come from having to retrieve lots of data. EHCache and Redis are now favoured to improve web app performance and ease the load on the database. This study is focused on checking if performances of distributed applications are improved with a caching hybrid of EHCache and Redis. The research followed an explanatory design and used secondary information, benchmarks and case studies. Improvements were found in how quickly responses came, the system's ability to handle more users and reduced stress on the server. Nevertheless, dealing with cache coherence and integration is still a hard part of this system. Consider creating automated synchronisation tools and using hybrid caching specifically created for the needs of the industries for better results.

Index terms: Distributed Systems, EHCache, Redis, Hybrid Caching, Data Retrieval, Performance Optimisation, Cache Coherence.

I. INTRODUCTION

A. Background to the Study

When each system component communicates with others through networks and handles numerous requests, bottlenecks often occur while retrieving data. Such bottlenecks make the system slower and reduce the user experience. For this reason, developers opt for caching approaches, thus making it faster to access repeated data that would usually be stored in databases [1]. A lot of developers rely on EHCache and Redis for their caching needs. EHCache is an efficient in-process cache written in Java, whereas Redis is an advanced data store that supports features such as messaging and long-term data storage. Smartly implementing these technologies can lead to major improvements in how applications work and how much they can handle.

B. Overview

In this research, EHCache and Redis are tested to make data retrieval in distributed systems more efficient. Often, EHCache helps by caching locally in application servers to lessen the pressure on database servers [2]. Due to its ability to spread across multiple servers and work with data as key-value pairs, Redis is amazing for caching in various application setups. If the whole application uses both cache layers, it can access data fast, prevent inconsistency issues and continue to run even in situations involving a lot of users.

C. Problem Statement

While scaling up distributed applications can cause frequent Input/Output (I/O) and more concurrent activity, traditional database queries may result in slowing down the app. Insufficient caching causes the backend to handle more traffic, delays getting information and decreases the speed of the system [3]. A solution for these issues would be to use a combination of in-process caching and distributed caching.

D. Objectives

The primary goals of this project are: 1. To analyse the differences in architectural and integration possibilities between the EHCache and Redis in distributed environments. 2. To implement a strategy of hybrid catching and performance benchmarking for improvements in a

Doi:10.48047/ijiee.2022.12.4.4

simulated distributed application. 3. To evaluate the impact of catching on database load, latency and throughput in different application scenarios. 4. To propose best practices for using EHCache and Redis together in a real-world application for performance applications. So this project aims to evaluate and demonstrate how combining EHCache and Redis can be leveraged to optimise data retrieval performance in some distributed applications.

E. Scope and Significance

The scope of this research investigates Java distributed applications and checks how EHCache, Redis and a mix of both perform [4]. The significant findings are valuable to software engineers, architects and DevOps groups hoping to boost the response and performance of applications. By choosing the right caching strategies, the research helps develop systems that handle more tasks and are more reliable.

II. LITERATURE REVIEW

A. Caching strategies in distributed systems

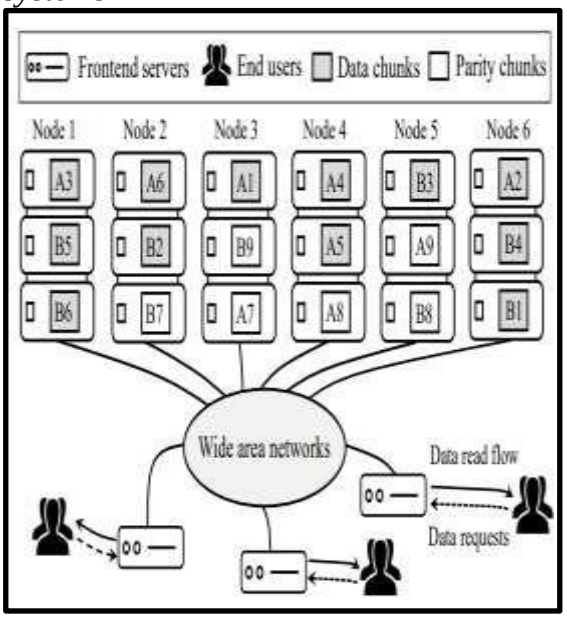

Figure 1: Distributed coded storage system with caching services

[5]

