CS 5513-001: ADVANCED DATABASE MANAGEMENT SYSTEM

GROUP 9 PROJECT PROPOSAL

1) Project Title: Extending MySQL with Three New Query Caching Strategies and Comparative Analysis

2) Names and Email Addresses of Authors:

Sasank Sribhashyam (sasank.sribhashyam-1@ou.edu)

Venkat Tarun Adda (adda0003@ou.edu)

Chandana Sree Kamasani (chandana.sree.kamasani-1@ou.edu)

3) Category and Objectives of the Project:

Category 3: Extending an Existing Open-Source Database System

This project falls under **Category 3**, where we aim to extend an existing open-source database system by prototyping additional query caching strategies, incorporating them into the database engine, and evaluating their impact on performance. The extension will enhance MySQL's query caching capabilities, which were deprecated in version 8.0, by integrating three alternative caching strategies. This work contributes to open-source database research by improving query efficiency and providing comparative insights into modern caching techniques.

Justification for Choosing MySQL as the Open-Source Database System:

MySQL is an ideal candidate for this project due to the following reasons:

- **Widespread Adoption**: MySQL is extensively used in web applications, enterprise solutions, and cloud platforms, making performance improvements highly impactful.
- **Extensibility**: MySQL supports pluggable storage engines and custom query execution modifications, allowing seamless integration of new caching mechanisms.
- **Existing Performance Limitations**: MySQL removed its query cache in version 8.0 due to performance inefficiencies, leading to increased execution times for repetitive queries. This presents an opportunity to implement alternative caching methods.
- **Benchmarking Capabilities**: MySQL's open-source nature allows modifications and performance testing using industry-standard benchmarking tools such as TPC-H and SysBench.

Project Objectives:

The primary goal of this project is to extend MySQL's query execution framework with three caching strategies and evaluate their effectiveness. The key objectives are:

A. Implement and Integrate New Query Caching Strategies

Extend MySQL to support the following caching techniques:

- **Segmented Least Recently Used (SLRU)**: Divides cache into separate segments to enhance eviction policies [Gao, 2010].

- **Two-Queue Caching (2Q)**: Utilizes separate queues for short-term and long-term query caching to improve efficiency [Johnson, 1994].

- **Adaptive Replacement Cache (ARC)**: Dynamically adjusts between recency and frequency-based caching for optimal performance [Megiddo, 2003].

- Modify MySQL's caching framework to incorporate these techniques while maintaining compatibility with the query execution engine.

B. Justify the Selection of Caching Strategies

- Conduct a literature review to evaluate the effectiveness of SLRU, 2Q, and ARC in database caching.

- Compare these strategies against traditional LRU-based caching and MySQL's buffer pool.

- Demonstrate the advantages of these techniques over MySQL's default caching behavior.

C. Benchmark and Evaluate Performance Under Different Workloads

Use real-world and synthetic datasets to measure performance under various conditions:

- **Read-heavy workloads** (e.g., analytical queries, data aggregation).
- **Write-heavy workloads** (e.g., high-frequency transactions, real-time updates).
- **Mixed workloads** (e.g., e-commerce applications, online banking).

Evaluate caching strategies using key performance metrics:

- **Cache hit rate** – The percentage of queries served from cache.
- **Query execution time** – The impact of caching on response times.
- **Memory consumption** – The storage overhead for each caching mechanism.
- **CPU utilization** – The computational cost of maintaining cache consistency.

D. Analyze Trade-offs and Provide Insights for Future Optimization

- Identify the strengths and weaknesses of each caching strategy based on empirical results.

- Determine which caching strategy is best suited for different types of workloads.

- Recommend improvements to MySQL's query caching framework.

E. Deliver an Extended Version of MySQL with Enhanced Query Caching

- Package the modified MySQL version with the implemented caching extensions.
- Demonstrate the extended system's execution using benchmark queries.
- Document findings in a final report with recommendations for database administrators and developers.

4) The Significance of the Project

A. Existing Knowledge and Gaps:

MySQL's original query caching mechanism was removed in version 8.0 due to inefficiencies in handling concurrent workloads, frequent invalidations, and performance bottlenecks. While MySQL's InnoDB Buffer Pool provides page-level caching, it does not cache query results, leading to redundant computation in repeated queries. This limitation significantly impacts performance, especially in read-heavy workloads where the same queries are executed frequently.

