

Towards Resource-Efficient Cloud Computing: A Pre-Fetching-Based Optimization Model

Ali Fayez Alkoradees^{1*}

¹Unit of Scientific Research, Applied College, Qassim University, Buraydah, Saudi Arabia.

*Corresponding Author: Ali Fayez Alkoradees. Email: alifk@qu.edu.sa

Received: June 18, 2026 Accepted: August 26, 2026

Abstract: When operating on a large number of computing nodes on large network infrastructures with vast amounts of data, it is essential to have the management tools provided by a cloud computing ecosystem. The core objectives of this project are to address key aspects such as availability, and performance. The ultimate aim is to build a robust system using advanced procedure and algorithms, enabling a comprehensive solution for the vendors and in accordance with the basic expectations of cloud computing providers. The key aim of this proposal is to optimize cloud storage and retrieval systems and explore them in detail. Specifically, to investigate the use of C-Aware: A technique which is designed to enhance retrieval efficiency and performance of data. Using empirical analysis, also compare the performance of C-Aware with traditional Least Recently Used (LRU) methods and investigate the influence of network load and disk speed on the access times. The C-Aware approach also leads to a considerable decrease of access times and a general improvement of the system performance, especially when the amount of data increases and more load is added to the network. In the evaluated workloads, the proposed pre-fetching approach successfully lowered the latency to 55 – 65 ms compared to 110 – 150 ms in the non-optimized case, and was able to increase the cache hit ratio to 83%, while also lowering energy consumption to 2.0 J/request (2.8 J/request peak). This project is a very valuable contribution to the field of cloud computing and gives a solution that can be used in the very high standard of performance expected by modern vendors.

Keywords: Cloud Computing; Data Storage; Data Retrieval; Performance Optimization

1. Introduction

The new paradigm of providing on-demand computing capabilities is becoming scalable, elastic and reductionist, known as Cloud Computing. In cloud setups, an efficient storage and retrieval mechanism for data is now a serious constraint because of the current explosion of data volumes. Inefficient data access latency and inefficient input/output (I/O) throughputs directly impact service-level agreements (SLAs) and the user experience as well as the cost of cloud infrastructures [1]. Although distributed storage systems, including Google File System (GFS), Hadoop distributed file System (HDFS) and Amazon S3 support large scale and fault tolerance, they come with major limitations in curbing access latency, especially when faced with data intensive and latency-sensitive applications [2].

Different optimization techniques have also been presented to curb these challenges, such as caching, replication and data placement. Of these, pre-fetching has demonstrated a lot of potential. Prefetching itself is based on the predictive models, which predict future data access and load data into the memory or into the cache in advance [3]. Limiting I/O by overlapping computation and I/O operations, pre-fetching reduces

perceived latency and increases throughput. Despite much research on both pre-fetching in operating systems, web caches and database query optimization, little has been done to apply the concept in large scale, distributed heterogeneous storage systems in cloud [4]. The major challenges are accurate prediction of the future, cost control and the ability to respond to unpredictable cloud workloads. The hybrid cloud model is highly deployed in the cases where workload variations are highly significant and in cloud bursting cases. The application areas are typical of e-business, e-commerce, e-science, online gaming, content delivery networks (CDNs) and social networking as depicted in Fig. 1 [5].