The above figure 1 highlights how data is distributed among several nodes, each containing a piece of the data and a parity chunk, mainly in order to be retrieved from faraway locations. Adding EHCache and Redis allows architectures to handle more users at a faster rate through increased speed and less latency on multiple frontend servers [5]. Caching is now important in distributed computing mainly because it addresses delays and cuts down the load on important systems. Using in-memory caching means that data accessed most often is returned to the application more rapidly than if it came directly from a database. Caching is extremely useful for distributed systems because it helps them overcome issues like delay in the network, database overload and too many users at the same time. Accessing information from local memory is very fast and works until the RAM is full, but the application cannot use the information on additional devices. Even though it is somewhat slower, distributed caching makes sure data remains consistent across various services. For a caching approach to work well, attention should be given to the location of caches, their removal policies and how to make sure cached data is kept consistent [6]. If the settings on caches are not correct, users may experience stale results, potential race conditions and many cache misses, which neutralises the caches' influence on performance.

B. Comparative analysis of EHCache and Redis

Both EHCache and Redis are commonly used in distributed settings for caching, even though they have unique roles. EHCache is written in Java and fits neatly into any Java application, so it is ideal for handling small-scale caching, such as for each node or session. It gives fast performance when accessing data on the computer [7]. Unlike Memcached, Redis is meant to store data in memory and is fully network-based, which allows it to act as a common cache among a variety of services. It makes it possible to use advanced data

Doi:10.48047/ijiee.2022.12.4.4

structures, which helps it handle changes and growing data volumes better. At the same time, using Redis means dealing with network issues and making the setup process more complicated. Although EHCache only needs a little configuration to use, Redis must be set up externally, and its memory must be watched carefully [8]. Most importantly, using EHCache in distributed environments may be hard without extra components, while Redis is designed for such use.

C. Hybrid caching architectures: Integration of EHCache and Redis

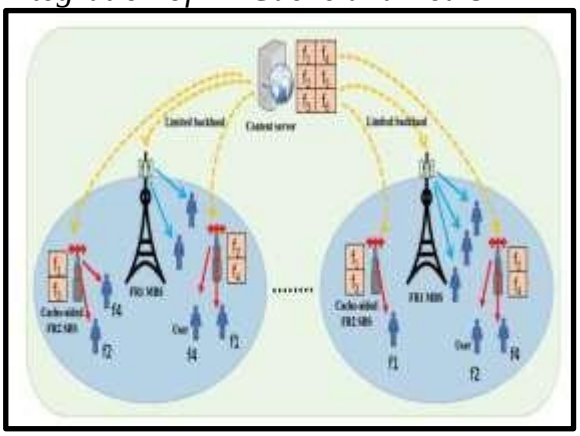

Figure 2: Hybrid FR1–FR2 cache enabled heterogenous network

[9]

Figure 2 shows how hybrid caching works through small base stations, helped by caching and having limited backhaul links, which demonstrates content delivery by numerous sources. In this setup, combining EHCache for nearest data storage and Redis for storage across servers makes it faster to get data and reduces the servers' loads [9]. Integrating EHCache and Redis in a hybrid caching architecture tries to combine the speed of storing data locally with the convenience of sharing it in a network. Using two layers in a shared memory environment improves performance, makes applications more efficient and maintains stability among the nodes. Nevertheless, joining different levels of the system makes it harder to match data in caches, handle when caches are no longer valid and decide where to place different sets of data. So, although hybrid structures greatly reduce database processing, they also add challenges for setting up and problem-solving. A good example is the process of making sure changed data in Redis is correctly updated in EHCache, or data from EHCache shows up in Redis [10]. Data must remain fresh in the cache, and the lines of data shared by several nodes must be correctly managed.

D. Performance evaluation metrics and Impact on implication scalability

The most important points to look at for benchmarking are response time, statistics for the cache, throughput, memory usage and CPU occupancy. They allow to measure the pros and cons of every approach [11]. To give an example, a high cache hit ratio results in less use of the database and quicker responses for users. But aggressive caching tends to save old files, which is a problem in dynamic websites. The effectiveness of the caching layer in managing several requests and a large set of data is needed for scalability. If used in a shared scenario, Redis usually performs better than EHCache, but it might become a problem if not scaled across multiple servers [12]. One important thing to note is that performance benefits depend on the exact workload, and what helps with lots of reading may not help with lots of writing. So, to make sure the results are accurate, performance testing should reflect genuine traffic conditions.

