GPredictor: A Graph Embedding-Based System for Concurrent Query Performance Prediction

1. Introduction

Query performance prediction is crucial for database systems as it helps meet service level agreements (SLAs) and enhances the efficiency of applications such as transaction scheduling, parameter tuning, and progress monitoring. Traditional performance prediction methods are mostly designed for single queries and cannot effectively handle the performance prediction of concurrent queries.

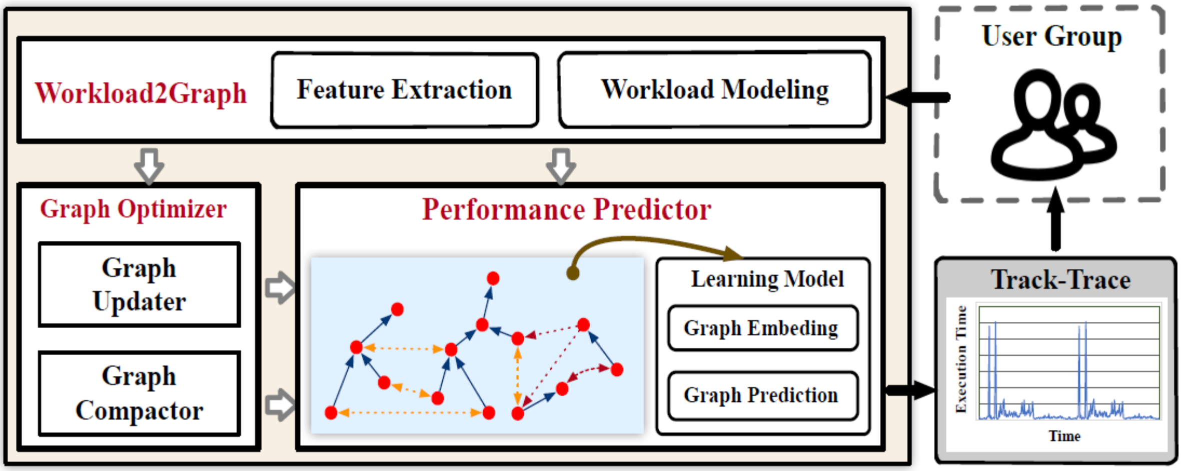
2. Problem Background

Existing methods for concurrent query performance prediction have the following issues:

- Inability to capture the correlations such as data sharing and resource competition between concurrent queries.
- Neglecting resource conflicts during query execution.
- Lack of dynamic workload handling capabilities, making them unsuitable for frequent query changes.

3. The GPredictor System

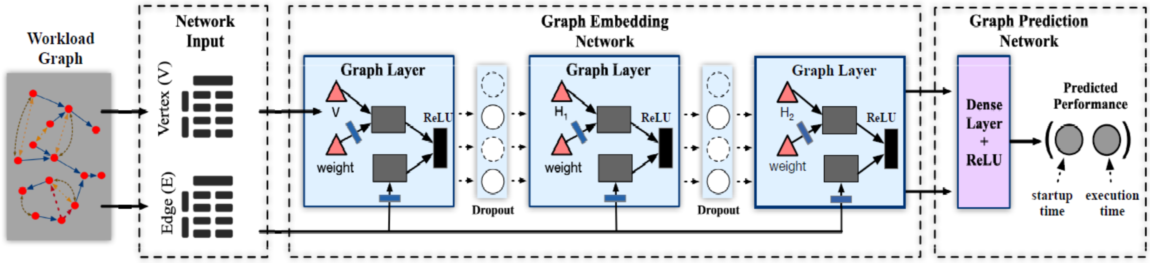
GPredictor is a performance prediction system based on graph embedding technology, designed specifically for concurrent and dynamic workloads. The system architecture, as shown in the diagram, includes the following main modules:



- **Workload2Graph**: Converts query workloads into graph models, extracting operator features and correlations.
- **Performance Predictor**: Uses graph embedding and deep learning models for performance prediction.
- **Graph Optimizer**: Optimizes the graph model, dynamically updating and compressing the graph structure to improve prediction efficiency.