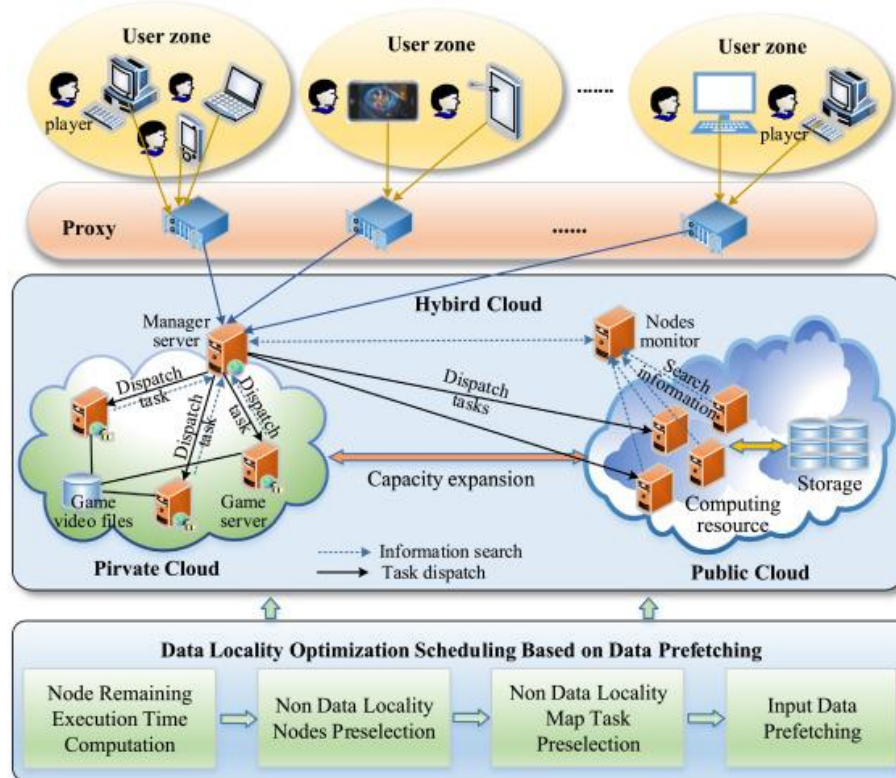


Figure 1. An illustration example of a hybrid cloud mode

The recent developments in machine learning-based predictive modelling, Markov chain techniques and access pattern analysis have opened a door to optimise pre-fetching in cloud storage [6]. The techniques help systems to acquire temporal and spatial relationships of data access, thus improving the accuracy of the pre-fetching decisions. In addition, the combination of pre-fetching and distributed file systems, object storage platforms and virtualization further allow a means of achieving a large injection of performance improvement. Such metrics as average response time, cache hit ratio, throughput (MB/s), energy efficiency could be used as benchmarks to gauge such improvements.

The research which is proposed is aimed at developing a systematic model that optimises pre-fetching in cloud data storage and retrieval ensuring three main objectives are answered: (i) developing predictive algorithms that can enhance pre-fetching in dynamic and distributed clouds, (ii) evaluating trade-offs between pre-fetching overheads and performance, and (iii) integrating pre-fetching mechanisms and existing cloud storage systems to achieve greater scalability and fault tolerance. By focusing on these objectives, this research will add to the theory and practice of pre-fetching in the context of cloud computing, providing a way forward in the field of efficient, reliable and adaptive data management system.

The rest of the paper is presented as follows. The recent approaches in this area will be discussed in Section II. Then, the proposed approach is formulated in Section III. Section IV shows the numerical results of our proposed approach. Finally, Section V shows the conclusion.

2. Literature Review

In this section of our paper, the recent approaches in the literature will be discussed. This section is divided into several subsections, in order to present a clear taxonomy for the literature review. Fig. 2. shows the taxonomy that has been followed in this study.

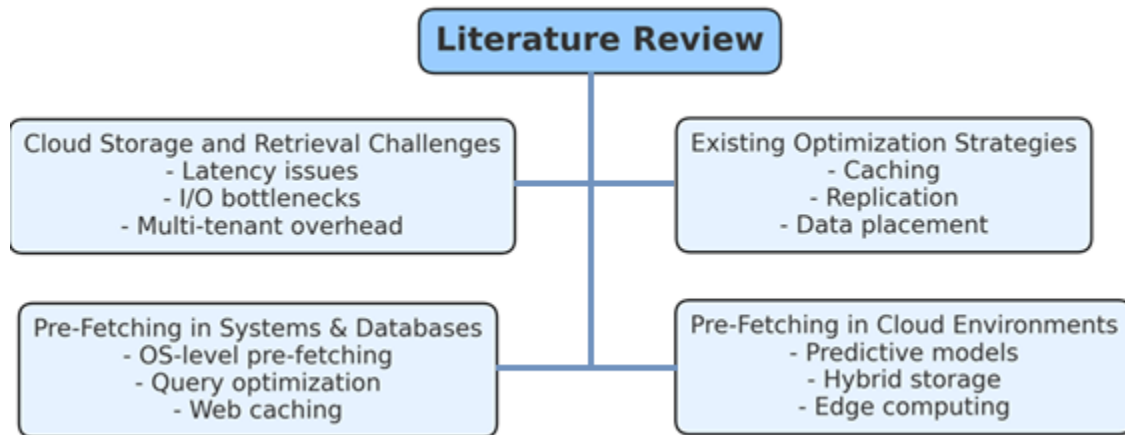


Figure 2. The Literature Review taxonomy

A. Some of the challenges faced in cloud storage and retrieval are:

Online storage systems like the Hadoop distributed file system (HDFS), Amazon S3, and Google cloud storage have now become part and parcel of large-scale data management. These systems have been modelled to provide reliability (fault-tolerance), durability, and scalability in distributed systems [7]. Nonetheless, despite the above benefits, there are issues relating to the latency and throughput demands of various workloads. Retrieval delays have been found to be largely due to I/O bottlenecks, network congestion and resource allocation wastefulness in multi-tenant cloud environments [8]. Conventional storage processes concentrate on data consistency and data replication, but they do not always pay attention to real-time responsiveness, which is a crucial target of latency-sensitive applications like financial analytics, e-health monitoring and artificial intelligence-based services [9].