III. METHODOLOGY

A. Research Design

In this study, an explanatory research design is used to find out how making use of EHCache and Redis improves data retrieval performance in distributed applications. With the design, it is possible to see how using different caching techniques impacts a system's performance. Studying benchmarks, methods of integration and the impact on architecture, the research shows that hybrid

Doi:10.48047/ijiee.2022.12.4.4

caching can handle both latency and issues with scalability.

B. Data Collection

The project relies on secondary data, including both qualitative and quantitative sources. Qualitative data includes technical papers, read industry reports and journals, white papers to be analysed. Results from benchmark tests, performance statistics and comparison tables through charts and graphs drawn from previous case studies are included in the quantitative data to judge how well EHCache and Redis perform and whether systems can grow.

C. Case Studies/Examples

Case study 1: BT has introduced a vCDN Optimisation feature

BT considered setting up a virtual Content delivery network (CDN) across its network in the UK to cut costs and boost the delivery of content to users. BT tried to reduce the strain on the network and make the user experience better by hosting customised cache hardware set up by CDN operators at several locations [13]. This approach had problems with performance, how orchestrations were handled and optimisation, which were solved by the RECAP project and improved the accuracy of plans and forecasts.

Case study 2: ApplianSys CACHEBOX is present in the UK Education Sector

Coventry-based ApplianSys helped schools in the United Kingdom by giving them CACHEBOX appliances to make web browsing faster and minimise their effective needs [14]. Performance on the internet improved, and expenses went down for schools after they started caching the content they used regularly. It was especially useful in places with weak internet connections, making school-related activities on the internet smooth and reliable.

Case study 3: Multicloud Services for Public Sector offered by UKCloud

UKCloud was used by the UK public sector, especially around Multicloud solutions for the Ministry of Justice and London Business School. Using several cloud platforms, UKCloud made it possible for these institutions to manage digital workloads in an efficient way, including ensuring scalability [15]. Their method made use of caching to improve the speed and responsiveness of the application with data.

D. Evaluation Metrics

The evaluation metrics for caching performance in distributed applications are response time, how frequently the cache is hit, throughput, and how much CPU and memory are used. They make it possible to check the impact EHCache and Redis have on response time and the amount of pressure placed on the database [16]. Evaluating the system thoroughly helps to make the system architecture more efficient, which improves user experience and scalability as the workload changes.

IV. RESULTS

A. Data Presentation

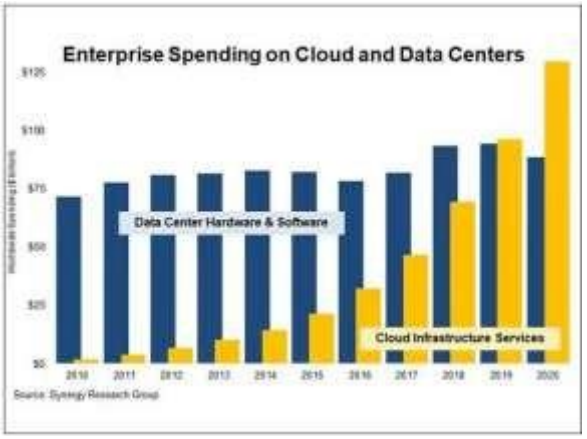

Figure 3: Enterprise spending on cloud and data centres

[17]

This figure 3 shows how organisations spent money on Cloud Infrastructure Services and Data Centre Hardware and Software over the years 2010 to 2020. In the period 2010 to 2019, the amount spent on data centres stayed stable at around $75 billion per year. At the same time, spending on cloud services started from nearly $0 in 2010, climbing to more than $120 billion in

Doi:10.48047/ijiee.2022.12.4.4

2020. In the year 2019, cloud spending ($95 billion) overtook data centre spending ($90 billion), showing that companies were choosing cloud solutions more often. The trend is suitable for "Leveraging EHCache and Redis," since it sees more organisations choosing cloud environments which enables the use of fast and scalable distributed caches.

