Journal of Discrete Mathematical Sciences & Cryptography
ISSN 0972-0529 (Print), ISSN 2169-0065 (Online)
Vol. 28 (2025), No. 5-A, pp. 1649–1657
DOI : 10.47974/JDMSC-2165

Enhancing cloud performance : A framework for connection pooling optimization

Ankit Kumar Tiwari *
Department of Computer Science & Engineering
JECRC University
Jaipur 302022
Rajasthan
India

Surendra Yadav †
Department of Computer Science & Engineering
Vivekananda Global University
Jaipur 303012
Rajasthan
India

Abstract

Performance optimization in cloud computing is of vital importance to ensure efficient resource use and to meet the demands of users. In this paper, we propose a framework that will optimize the efficiency of the cloud system through connection pooling. Traditional ways are normally inefficient due to continuous connection creation and termination, which adds to the overhead and causes delay. Our proposed framework resolves these issues by integrating the connections and reusing them to decrease latency, increase throughput, and improve scalability. Simulations run under different load conditions, such as Low, Medium, and High, showed dramatic improvements in performance. For example, at Low Load, it had an average of 0.002579 seconds in query time, and the throughput became 3.33 transactions per second. At High Load, it gave an average of 0.005134 seconds in query time and a throughput of 5 TPS. These results prove that connection pooling does improve performance bottlenecks, allowing cloud infrastructure to handle most diversities of workloads. Conclusively, this framework is a hands-on approach to the optimization of cloud performance and will make any web application scale better and be responsive to growing online service demands.

Subject Classification: *Primary 68M25, Secondary 68M20.*

Keywords: *Cloud computing, Connection pooling, Performance optimization, System scalability, Throughput, Latency reduction, Web application efficiency, Resource management.*

* *E-mail:* **ankittiwari148@gmail.com (Corresponding Author)**
† *E-mail:* **surendra.yadav@vgu.ac.in**

1. Introduction

Due to the rapid growth of online services and applications, requirements for their effective cloud computing solutions have also been realized. Cloud environments, scalable and flexible, form the backbone in meeting such diverse requirements of modern web applications. Connection management in traditional approaches frequently sets up and tears down connections, usually resulting in performance bottlenecks, increased latency, and reduced throughput. One of the well-established techniques to resolve such inefficiencies is connection pooling, which reuses existing connections rather than making a new connection. This would reduce the overhead associated with connection management and improve system performance and scalability. However, despite its efficiency, connection pooling in cloud environments remains relatively unexplored [1, 2].

We present herein a design of a framework that implements connection pooling for performance optimization within a cloud environment. In this paper, we consider latency minimization, throughput maximization, and scalability for the framework that we are going to design. We will validate our approach against different load conditions simulated on it and show how this work turns into practical solutions for performance challenges in cloud systems. In this paper, we describe the design, implementation, and assessment of our connection pool framework. We detail how our findings can be used to gain insight into cloud performance optimization [11, 12]. One of the most critical factors affecting high performance on cloud systems is efficient management of the connections [3, 4].

The primary objectives of this research are as follows:

1) **To Explore Connection Pooling Techniques**:
   a) Investigate various connection pooling techniques and their applicability in cloud computing environments.
   b) Identify key factors influencing the efficiency of connection pooling in cloud systems.
2) **To Develop an Algorithmic Framework**:
   a) Design and implement a robust framework for optimizing cloud performance using connection pooling techniques.
   b) Ensure the framework effectively manages and reuses connections to reduce latency and improve throughput.
3) **To Simulate and Evaluate Performance**:
   a) Conduct comprehensive simulations under different load conditions (Low Load, Medium Load, High Load) to evaluate the performance of the proposed framework.

b) Measure key performance metrics such as average query time and throughput in varying operational scenarios.

4) **To Analyze and Validate Results**:
   a) Analyze the empirical data obtained from simulations to validate the effectiveness of the connection pooling framework in optimizing cloud performance.
   b) Identify performance bottlenecks and areas for improvement within the framework.

2. Literature Survey

The recent trends in cloud computing have huge research contributions geared toward bettering efficiency, performance, and cost-effectiveness. For example, Li et al. [5] proposed Pond; a memory pooling system based on the Compute Express Link standard, and reduced the DRAM costs for cloud providers by 7%. Berger et al. [6] studied how CXL memory pools can efficiently utilize DRAM in public cloud servers and also offered a ringside view into the design considerations and deployment options. Kumar and Agrawal [7] remarked on the potential of new emerging architectures of Edge, Fog, and Cloud computing that can bring an improvement in in-network performance towards handling IIoT data analysis tasks. The other main thrust area which has evinced interest pertains to optimizing algorithms. Mangalampalli, Karri, and Kumar [8] in 2023 proposed Grey Wolf Optimization for dynamic resource allocation in cloud computing that optimized some of the metrics involving makespan, resource utilization, and energy consumption. Adaikalaraj and Chandrasekar developed an improved lion optimization-based intelligent virtual machine programming scheme with a min-max algorithm [9]. Major contributions include improvements in network speed, load balancing, and power conservation by Wang et al. [10].