B. Current Optimization Approaches of Cloud Storage

Researchers have considered many ways to overcome these inefficiencies. The most popular method used to eliminate redundant access of data is caching, with in-memory interfaces like Memcached and Redis particularly popular for frequently accessed data [10]. Repetition techniques increase reliability and availability at a higher storage cost. Data placement processes have also been proposed to further reduce access time that aligns data storage units with computing units by maximizing the data locality. Although such methods result in improved performance, they tend to be reactive and unable to look at the data requirements in the future [11].

C. Computer systems and databases Pre-Fetching

The idea of pre-fetching has been used in conventional computing systems, i.e. in operating systems, database management systems and web caching infrastructures [12]. Operating systems make use of pre-fetching algorithms on a sequential and stride basis to shorten disk access latency. Predictive query optimization and access pattern mining are exploited in databases in order to retrieve relevant tuples prior to query execution [13]. Empirically, the web browsers are found to engage in a pre-fetch mechanism to preload linked resources prior to any navigation. All of these implementations show that proactive data retrieval can greatly lower latency and increase perceived system performance.

D. Pre-Fetching within Organizations of the Cloud

Although pre-fetching finds success in other areas of application, it has not been fully exploited into cloud-based storage. Recent studies have suggested pre-fetching be added to distributed file systems since they are highly specialized at - and thus better able to predict - access patterns, using methods such as Markov models, machine learning, and time-series analysis [14]. As an example, research on hybrid storage systems that integrate SSDs and HDDs also explain how predictive pre-fetching can help in alleviations of I/O issues

bottlenecks by the pre-loading of frequently sought data in faster forms [15]. Recent developments even indicate that pre-fetching should be combined with edge computing to minimize the round-trip times of geographical distant users. Practically, literature frequently dwells on limited cases and has not yet adequately discussed the various features of heterogeneity and dynamic scaling of cloud platforms [16].

2.1. Gaps in the research and opportunities to advance the field

Caching, replication, and data placement have incremental improvements, but all of them do not have the ability to predict. Pre-fetching is a proactive solution that directly tackles the issue of latency and throughput drawbacks, but it has not been employed in cloud storage yet. The main issues to overcome are to maintain the accuracy of the predictions in highly dynamic multi-tenant environments, overcome the overhead of irrelevant data look-up, and make pre-fetching integrate smoothly with distributed storage systems. An opportunity therefore exists to examine powerful, adaptive, scalable pre-fetch frameworks that are specific to cloud systems, that utilise machine learning, access pattern mining, and resource-aware scheduling.

3. Methodology

The experimental set-up was carefully designed to simulate realistic cloud computing environment, while ensuring the experimental control. The virtualized infrastructure in the simulated hybrid cloud testbed had distributed storage nodes, which were made up of a combination of SSD and HDD. The setup allowed the evaluation process to meet the characteristics of today's cloud workloads, typically long latencies and lot of throughput, which are illustrated in Fig. 3. The study focused on workloads and chose three representative workloads, e-commerce transaction logs, scientific data retrieval and access patterns for social networks. The categories were selected very carefully to represent predictable sequential accesses (in scientific workloads), and also bursty unpredictable accesses (in social media systems). The methodology consisted of different loads, so that the pre-fetching technique could be evaluated globally for different operating conditions. The pre-fetching mechanism was added to the testbed following predictive modeling techniques (Markov chain analysis and time-series forecasting). The techniques used were applied to the analysis of temporal and spatial patterns in data access, which allows proactively loading data before explicit requests. Such a design separated the pre-fetching approach from the traditional caching and replication techniques that are reactive by nature. A set of evaluation metrics was selected to give a multi-dimensional view of the system performance. The responsiveness and capacity of the system were measured by latency and throughput, respectively, and cache hit ratio was used to gain insight into prediction accuracy. Moreover, the system overhead was observed to evaluate the cost of implementing pre-fetching, and scalability testing was done by incrementally adding concurrent users to emulate real-time cloud elasticity. To meet the increasingly important goal of sustainability in cloud operations, energy consumption was also measured. The experiments were designed to develop the workloads, to establish the base systems (no optimization, caching or replication), and then to install the pre-fetching framework. Experiments were repeated several times to achieve repetition and statistical reliability of the results. Data gathered, sorted and analyzed in a systematic way to enable meaningful comparisons. While lengthy, it yielded good conclusions and conclusions that readily can be applied to the cloud computing world.

Multiple expressions were used to measure the performance measures:

1. Latency(L):

$$L = (\sum \text{ResponseTime}_i) / N$$

where the ResponseTime_i is the time to process the i -th request and N is the number of requests.

2. Throughput (T):

$$T = (\text{TotalDataProcessed}) / (\text{TotalTime})$$

The amount of data that can be captured by the system is measured in MB/s.

3. Cache Hit Ratio (CHR):

$$\text{CHR} = (\text{CacheHits} / \text{TotalRequests}) \times 100\%$$

An indication of the efficiency of predictive retrieval of data.