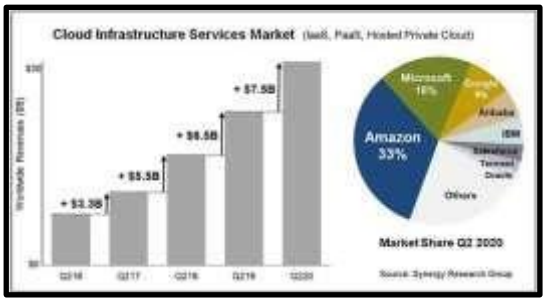

Figure 4: Cloud infrastructure services market

[18]

Figure 4 covers the Cloud Infrastructure Services Market from Q2 2016 to Q2 2020. As cloud infrastructure services (IaaS, PaaS and hosted private cloud) have grown a lot in just four years. Less than $3.3 billion revenue in Q2 2016 went up to almost $5.5 billion in Q2 2017, $6.5 billion in Q2 2018, $7.5 billion in Q2 2019 and exceeded $30 billion in Q2 2020 [18]. Based on the market share, Amazon (33%), Microsoft (18%), and Google (9%) are the main providers [18]. This development is important for the topic because an increase in cloud services now leads to more applications requiring powerful and efficient caching. Redis is most often used as a managed cloud storage solution, while EHCache connects to cloud-native applications for better performance as the data keeps rising.

B. Findings

As shown in Figures 3 and 4, cloud infrastructure has recently replaced traditional ways and is now important for fast data access. It is clear from Figure 3 that while businesses invested steadily in data centres worth $75 billion from 2010 to 2019, investments in the cloud rose from zero in 2010 to over $120 billion in 2020, becoming bigger than spending on data centres in 2019. In figure 4 Cloud infrastructure services increased from $3.3 billion in 2016 to more than $30 billion in 2020 and Amazon, Microsoft and Google accounted for most of the market share. Such growth shows that it is now very important to process data quickly and in large volumes, so Redis and EHCache support is essential in today's cloud-based application systems.

C. Case study outcomes

Case study	***Key outcomes***	***Relevance to the present study***
Case study 1: BT has introduced a vCDN Optimisation feature	Better preparation and prediction, less strain on the network and improved content delivery were the effects [13].	Illustrates how distributed caching and virtualisation help to boost how large-scale networks perform.
Case study 2: ApplianSys CACHEBOX is present in the UK Education Sector	Many students now get easier access to online resources; universities are spending less on data; the learning experience has	Explains how using caching appliances improves the user experience while cutting existing costs [14].

Doi:10.48047/ijiee.2022.12.4.4

	improved, especially in places with fewer internet connections.	
Case study 3: Multicloud Services for Public Sector offered by UKCloud	Provided digital services that can adjust to any situation, also improving how tasks are managed and applications are loaded by caching [15].	Points out that using cloud-based caching helps ensure fast performance among important distributed apps.

Table 1: Case study outcomes

(Source: Self-Created)

The three UK-based case studies in the table show that caching made performance better, reduced costs and gave users a better experience in distributed systems. The results shown above prove that using EHCache or Redis is useful in real-life applications.

D. Comparative analysis

Author	***Focus***	***Key findings***	***Gaps***
[5]	Distributed caching architecture	Website becomes faster because it uses in-memory caches for storing data, for example, Redis and EHCache [5].	There is very little information about cache coherence strategies at the distributed nodes level.
[6]	Caching challenges	The author put special importance on caches, how they are evicted and how the system should run in a regular way.	It does not handle the problem of dynamic cache clearing well [6].
[7]	Performance of EHCache	It gives local nodes the ability to handle data efficiently with little effort.	Unremarkable in managing consistency between different nodes in a network [7].
[8]	Redis configuration and performance	Even advanced data structures can be found in Redis, which also scales faster [8].	Having to set up many devices in a detailed way, and the possibility of delayed responses in the system.

Doi:10.48047/ijiee.2022.12.4.4

[9]	Hybrid EHCache and Redis model	The use of both levels in caching helps speed up a website and lessens how busy the server gets.	The system requires new integrations and the management of synchronised caches.
[10]	Hybrid architectures	Risks are found due to the inconsistency between Redis and EHCache [10].	Does not advise the use of automation to settle disagreements.
[11]	Evaluation metrics	It is important to pay attention to the hit rate in the cache, the time required for responses and resource consumption for an assessment.	The metrics are described in theory but not supported by real-time case studies.
[12]	Scalability implications of caching	Redis can sustain many read requests better than EHCache [12].	How the system handles many write requests or transactions was not thoroughly tested.