3. Research Methodology

The research takes a step-by-step process in the optimization of cloud performance using connection pooling techniques. Literature review: the current literature in the area of connection pooling within cloud environments is focused on establishing gaps and opportunities. The study will then come up with an algorithmic framework dwelling on ways of reducing latency, increasing throughput, and enhancing scalability through deep tuning on connection poolers like pg-Bouncer and DBCP in PostgreSQL. The framework is implemented and tested against simulated real-world conditions to examine performance metrics of the system such as query time and throughput. Finally, research results are evaluated,

providing guidelines on how to optimize cloud systems in order to achieve scalability and efficiency.

4. Proposed Approach

A. *Connection Pooling*

Connection pooling within cloud computing manages database connections effectively for a given distributed application and offers great performance improvements. The reason for this is simply that it saves resources and reduces latency by reusing existing connections rather than making new ones for any requested job. This will become crucial in a cloud environment where applications may need to scale up due to fluctuating demand. This makes connection pooling improve scalability while reducing resource usage and speeding up response times, hence turning it into a very critical component in cloud application performance optimization.

B. *Algorithm for Proposed Approach*

Algorithm 1: Dataset Generation Algorithm	**Algorithm 2:** Simulation Scenario Design Algorithm
1. **Input**: Dataset Schema 2. **Output**: Generated dataset 3. Define schema for dataset with tables & column 4. Determine dataset size and characteristics 5. Use dataset generation tool to create synthetic data 6. Validate generated data for consistency& integrity 7. **Optional:** Add variability to dataset	1.**Input**: System performance parameter 2.**Output**: simulation Scenario 3. Identify key performance factors (concurrent users, query types) 4. Define simulation scenarios with different workload patterns 5. Specify input parameters for each scenarios (size, query, concurrent) 6. Determine duration & frequency for each simulation 7. Validate realism & diversity for scenario
Algorithm 3: Data Pre-processing & Loading Algorithm.	**Algorithm 2:** Performance Measurement & Analysis Algorithm
1. **Input**: Raw dataset 2. **Output**: Loaded and Optimized dataset 3. prepare dataset for loading (eg. Format conversion) 4. Create database schema for database structure 5. Load database into tables 6. Verify data loading success & resolve errors 7. **Optional:** Perform indexing/optimization for query performance	1. **Input**: Simulation results 2. **Output**: Performance metrics analysis 3. Instruments environment to collect performance metrics 4. Execute simulation scenarios as per parameter and workload 5. Capture performance data at intervals or upon scenario completion 6. Analyze data to identify trends, patterns and outliers 7. compare performance metrics across scenarios

Performance Metrics

1. **Average Query Time**:

$$\text{Average Query Time} = \frac{\sum_{i=1}^{N} T_i}{N}$$

Where T_i is the query time for the i-th query and N is the total number of queries.

2. **Throughput**:

$$\text{Throughput} = \frac{\text{Number of Transactions}}{\text{Total Time}}$$

Where "Number of Transactions" is the total number of transactions processed, and "Total Time" is the time taken to process these transactions.

3. **Latency**:

$$\text{Latency} = T_{\text{response}} - T_{\text{request}}$$

Where T_{response} is the time when the response is received and T_{request} is the time when the request was made.

4. **Connection Pool Utilization**:

$$\text{Connection Pool Utilization} = \frac{\text{Number of Active Connections}}{\text{Total Pool Size}}$$

Where "Number of Active Connections" is the current number of active connections, and "Total Pool Size" is the maximum number of connections in the pool.

5. Result Analysis

A. *Dataset*

It constitutes the database for complete performance metrics measures extracted in a simulated cloud environment and represents processor and memory usage, network traffic, power consumption, instruction count, execution time, energy efficiency, task types, priorities, states, etc. It copies the real-world conditions of cloud systems and hence offers a very useful resource into the investigation of energy efficiency and performance optimization. It can be used to develop, with the aid of machine learning, techniques that would achieve energy efficiency, reduce execution times, and decrease the overall operational cost, hence going a step further toward greener and more cost-effective solutions in cloud computing. (The description of data set column details shown in figure 1)