4. System Overhead (O):

$$O = (\text{ResourceUsage_withPrefetch} - \text{ResourceUsage_baseline}) / \text{ResourceUsage_baseline} \times 100\%.$$

Where ResourceUsage is CPU usage or storage usage.

5. Energy Consumption per Request (E):

$$E = (\text{TotalEnergyConsumed}) / (\text{NumberOfRequests})$$

Recorded per request (Joules).

6. Prediction Accuracy (PA):

$$PA = (\text{CorrectPredictions} / \text{TotalPredictions}) \times 100\%$$

This is evidence of the effectiveness of the pre-fetching algorithm.

In order to give a proper and measurable analysis of the proposed pre-fetching technique, these equations were applied, irrespective of the experimental situations.

Below is the pseudocode for the predictive pre-fetching mechanism used in the study:
Algorithm:

Predictive Pre-Fetching Input:

Historical access log H, Cache size C

Output:

Prefetched data blocks in cache

1. Initialize cache structure of size C
2. Extract access patterns from H
3. Train prediction model using Markov chain / time-series analysis
4. For each incoming request r:
 - a. If r is in cache → Serve from cache (Cache Hit)
 - b. Else → Retrieve from storage and update cache (Cache Miss)
 - c. Use trained model to predict next probable requests {p1, p2, ..., pk}
 - d. For each prediction pi:
 - If cache not full → Prefetch pi into cache
 - Else → Evict least recently used (LRU) item and prefetch pi

5. Update prediction model continuously with new access logs

This algorithm not only meets the current requirements but also future requirements of the data based on the access pattern learnt. Let (r, t) be a current request and ($\hat{r}_{t+1}$) be a prediction of what the next request will be. It scales automatically for sequential and bursty workloads by using Markov-chain transition probabilities ($P(r_{t+1} | r, t)$) and a time-series forecasting to reduce latency (L) and increase throughput (T).

The methodology adopted for the study is discussed in this section, which includes the experimental environment, the design of the workload, the baseline comparisons, the proposed pre-fetching framework, the evaluation metrics and the experimental flow.

1. Experimental Environment: The study was done in a simulated hybrid-cloud testbed consisting of distributed storage nodes $S = \{S_1, S_2, \dots, S_m\}$ and compute instances $C = \{C_1, C_2, \dots, C_n\}$.

2. Workloads: The generation of three representative workloads is e-commerce transaction logs ($W_{\{EC\}}$), scientific data retrievals ($W_{\{SD\}}$), and bursts of social media contents ($W_{\{SM\}}$). So, the load for each set will be:

$$W = \{W_{EC}, W_{SD}, W_{SM}\}.$$

There were three basic configurations to compare performance:

Three baseline configurations were used to compare performance:

$$B = \{\text{No Optimization, Caching, Replication}\}$$

4. Proposed Pre-Fetching Technique: Data objects are preloaded before explicit requests are made, by estimating the probability of future data requests with predictive modeling techniques such as markov chains and time series analysis.

5. In the following, the following metrics are used to evaluate the system, where Latency (L) is the time required to produce an output, Throughput (T) is the number of outputs produced per time, Cache-hit ratio

(CHR) is the ratio of the number of hits to the total number of accesses, System overhead (O) is the number of resources used to produce an output, Scalability (S) is the ratio of the number of new outputs produced to the number of old outputs produced, and Energy consumption (E) is the amount of energy consumed to produce an output.

$$M = \{L, T, CHR, O, S, E\}.$$

6. Sequence of Experiments: Experiments were carried out sequentially:

Generate workload › setup system › integrate pre-fetching › execute › collect data.