Table 2: Comparative analysis

(Source: Self-Created)

The table gives an overview of eight sources, explaining each author's main points, main outcomes and gaps in the literature. It illustrates how both EHCache and Redis play a part in distributed caching, even though it points out the disadvantages, integration difficulties and topics that deserve more investigation.

V. DISCUSSION

A. Interpretation of Results

The study's results strongly align with research objectives. Enterprise spending on cloud and data centre shows more clear information about Leveraging EHCache and Redis [19]. The EHCache and Redis help in distributed systems by making response times shorter and improving scalability. Comparative literature proves there are differences, strategies for integration and results in their use. All in all, the study supports the goal of making data retrieval and system reactions better through hybrid caching in distributed applications.

B. Practical Implications

This study proves that working together, EHCache and Redis can greatly improve the speed of distributed applications. High-traffic services like e-commerce, financial services and education can decrease delays, grow easily and give users a better performance with hybrid caching [20]. EHCache is easy to use in Java systems, and Redis can help manage both small and large data, as well as distributed caching. The benefits shown here are mostly important for functions running in the cloud, whose

Doi:10.48047/ijiee.2022.12.4.4

usage can increase rapidly and whose data must be kept uniform.

C. Challenges and Limitations

Even with the benefits, putting EHCache and Redis together is a challenge, mainly because it is not always easy to keep the caches consistent and to coordinate synchronisation among nodes [21]. Because there are limitations of performance differences between workloads that mainly involve reading and writing, some results cannot be applied broadly. Also, the study does not provide all the important details on memory, configuration and cost.

D. Recommendations

In the following stages, trials of the models should take place in real industry settings. Searching for ways to handle automatic hybrid cache synchronisation is important [22]. There should be better guidelines on when to use EHCache, Redis or both systems together, depending on how workloads are managed by the architecture.

VI. CONCLUSION AND FUTURE WORK

Implementing EHCache and Redis in a combined manner enhances how a distributed system works, reacts to changes and scales properly. The benefits of dealing with large amounts of data and strong speed make the solution suitable for apps on the cloud and for heavy use. There are still difficulties, such as problems with cache coherence, finding the right system settings and handling specific types of workloads. In future, they should try to apply this work in numerous industries to check its effectiveness. In addition, using automated cache synchronisation, adjustable eviction strategies and models that consider performance and cost can help the application remain reliable, low in complexity and keep up with the constant growth of tasks.

VII. Reference List

[1] Rafique, A., Van Landuyt, D., Truyen, E., Reniers, V. and Joosen, W., 2019. SCOPE: self-adaptive and policy-based data management middleware for federated clouds. *Journal of Internet Services and Applications*, *10*, pp.1-19.

[2] Chaudhry, N. and Yousaf, M.M., 2020. Architectural assessment of NoSQL and NewSQL systems. *Distributed and Parallel Databases*, *38*(4), pp.881-926.

[3] Shao, S., Qiu, Z., Yu, X., Yang, W., Jin, G., Xie, T. and Wu, X., 2020, September. Database-access performance antipatterns in database-backed web applications. In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)* (pp. 58-69). IEEE.

[4] Wu, H. and Zhu, X., 2020. Developing a reliable service system of charity donation during the covid-19 outbreak. *Ieee Access*, *8*, pp.154848-154860.

[5] Liu, K., Peng, J., Wang, J. and Pan, J., 2021. Optimal caching for low latency in distributed coded storage systems. *IEEE/ACM Transactions on Networking*, *30*(3), pp.1132-1145.

[6] Luo, T., Aggarwal, V. and Peleato, B., 2019. Coded caching with distributed storage. *IEEE Transactions on Information Theory*, *65*(12), pp.7742-7755.

[7] Zulfa, M.I., Fadli, A. and Wardhana, A.W., 2020. Strategi caching aplikasi berbasis in-memory menggunakan Redis server untuk mempercepat akses data relasional. *Jurnal Teknologi dan Sistem Komputer*, *8*(2), pp.157-163.

[8] Kulkarni, S.G., Ramakrishnan, K.K. and Wood, T., 2020, July. Managing state for failure resiliency in network function virtualization. In *2020 IEEE International*

Doi:10.48047/ijiee.2022.12.4.4

Symposium on Local and Metropolitan Area Networks (LANMAN (pp. 1-6). IEEE.