- **Segmented LRU (SLRU)** approach extends traditional LRU caching by dividing the cache into multiple segments, each managing query access frequency separately. This improves cache retention for frequently accessed queries while ensuring that newer queries are also accommodated efficiently. While SLRU has been tested in CPU caching and storage systems, its application to MySQL query caching remains unexplored **[Gao, 2010]**.
- **Two-Queue Caching (2Q)** introduces a more refined eviction policy by maintaining two separate queues—one for queries accessed recently and another for queries accessed frequently. This prevents cache pollution caused by one-time queries while ensuring that frequently accessed queries remain in cache longer. While 2Q has been successfully applied in database buffer management, its performance in query result caching for MySQL has yet to be systematically evaluated **[Johnson, 1994].**
- **Adaptive Replacement Cache (ARC)** dynamically balances between recency and frequency-based caching. Unlike LRU-based approaches, which statically prioritize either recent or frequently accessed queries, ARC automatically adapts to workload changes, ensuring optimal cache utilization without manual tuning. ARC has demonstrated superior performance in storage caching but has not been widely tested in database query caching scenarios, making its integration into MySQL an important area of study **[Megiddo, 2003]**.

- **Comparison with Existing MySQL Caching Strategies:** The three new caching strategies (SLRU, 2Q, and ARC) will be compared against the following existing MySQL caching mechanisms:

- **InnoDB Buffer Pool:** MySQL's primary caching mechanism that stores frequently accessed data pages in memory. The InnoDB buffer pool plays a crucial role in optimizing query performance by reducing disk I/O operations. When a query requests data, MySQL first checks if the required pages are already in the buffer pool, thereby minimizing access to slower storage devices. The

buffer pool utilizes a least recently used (LRU) eviction policy, where older and less frequently accessed data pages are replaced when new data needs to be loaded. Properly tuning the buffer pool size is essential to ensuring that frequently used data remains cached in memory, thus improving overall database responsiveness. **[MYSQL Documentation]**

- **Prepared Statement Cache:** This caching mechanism stores execution plans for repeated queries to optimize query processing. When a query is executed multiple times with different parameter values, MySQL does not need to recompile or optimize the SQL statement each time. Instead, the prepared statement cache allows the database to reuse previously generated execution plans, significantly reducing query compilation overhead. This approach is especially beneficial for applications with frequent query execution, such as web applications with dynamic content. However, the efficiency of the prepared statement cache depends on how well the database engine can manage memory allocation and eviction strategies for stored plans. **[MYSQL Documentation]**

- **Key-Based Query Cache (Deprecated in MySQL 8.0):** This caching mechanism previously stored entire query results for reuse but was removed due to performance issues in multi-threaded environments. In versions prior to MySQL 8.0, when a query was executed, its result set was stored in memory, allowing subsequent identical queries to retrieve the results instantly without re-execution. However, due to its global locking mechanism, the query cache often became a bottleneck, particularly in high-concurrency workloads. The cache had to be invalidated whenever the underlying data changed, leading to frequent cache purges and reduced effectiveness. As a result, MySQL developers removed this feature, recommending external caching solutions such as ProxySQL and Redis for achieving similar performance benefits without contention issues. **[MYSQL Documentation]**

B. How This Project Fills the Gaps:

- **Extending MySQL with Advanced Caching Mechanisms:** By implementing SLRU, 2Q, and ARC within MySQL, this project directly addresses the shortcomings of MySQL's existing caching mechanisms. These strategies offer superior cache eviction policies that improve cache hit rates, reduce query execution times, and enhance overall system efficiency. Unlike MySQL's previous query cache, which invalidated entire cache entries whenever a related table was modified, our approach ensures more selective eviction, allowing frequently used queries to remain accessible longer. This enhancement provides MySQL users with more robust and efficient caching options tailored for various workload conditions.

- **Empirical Evaluation with Real-World Workloads:** Unlike previous studies that focus solely on theoretical caching behavior, this project will conduct extensive empirical testing using benchmark workloads that reflect practical database usage. By simulating diverse query patterns, including read-heavy, write-heavy, and mixed workloads, we aim to measure how each caching strategy performs under different database conditions. The results will provide quantitative insights into cache efficiency, memory overhead, and computational impact, offering database administrators concrete data to make informed decisions when optimizing their systems. This hands-on evaluation will bridge the gap between theoretical caching strategies and their real-world application in MySQL.