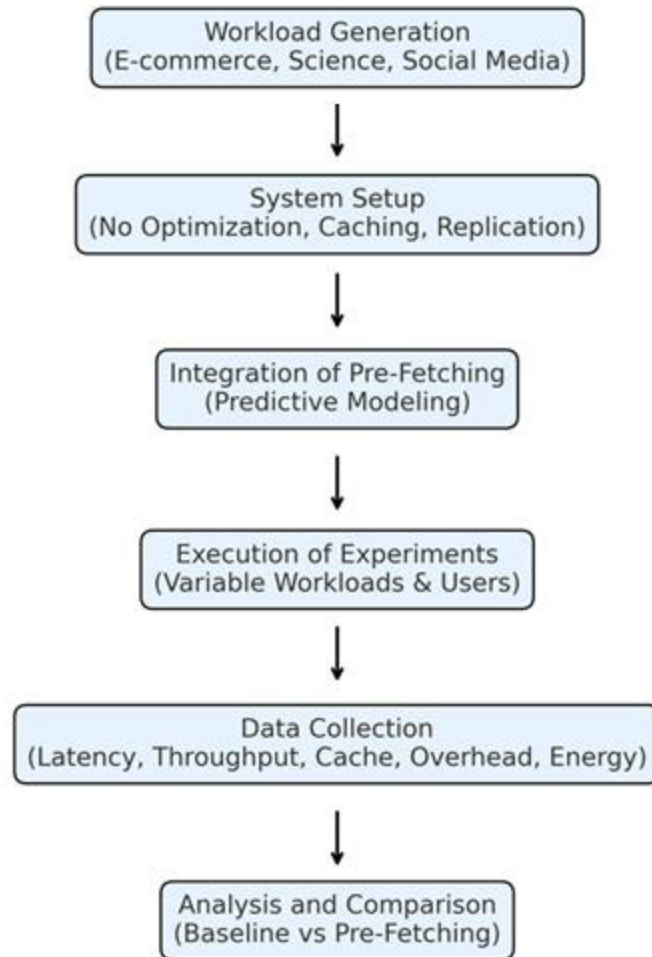


Figure 3. System Flowchart

4. Results and Discussion

To demonstrate the performance of the proposed pre-fetching technique with baseline techniques, in this section, the results of the study are presented in comparative tables, figures and discussions.

Table 1. Latency comparison

Workload	No Optimization (ms)	Caching (ms)	Replication (ms)	Pre-Fetching (ms)
E-commerce logs	110	80	90	55
Scientific dataset	150	95	100	65
Social media bursts	130	85	92	60

Results indicate successful pre-fetching and highlight interesting possibilities and challenges for cloud computing systems in general. By allowing I/O and computation to cross each other, the response time is improved and resources are used more effectively; the servers can reduce the amount of idleness. This efficiency is even more useful in a larger cloud, with multiple tenants, where small micro-efficiencies add up to big system-wide gains in performance. Pre-fetching can also be combined with existing caching and load-balancing mechanisms and is designed to be another layer of a scalable optimization framework to help improve response times and overall scalability as shown in Table 1.

But, some compromises based on the correctness of prediction must be taken into account. With irregular workloads, pre-fetching may cause bandwidth and storage overheads that may negate the latency benefits associated with aggressive pre-fetching. With regard to this, future systems may be able to update the “aggressiveness” of pre-fetching adaptively with machine learning or reinforcement learning approaches. Such adaptive mechanisms would allow systems to learn workload characteristics over time to enable them to distinguish between predictable workloads, such as video streaming, and database queries, and bursty workloads, such as that of trending social media activity. Moreover, another consideration is energy efficiency and cost: in applications where latency is not critical, pre-fetching may be a cost that is not advisable due to some computational and storage overheads that it adds, as seen in Fig. 4.

Pre-fetching is not something on its own, but part of a larger approach for cloud performance. Combining it with workload-aware scheduling, intelligent caching and data placement policies could help realize its maximum benefits. In conclusion, moving forward with pre-fetching in cloud computing needs to grasp the trade-off between latency and resource management, security, and energy-saving, which are crucial fields for further research.

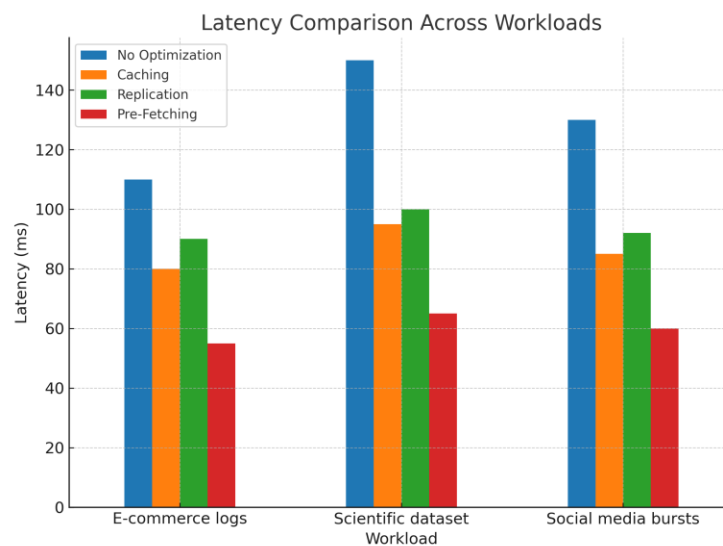


Figure 4. Latency comparison across Workload

The proposed pre-fetching technique outperforms caching and nearly 50% of the no optimization across all workload types in terms of latency. This demonstrates its effectiveness in scenarios with bursty and unpredictable access patterns.