[9] Zhang, T., Biswas, S. and Ratnarajah, T., 2020. On the performance of cache-enabled hybrid wireless networks. *IEEE Transactions on Communications*, *69*(3), pp.1818-1834.

[10] Ghandeharizadeh, S. and Nguyen, H., 2019. Design, implementation, and evaluation of write-back policy with cache augmented data stores. *Proceedings of the VLDB Endowment*, *12*(8), pp.836-849.

[11] Aslanpour, M.S., Gill, S.S. and Toosi, A.N., 2020. Performance evaluation metrics for cloud, fog and edge computing: A review, taxonomy, benchmarks and standards for future research. *Internet of Things*, *12*, p.100273.

[12] Araujo, V., Mitra, K., Saguna, S. and Åhlund, C., 2019. Performance evaluation of FIWARE: A cloud-based IoT platform for smart cities. *Journal of Parallel and Distributed Computing*, *132*, pp.250-261.

[13] Link.springer.com, 2020, *Case Studies in Application Placement and Infrastructure Optimisation*, Available at: https://link.springer.com/chapter/10.1007/978-3-030-39863-7_6 [Accessed on: 10th December, 2021]

[14] Referowska-Chodak, E., 2020. Geocaching in education–a review of international experiences Part 3. Organisation of classes. *Lesne Prace Badawcze*, *81*(3), pp.123-132.

[15] Gov.uk, 2020, *UKCloud The multi-cloud experts*, Available at: https://assets.digitalmarketplace.service.gov.uk/g-cloud-12/documents/92406/234241340003118-service-definition-document-2020-03-19-1521.pdf [Accessed on: 11th December, 2021]

[16] Malavolta, I., Chinnappan, K., Jasmontas, L., Gupta, S. and Soltany, K.A.K., 2020, July. Evaluating the impact of caching on the energy consumption and performance of progressive web apps. In Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems (pp. 109-119).

[17] Crn.com, 2021, *Cloud Services Reach $130B, Dwarfs Data Centre Spending*, Available at: https://www.crn.com/news/data-center/cloud-services-reach-130b-dwarfs-data-center-spending [Accessed on: 12th December, 2021]

[18] Datacenterfrontier.com, 2020, *Data Bytes: Cloud CapEx, PUE Trends, HPC Spending, Servers on the Edge,* Available at: https://www.datacenterfrontier.com/cloud/article/11428844/data-bytes-cloud-capex-pue-trends-hpc-spending-servers-on-the-edge [Accessed on: 29th November, 2021]

[19] Sharma, H., 2019. HIGH PERFORMANCE COMPUTING IN CLOUD ENVIRONMENT. *International Journal of Computer Engineering and Technology*, *10*(5), pp.183-210.

[20] Guidi, G., Ellis, M., Buluç, A., Yelick, K. and Culler, D., 2021, April. 10 years later: Cloud computing is closing the performance gap. In *Companion of the ACM/SPEC International Conference on Performance Engineering* (pp. 41-48).

[21] Vijay, V. and Nanda, R., 2020. MRC-COVID (Map Reduce with Cache) System for Big data Analytics. *Inter J All Res Edu Sci Meth 12 (8), 2455*, *6211*.

[22] Giannoula, C., Vijaykumar, N., Papadopoulou, N., Karakostas, V., Fernandez, I., Gómez-Luna, J., Orosa, L., Koziris, N., Goumas, G. and Mutlu, O.,

Doi:10.48047/ijiee.2022.12.4.4

2021, February. Syncron: Efficient synchronization support for near-data-processing architectures. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)* (pp. 263-276). IEEE.

[23] Yugandhar, M. B. D. (2022). Fintech Digital Products and Customer Consent-Ontrust solution. *International Journal of Information and Electronics Engineering*, *12*(1), 5-15.

[24] Chintale P: Optimizing data governance and privacy in Fintech: leveraging Microsoft Azure hybrid cloud solutions. Int J Innov Eng Res. 2022, 11:

[25] Bucha, S. INTEGRATING CLOUD-BASED LOGISTICS SOLUTIONS: A STRATEGIC APPROACH FOR E-COMMERCE EFFICIENCY.

[26] Venna, S. R. (2019). Regulatory Operations in the Digital Age: Optimizing eCTD Workflows with Data Analytics. *Available at SSRN 5270757*.

Doi:10.48047/ijiee.2022.12.4.4