- **Providing Actionable Insights for Open-Source Databases:** The findings from this study will serve as a valuable resource for MySQL developers, database administrators, and researchers looking to improve database performance through optimized caching. By documenting the trade-offs between SLRU, 2Q, and ARC, we will provide actionable recommendations for selecting the most effective caching mechanism based on workload type and system constraints. Furthermore, the integration of these caching techniques into MySQL's open-source ecosystem will encourage further research and development, potentially leading to future enhancements that refine query caching strategies in open-source databases.

5) Research Methodology and Time Table

Task 1: Literature Review and Problem Identification

Start Date: January 25, 2025

End Date: February 10, 2025

Description:

- Conduct a literature review on query caching, focusing on MySQL's caching mechanisms and alternative strategies. Analyze caching algorithms like SLRU, 2Q, and ARC, exploring their theoretical foundations and practical applications.

- Examine MySQL's InnoDB Buffer Pool and the deprecated Query Cache, identifying their limitations to justify improved caching strategies. Assess the impact of caching on query execution time, cache hit rate, memory usage, and CPU utilization.

- Define the research problem by highlighting MySQL's performance bottlenecks due to the lack of an optimized query cache. Develop key research questions to guide the design, implementation, and evaluation of enhanced caching strategies.

Deliverables:

- Literature Review Report: A comprehensive document summarizing key findings from academic papers and industry research on query caching techniques.

- Problem Statement: A clearly defined research problem emphasizing the limitations of MySQL's existing caching mechanisms and the need for improved strategies.

- Research Questions: A list of fundamental questions that will be addressed in subsequent phases of the project to evaluate the effectiveness of the proposed caching strategies.

Person(s) in Charge: Sasank Sribhashyam, Venkat Tarun Adda, Chandana Sree Kamasani

Task 2: MySQL Codebase Analysis and Prototype Design

Start Date: February 11, 2025

End Date: February 25, 2025

Description:

- Analyze MySQL's source code, focusing on the query execution pipeline, storage engine interactions, and caching mechanisms. Identify integration points for SLRU, 2Q, and ARC without disrupting core functionality.
- Examine InnoDB's buffer management to address potential conflicts with new caching strategies. Design an architectural framework ensuring compatibility with MySQL's query processing workflow.
- Define data structures and algorithms for efficient SLRU, 2Q, and ARC implementation. Develop a prototype blueprint covering memory allocation, eviction policies, and query retrieval.
- Evaluate feasibility as a MySQL plugin or direct modification, setting performance expectations and key success metrics for benchmarking.

Deliverables:

- MySQL Modification Plan: A detailed document outlining the changes required in MySQL's source code to support the new caching mechanisms.
- Caching Design Document: A technical report describing the architecture, data structures, and algorithms for implementing SLRU, 2Q, and ARC in MySQL.
- Prototype Blueprint: A high-level implementation plan detailing how each caching strategy will be integrated and tested.

Person(s) in Charge: Sasank Sribhashyam, **Venkat Tarun Adda,** Chandana Sree Kamasani

Task 3: Implementation of Query Caching Strategies

Start Date: February 26, 2025

End Date: March 25, 2025

Description:

- Integrate SLRU, 2Q, and ARC caching strategies into MySQL's query execution framework using efficient data structures to optimize cache hit rates and eviction policies.

- Modify MySQL's query pipeline to incorporate caching layers, ensuring efficient storage and retrieval. Implement insertion, eviction, and replacement policies for effective memory management.

- Develop logging and debugging tools to track cache performance and optimize algorithms to minimize CPU and memory overhead while improving query response time.

- Ensure multi-threaded compatibility, addressing concurrency and consistency. Conduct unit testing to validate correctness and performance.

- Enable configurable caching parameters for user control and document implementation challenges and design trade-offs.

Deliverables:

- Implemented Caching Strategies: Fully functional implementations of SLRU, 2Q, and ARC integrated into MySQL.

- Source Code Documentation: Detailed documentation explaining the code structure, caching algorithm workflows, and integration points within MySQL's query execution engine.

- Configuration and Setup Guide: Instructions on enabling and tuning the caching strategies for different workload scenarios.

- Testing and Debugging Logs: Logs detailing cache operations, including hit rates, eviction statistics, and performance benchmarks from preliminary tests.