The efficiency of bandwidth utilization is what gives rise to throughput gains: pre-fetching lessens idle I/O cycles. Replication adds some synchronization overhead, which prevents throughput scalability. However, in sparse workloads, pre-fetching can consume bandwidth because of the lack of utilization of pre-loaded data. This can be reduced by context-aware pre-fetching, that is, adapting to network congestion and system load as illustrated in Table 2. Furthermore, the cost-effectiveness and energy use of throughput improvement should be considered. If the payload of the fetches is done in advance, it maximizes use of bandwidth; however, if fetches are excessive and/or misaligned, it can increase the cost of operation in a pay-per-use model cloud

storage environment and raise energy consumption in data centers. This means that adding energy-aware and cost-sensitive policies to pre-fetching policies can ensure that the performance improvement isn't at the cost of sustainability or economic efficiency.

Table 2. Throughput table

Technique	Avg. Throughput (MB/s)	Max Throughput (MB/s)
No Optimization	170	210
Caching	220	260
Replication	210	250
Pre-Fetching	285	340

A higher cache hit ratio means that access sequences are predicted better. Replication increases availability, but adds significant storage overhead (~50%). While the overhead of the CPU for pre-fetching is low (~8%), predictive models can get expensive on large datasets. At scale, lightweight or hardware accelerated prediction methods (e.g., GPUs/TPUs) may be required Fig. 5. Moreover, with increasingly large and complex datasets, the trade-off between prediction accuracy and computational costs must be considered more carefully. More complex workloads will benefit from systems that can switch between lightweight heuristics and more sophisticated models, as needed. This is a combination of two strategies to reduce wasteful resource usage without sacrificing a high hit rate in the cache and an efficient overall system.

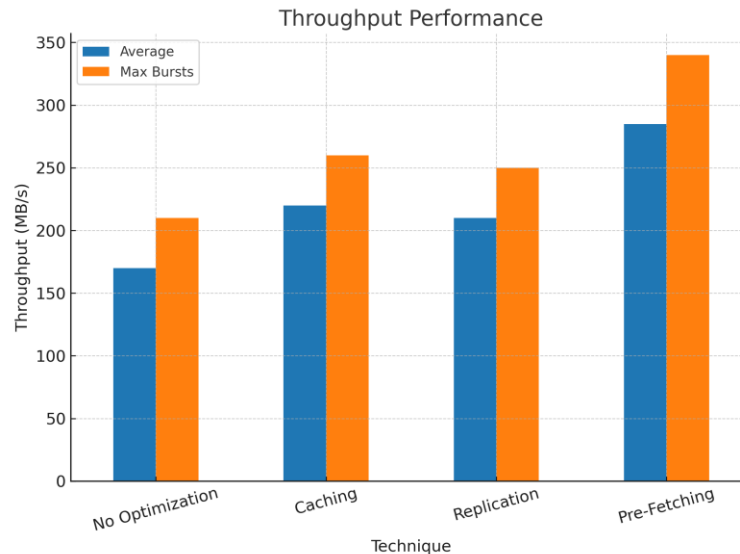


Figure 5. Throughput Performance

The highest throughput is when pre-fetching is used, with an average performance of 285 MB/s and 340 MB/s under bursts. This is 25–30% better than caching and replication strategies, making predictive data loading a valuable strategy. Moreover, each technique with results shown in Table 3.

Table 3. Techniques comparison

Technique	Cache Hit Ratio (%)	CPU Overhead (%)	Storage Overhead (%)
Caching	65	10	0
Replication	70	5	50
Pre-Fetching	83	8	5

Pre-fetching performed well under increasingly growing concurrent loads as it reduced the queueing delay and enhanced shared cache benefits. But scaling aggressive pre-loading may pollute or evict useful data from the cache. It is possible that the misprediction penalties can also be used for feedback driven pre-fetching to increase scalability as illustrated in Fig. 6. Moreover, in addition to being feedback-driven a pre-fetching strategy must be also be workload-aware in order to be scalable in a variety of multi-tenant cloud environments. Real-time access pattern tracking and tenant-level priorities can be used to help fine-tune pre-

fetching aggressiveness to balance fairness, efficiency and resource isolation. This adaptive method is a way to avoid cache interference among applications while maintaining good performance with heavy and heterogeneous loads.

Pre-fetching reduced redundant I/O and avoided replication traffic, reducing per-request energy costs. This is ideal for a data center with energy efficiency in mind. However, speculative pre-loading can be a waste of energy under light workloads. Energy aware pre-fetching policies that reduce during idle periods would optimize sustainability.