The image contains histograms for some numeric columns from the cloud computing performance dataset, all of which represent the distribution of some metrics. One can notice that CPU usage, memory usage, and energy efficiency all have a uniform distribution across their respective ranges: 0-100%, 0-100%, and 0-1. (Show in figure 2)

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000000 entries, 0 to 1999999
Data columns (total 12 columns):
 #   Column                      Dtype
---  ------                      -----
 0   vm_id                       object
 1   timestamp                   object
 2   cpu_usage                   float64
 3   memory_usage                float64
 4   network_traffic             float64
 5   power_consumption           float64
 6   num_executed_instructions   float64
 7   execution_time              float64
 8   energy_efficiency           float64
 9   task_type                   object
 10  task_priority               object
 11  task_status                 object
dtypes: float64(7), object(5)
memory usage: 183.1+ MB
```

Figure 1

Dataset Structure

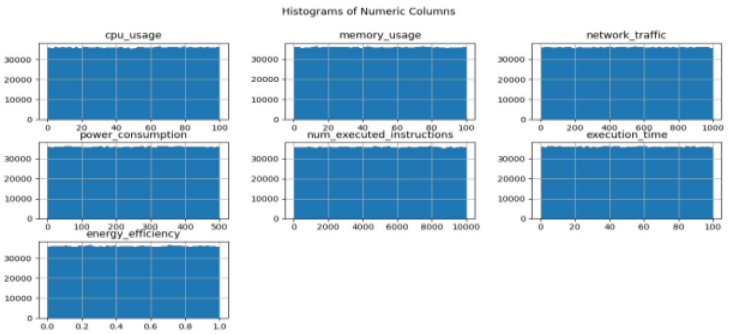

Figure 2

Histogram of Numeric Columns

Network traffic also has a uniform distribution across a wider range of 0-1000 units; another is power consumption, uniformly distributed across an even larger range of 0-500 units; and another is the number of executed instructions, uniformly distributed across a much larger one of 0-10,000 units. Here, it ranges uniformly from 0 to 100 units of execution time. This uniformity could be an indication that either this dataset has been simulated, or due to the fact that it is balanced to ensure good coverage for all possible values in order to have unbiased analysis and optimization in cloud performance studies. (Show in figure 3)

5.1 *Result Analysis*

For evaluation purposes, three testing scenarios have been prepared namely: Low Load, Medium Load, and High Load. Key performance indicators, to give details about the capacity and behavior of the system,

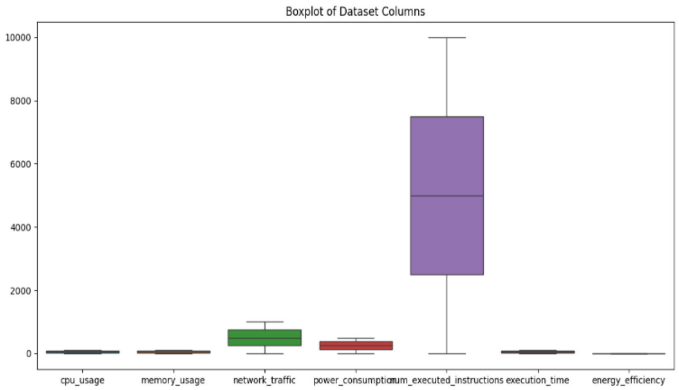

Figure 3

Bloxplot of data columns

are CPU use, memory consumption, reaction time, throughput, and network traffic, among others. Each condition runs for 60 minutes, and specific conditions are focused on below:

Low Load: Parameters:10 concurrent users, read only queries (SELECT).

Objective: Set up a performance baseline under minimal stress, demonstrating that the system is efficient at light traffic, comparison graph shown in Figure 4.

Medium Load: Parameters: 50 concurrent users with mixed queries SELECT, UPDATE. Determine the system's capacity to maintain read/ write balance under moderate dynamic stress conditions. Key emphasis on resource utilization and response times.

High Load: Parameters: 100 concurrent users with mixed queries SELECT, UPDATE, INSERT. Stress-test the system to reveal bottlenecks

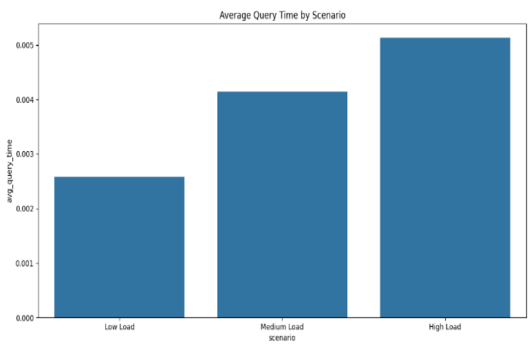

Figure 4

Average Query Time

and performance degradation at peak load, and analyze resource limits and opportunities for optimization.

In a medium load scenario with 50 concurrent users, the average query time is 0.004142 seconds, and throughput remains stable at 3.33 queries per second, reflecting the system's ability to handle a moderate load efficiently. In a high load scenario with 100 concurrent users, the average query time rises to 0.005134 seconds, while throughput increases to 5 queries per second. (Show in Table.1)

Table 1

Performance Matrices

Scenario	Avg Query Time (s)	Throughput (queries/s)
Low Load	0.002579	3.333333
Medium Load	0.004142	3.333333
High Load	0.005134	5.000000