4. Implementation Details

The following diagram provides a detailed implementation method of GPredictor:



Explanation

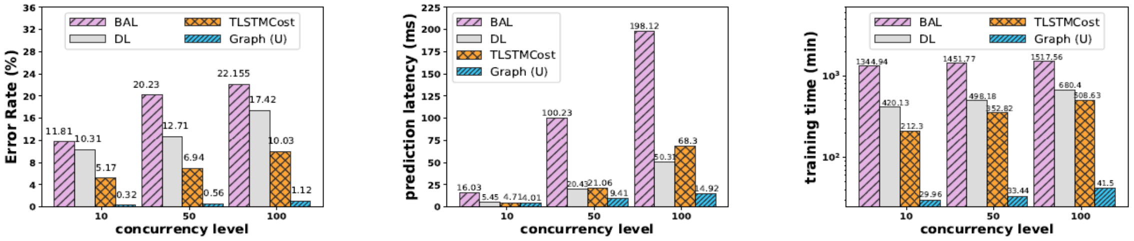
1. **Workload Graph**
   - This part shows how to convert the query plan into a graph model, with vertices (Vertex) representing the operators in the query and edges (Edge) representing the correlations between the operators.
2. **Network Input**
   - Includes vertex matrix (Vertex Matrix, V) and edge matrix (Edge Matrix, E). These matrices serve as the network input, representing the structure and features of the graph.
3. **Graph Embedding Network**
   - This network contains multiple graph layers (Graph Layer), each processing the features of vertices and edges through weighted aggregation and activation functions (such as ReLU), embedding each vertex into a vector representation.
   - Dropout layers are used to prevent overfitting.
4. **Graph Prediction Network**
   - Uses a three-layer perceptron to predict query performance, including an input layer, hidden layer, and output layer.
   - Finally, it outputs predicted performance metrics for each operator, such as startup time and execution time.

5. Main Contributions

1. **Graph Embedding Model**: Proposes the first system based on graph embedding technology for concurrent query performance prediction, capturing complex correlations between queries.
2. **Workload Graph Model**: Constructs a graph model reflecting the features and correlations of query operators, including data sharing and resource competition.
3. **Graph Embedding and Prediction**: Uses graph embedding methods to learn performance-related features and employs deep learning models to predict query performance.
4. **Dynamic Adaptation**: Proposes a graph compression algorithm to reduce graph size, lower prediction overhead, and enhance prediction efficiency.
5. **Experimental Validation**: Conducts experiments under multiple workloads and environments, demonstrating the high accuracy and superior performance of GPredictor.

6. Experiments and Results

The following figure shows the performance comparison of GPredictor with three state-of-the-art methods (BAL, DL, TLSTMCost) under three different concurrency levels (10, 50, 100), analyzing prediction error, prediction latency, and training time:



(a) Prediction Error (b) Prediction Latency (c) Training Time

Figure 7: Performance Comparison with Baselines on the JOB Workloads on PostgreSQL.

Explanation of Results

- **Prediction Accuracy**: GPredictor shows high accuracy in query performance prediction across different datasets and scenarios.
- **Prediction Efficiency**: Dynamic updating and graph compression techniques enable GPredictor to quickly respond to query changes and reduce prediction latency.
- **Resource Utilization**: Effectively captures and manages resource competition, improving system resource utilization and query processing efficiency.

7. Conclusion

GPredictor offers an innovative and efficient solution for concurrent query performance prediction, leveraging graph embedding technology and dynamic optimization methods to significantly enhance prediction accuracy and efficiency. The system encodes query features using graph structure models and predicts using deep learning models, supporting dynamic workloads and graph compression algorithms to ensure adaptability and scalability.

Experimental results indicate that GPredictor outperforms existing state-of-the-art methods in terms of accuracy and efficiency. Its ability to accurately predict query execution time and handle dynamic workloads makes it a powerful tool for database management and optimization, with broad application prospects and practical value.

In summary, GPredictor not only sets a new standard for query performance prediction but also opens new possibilities for advanced database management and optimization. Its innovative approach and significant results highlight its extensive potential and major impact in this field.