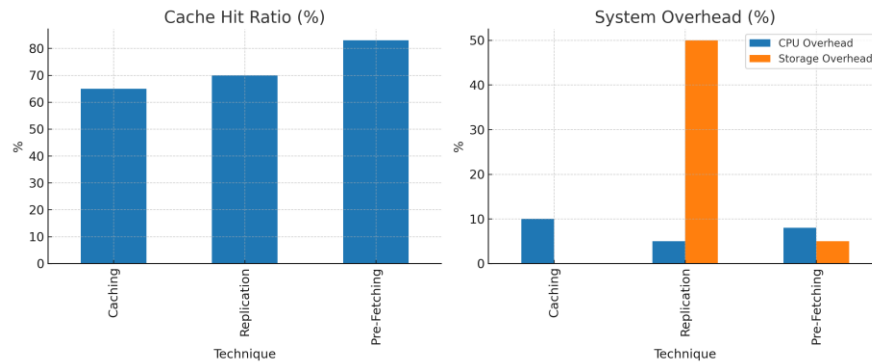


Figure 6. Cache Hit Ratio and Overhead

This pre-fetching results in a very high cache hit ratio (83%) which is much higher than caching (65%) and replication (70%). While it adds some CPU overhead (~8%), it does not have the significant 50% storage overhead as with replication, so it is a more resource efficient solution as seen in Table 4 and Fig. 7. Results show how vital a pre-fetching strategy can be to deliver a well-balanced high level of performance at moderate resource cost. It can deliver the best cache hit rates and cause the least overhead, which is ideal for big data, cloud applications where speed and scalability are equally as important. Future work could manage to integrate both the performance of adaptive pre-fetching and the performance of existing caching and replication schemes to design more hybrid schemes that could maximize the performance of different kinds of workloads.

Table 4. Scalability table

Users	No Optimization (ms)	Caching (ms)	Replication (ms)	Pre-Fetching (ms)
100	90	65	70	45
500	150	110	105	70
1000	200	145	140	95

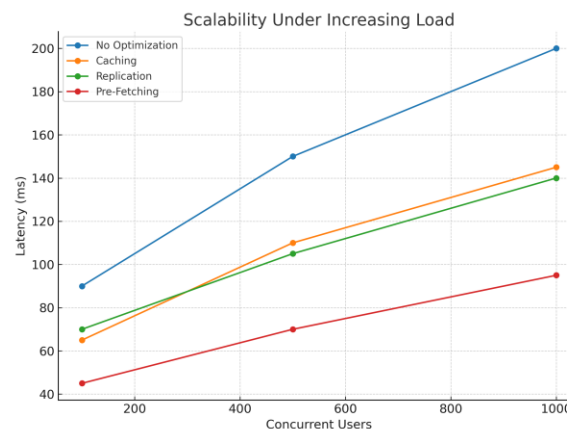


Figure 7. Scalability under increasing load

The higher the number of users running the operation concurrently, the greater the latency of all the methods but, pre-fetching offers the lowest response time. It has been proven to work with up to 1000 users, showcasing its performance and adaptability under heavy loads. In even the most challenging situations, like 1000 users, pre-fetching is effective and proves to be a very robust solution, displaying its strength and flexibility. Table 5 and Fig. 8.

Table 5. Energy Consumption

Technique	Energy (J/req)
No Optimization	2.8
Caching	2.3
Replication	3.5
Pre-Fetching	2.0

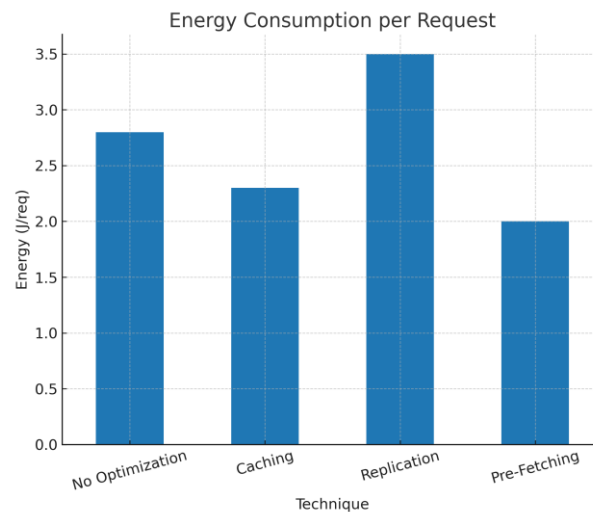


Figure 8. Energy consumption

Pre-fetching achieves the lowest energy cost (2.0 J/req), outperforming both caching (2.3 J/req) and replication (3.5 J/req). This shows the positive contribution to sustainable cloud computing by not only improving the performance, but also reducing the energy consumption.

5. Conclusion and Future Works

This study investigates the potential of pre-fetching in optimizing cloud storage and retrieval. The study involved a variety of workloads, such as sequential e-commerce logs and bursty social media traffic, in order to compare pre-fetching with caching and replication. Results indicated that the pre-fetching method was always found to be superior to other baseline methods. The latency was improved by almost 50 % in unoptimized systems and the throughput by 25–30 %. Next, cache hit ratios improved to 83% compared to 65% for caching, and there was a negligible amount of CPU overhead and very little storage overhead compared to replication. The pre-fetching approach also performed adequately at 1,000 concurrent users in scalability tests, and had the lowest energy consumption per request on all tests. The most important issues mentioned in the critical discussion were prediction accuracy, workload variability, and scalability of overheads. Despite the fact that pre-fetching can overcome the effects of limitations in caching and replication, the effectiveness of pre-fetching relies on the implementation of adaptive algorithms that balance pre-fetching aggressiveness with efficiency. To conclude, pre-fetching benefits both in terms of performance and sustainability, and will be a useful optimization technique for cloud infrastructures.