These results highlight the significant impact of connection pooling in managing database connections efficiently, thus reducing the overhead associated with opening new connections. Despite the increase in query times with higher loads, the throughput gain demonstrates the system's enhanced scalability and performance under increased demand. Connection pooling not only supports handling larger volumes of requests and complex queries but also optimizes resource utilization and operational costs, contributing to improved user experience and cloud service efficiency.

6. Conclusion

The results of the research show that connection pooling plays an important role in cloud performance optimization, working under various load scenarios. At a low load, connection pooling delivered an average query time of 0.002579 seconds and a throughput of 3.33 queries per second, thus efficiently working with few users. At medium loads, it increased slightly to a query time of 0.004142 seconds while keeping the same throughput, showing that it is stable under medium workloads. Under high loads, connection pooling scaled pretty well up to 5 queries per second with a query time of 0.005134 seconds, which proves that it resisted heavy loads.

References

[1] J. Singh, "Genetic approach based optimized load balancing in cloud computing: a performance perspective," in *Proc. 9th Int. Conf. Comput. Sustain. Glob. Dev. (INDIACom)*, pp. 814–819 (Mar. 2022).

[2] A. R. Carvalho, H. Ball, M. J. Biercuk, M. R. Hush, and F. Thomsen, "Error-robust quantum logic optimization using a cloud quantum computer interface," *Phys. Rev. Appl.*, vol. 15, no. 6, p. 064054 (Jan. 2021).

[3] I. Attiya, M. Abd Elaziz, and S. Xiong, "Job scheduling in cloud computing using a modified Harris Hawks Optimization and Simulated Annealing Algorithm," *Comput. Intell. Neurosci.*, vol. 2020 (Jan. 2020).

[4] A. N. Malti and M. Hakem, "A new hybrid multi-objective optimization algorithm for task scheduling in cloud systems," *Clust. Comput.*, vol. 27, no. 3, pp. 2525–2548 (Jul. 2023).

[5] H. Li, D. S. Berger, L. Hsu, D. Ernst, P. Zardoshti, "Pond: CXL-based memory pooling systems for cloud platforms," in *Proc. 28th ACM Int. Conf. Archit. Support Program. Lang. Oper. Syst. (ASPLOS)*, vol. 2, pp. 574–587 (Mar. 2023).

[6] D. S. Berger, D. Ernst, H. Li, P. Zardoshti, M. Shah, S. Rajadnya, and R. Bianchini, "Design tradeoffs in CXL-based memory pools for public cloud platforms," *IEEE Comput. Soc.*, vol. 43, no. 2, pp. 30–38 (Mar. 2023).

[7] R. Kumar and N. Agrawal, "Analysis of multi-dimensional Industrial IoT (IIoT) data in Edge-Fog-Cloud based architectural frameworks: A survey on current state and research challenges," *J. Ind. Inf. Integr.* (Mar. 2023), doi: 10.1016/j.jii.2023.100504.

[8] S. Mangalampalli, G. R. Karri, and M. Kumar, "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization," *Clust. Comput.*, vol. 26, no. 6, pp. 3803–3822 (Dec. 2023).

[9] J. R. Adaikalaraj and C. Chandrasekar, "To improve the performance on disk load balancing in a cloud environment using improved Lion optimization with min-max algorithm," *Measurement: Sensors*, vol. 27, p. 100834 (Jun. 2023).

[10] S. Wang, X. Li, Q. Z. Sheng, and A. Beheshti, "Performance analysis and optimization on scheduling stochastic cloud service requests: A survey," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 3, pp. 3587–3602 (Sep. 2022).

[11] A. K. Tiwari and S. Yadav, "Algorithmic model for cloud performance optimization using connection pooling technique," *J. Stat. Manag. Syst.* (Mar. 2024). ISSN 0972-0510 (Print), ISSN 2169-0014.

[12] A. K. Tiwari and S. Yadav, "Boosting cloud infrastructure performance: The role of connection pooling in handling diverse workloads," in *IET Conf. Proc.*, vol. 2024, no. 38, pp. 70–73.

Received November, 2024