Person(s) in Charge: Sasank Sribhashyam, **Venkat Tarun Adda,** Chandana Sree Kamasani

Task 4: Performance Evaluation and Analysis

Start Date: March 26, 2025

End Date: April 10, 2025

Description:

- Design and execute benchmark tests to evaluate SLRU, 2Q, and ARC caching in MySQL using tools like TPC-H, SysBench, and custom workloads. Simulate read-heavy, write-heavy, and mixed workloads to assess cache efficiency.

- Compare these caching strategies against MySQL's default (InnoDB Buffer Pool) by measuring: Cache hit rate (queries served from cache), Query execution time (response time improvements), Memory consumption (RAM usage per strategy), CPU utilization (computational overhead).

- Conduct stress tests under high concurrency, analyze performance trends, and identify bottlenecks. Suggest optimizations and visualize results with graphs and comparative tables.

- Evaluate scalability by testing cache performance under increasing database sizes and workload intensities.

Deliverables:

- Performance Evaluation Report: A comprehensive document summarizing experimental results, including benchmark statistics, cache hit rates, and execution time comparisons.
- Graphical Analysis: Graphs, tables, and charts showcasing the differences in performance between SLRU, 2Q, ARC, and MySQL's default caching mechanisms.
- Stress Test Findings: Documentation on caching performance under high load, concurrency conditions, and memory constraints.
- Optimization Recommendations: Suggestions for further improving cache efficiency based on experimental observations.

Person(s) in Charge: Sasank Sribhashyam, Venkat Tarun Adda**, Chandana Sree Kamasani**

Task 5: Comparative Study and Discussion of Results

Start Date: April 11, 2025

End Date: April 20, 2025

Description:

- Compare the performance of SLRU, 2Q, and ARC against MySQL's existing caching mechanisms. Analyze key performance metrics such as cache hit rate, query execution time, memory consumption, and CPU overhead.
- Identify trade-offs between caching strategies, including their adaptability to different workload types (read-heavy, write-heavy, mixed).
- Provide recommendations on the best caching strategy for different MySQL workloads.
- Discuss real-world applicability and suggest potential optimizations for future research.

Deliverables:

- Comparative Analysis Report: A concise document highlighting key performance differences.
- Graphical Summary: Tables and charts visualizing results.
- Performance Trade-offs Report: Summary of the best caching strategies for different workload conditions.

- Use-Case Recommendations: Insights on when to use SLRU, 2Q, or ARC for maximum efficiency.

Person(s) in Charge: Sasank Sribhashyam, Venkat Tarun Adda**,** Chandana Sree Kamasani

Task 6: Final Report and Presentation Preparation

Start Date: April 21, 2025

End Date: May 1, 2025

Description:

- Compile all research findings, experimental results, and insights into a well-structured final report.
- Ensure the report includes comprehensive documentation of the caching strategies implemented, benchmarking results, comparative analysis, and conclusions.
- Prepare a professional presentation summarizing the project's objectives, methodologies, key results, and recommendations.
- Review and refine the report and presentation to ensure clarity, accuracy, and completeness.

Deliverables:

- A final project report formatted according to academic standards.
- A presentation slide deck summarizing the project's key findings and conclusions.
- A final review of all project deliverables to ensure completeness and quality.

Person(s) in Charge: Sasank Sribhashyam, **Venkat Tarun Adda, Chandana Sree Kamasani**

6) References

[1] J. Ni, S. Xie, Z. Li and C. Jia, "Optimization Design Method of Cache Extension in MySQL Database," 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, 2018, pp. 2295-2299, doi: 10.1109/.

[2] K. Hurt and B. K. Lee, "Proposed enhancements to fixed segmented LRU cache replacement policy," 2013 IEEE 32nd International Performance Computing and Communications Conference (IPCCC), San Diego, CA, USA, 2013, pp. 1-2, doi: 10.1109/PCCC.2013.6742754.

[3] Gao, Hongliang and Chris Wilkerson. “A Dueling Segmented LRU Replacement Algorithm with Adaptive Bypassing.” (2010).

[4] Theodore Johnson and Dennis Shasha. 1994. 2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 439–450.

[5] Nimrod Megiddo and Dharmendra S. Modha. 2003. ARC: a self-tuning, low overhead replacement cache. In Proceedings of the 2nd USENIX conference on File and storage technologies (FAST'03). USENIX Association, USA, 9.

[6] MYSQL Documentation - Query Caching