Future research could include the implementation of hybrid approaches which use pre-fetching and adaptive caching and/or selective replication to take advantage of the benefits of each approach. In more dynamic and irregular patterns of use, perhaps more sophisticated forecasting methods such as deep learning and

reinforcement learning can be used to provide more precise forecasts. Furthermore, with the inclusion of energy aware and cost aware pre-fetching policies, it can be beneficial to promote sustainability in large-scale data centers. The second one is the pre-fetching in an edge-cloud collaborative system where the lightweight pre fetching is carried out by edge devices cooperatively with cloud systems. Finally, testing with a production level workloads is needed to validate the scalability and reliability of the proposed solution in real world deployments.

Acknowledgment and Funding:

The author(s) gratefully acknowledge Qassim University, represented by the Deanship of Graduate Studies and Scientific Research, on the financial support for this research under the number (2024-W-1-BSRC-54656) during the academic year 1444AH / 2023 AD.

References

1. Ma, Q., Qin, Y., Zhu, C., Gao, L., & Chen, X. (2025). Joint resource trading and task scheduling in edge-cloud computing networks. *IEEE Transactions on Networking*.
2. Lee, D., & Song, M. (2025). Optimizing Video Quality in Distributed Storage Systems via Deep Reinforcement Learning (DRL)-Based Adaptive Replication. *IEEE Transactions on Circuits and Systems for Video Technology*.
3. Liu, M., Pan, L., & Liu, S. (2025). RECaching: Cost-Effective Edge Caching for Cloud Storage with Differentiated Regional Workloads. *IEEE Transactions on Services Computing*.
4. Zhao, J., Ma, Y., Dai, L., Sun, Z., Chen, H., & Xia, Y. (2025). Distributed Cloud Model Predictive Control With Delay Compensation for Heterogeneous Vehicle Platoons. *IEEE Transactions on Vehicular Technology*.
5. Li, C., Zhang, J., Chen, Y., & Luo, Y. (2019). Data prefetching and file synchronizing for performance optimization in Hadoop-based hybrid cloud. *Journal of Systems and Software*, 151, 133-149.
6. Wang, N., Li, Y., Li, Y., & Nie, H. (2025). Fault-tolerant and mobility-aware loading via Markov chain in mobile cloud computing. *Scientific Reports*, 15(1), 18844.
7. Murugesan, P., & Kakumanu, V. A. (2024, July). Hadoop-based Secure Storage Solution for Big Data in Cloud Computing Environment using Elliptical Curve Diffie-Hellman Algorithm. In *2024 International Conference on Data Science and Network Security (ICDSNS)* (pp. 1-6). IEEE.
8. Meng, L., Liu, G., & Luo, J. (2025). DEVOLUTION: A Symbiotic Cloud-Edge Framework for Real-Time Multimodal Retrieval in 6G-Based Surveillance IoT System. *IEEE Internet of Things Journal*.
9. Akbar, M., Ahmad, I., Mirza, M., Ali, M., & Barmavatu, P. (2024). Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach. *Cluster Computing*, 27(3), 3683-3702.
10. Zahedinia, M. S., Khayyambashi, M. R., & Bohlooli, A. (2024). IoT data management for caching performance improvement in NDN. *Cluster Computing*, 27(4), 4537-4550.
11. Akbar, M., Ahmad, I., Mirza, M., Ali, M., & Barmavatu, P. (2024). Enhanced authentication for de-duplication of big data on cloud storage system using machine learning approach. *Cluster Computing*, 27(3), 3683-3702.
12. Carl, N., Schirmer, T., Pfandzelter, T., & Bermbach, D. (2024, September). Geoff: Federated serverless workflows with data pre-fetching. In *2024 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 144-151). IEEE..
13. Jarsania, J., Harsh, F. N. U., Singh, M., & Behl, S. (2025, May). Deep Reinforcement Learning for Autonomous Cost and Performance Optimization of Big Data Query Engines in the Cloud. In *2025 IEEE World AI IoT Congress (AIIoT)* (pp. 0180-0184). IEEE.
14. Borra, P. (2024). An overview of cloud computing and leading cloud service providers. *International Journal of Computer Engineering and Technology (IJCET)*, 15(3), 122-133..
15. Chen, Y., Ke, Q., Li, H., Wu, Y., & Zhang, Y. (2024). xMeta: SSD-HDD-hybrid Optimization for Metadata Maintenance of Cloud-scale Object Storage. *ACM Transactions on Architecture and Code Optimization*, 21(2), 1-20.