ML System Optimization (S1-25_AIMLCZG516) - Assignment 1

Group -10

Sl No	Name	BITS ID	% Contribution
1	Ritankar Dey	2024ac05526	100 (A1, A2 and A3)
2	**Somnath Paul**	**2024ac05549**	**100 (A1, A2 and A3)**
3	Mir Shoaib Ali	2024ac05244	100 (A1, A2 and A3)
4	Arjun Ramesan	2024ac05026	100 (A1, A2 and A3)
5	Akkireddy Pranith	2024ac05473	100 (A1, A2 and A3)

A1 – Literature Survey

Abstract

Gradient Boosted Decision Trees are widely used in machine learning due to their strong predictive performance; however, their scalability remains a challenge when applied to large datasets and distributed environments. XGBoost (eXtreme Gradient Boosting) addresses these challenges through system-level optimizations that enable efficient parallel and distributed training. This report presents a structured analysis of XGBoost from a machine learning systems perspective. A literature survey is conducted to examine existing scalable boosting techniques, followed by a formal problem formulation focusing on performance scalability. A cost model is then developed to analyze computation, communication, and synchronization overheads in distributed XGBoost training. The study highlights key scalability bottlenecks and provides insights into the trade-offs that limit linear speedup in large-scale deployments.

1. Introduction

Modern machine learning applications increasingly rely on large-scale data and distributed computing platforms to meet performance and scalability requirements. Ensemble learning techniques, particularly Gradient Boosted Decision Trees (GBDTs), have achieved significant success across a wide range of predictive tasks. Despite their effectiveness, traditional implementations of gradient boosting face limitations in handling large datasets due to computational complexity, memory constraints, and limited parallelism.

XGBoost, short for eXtreme Gradient Boosting, is a system-optimized implementation of gradient boosting that addresses these challenges through algorithmic approximation and system-aware design. This report analyzes XGBoost from a machine learning systems and operations (MLSysOps) perspective. The objectives are threefold: (i) to survey existing research on scalable gradient boosting, (ii) to formally define the performance scalability problem in distributed XGBoost training, and (iii) to develop a cost model that explains the impact of computation and communication overhead on training efficiency.

2 Background and Motivation

Gradient Boosted Decision Trees (GBDTs) are widely recognized for their strong predictive performance in classification, regression, and ranking tasks. Due to their ability to combine multiple weak learners into a powerful ensemble, boosting-based models have achieved state-of-the-art results in both academic benchmarks and industrial applications. However, as data volumes and feature dimensions continue to grow, traditional implementations of gradient boosting face significant scalability challenges. Training becomes increasingly time-consuming, memory-intensive, and difficult to deploy efficiently in distributed environments. These limitations have motivated research efforts aimed at redesigning boosting algorithms to better align with modern hardware architectures and large-scale data processing requirements.

---

3 Scalability Challenges in Traditional Boosting

A fundamental obstacle to scaling gradient boosting lies in its inherently sequential learning process, where each tree is constructed based on the residual errors of previously trained trees. This dependency restricts straightforward parallelization across boosting iterations. While limited parallelism is possible during tree construction, earlier GBDT systems largely focused on algorithmic accuracy rather than system-level efficiency. As a result, issues such as inefficient memory access, poor cache utilization, excessive disk I/O, and high computational overhead often dominate training time. Research has shown that these system bottlenecks, rather than pure algorithmic complexity, are the primary factors limiting the scalability of boosting methods on large datasets.

---

4 Histogram-Based Split Finding and Approximation

To address the computational inefficiency of exact split evaluation, recent work has proposed approximate split finding techniques based on feature quantization and histogram construction. The Weighted Quantile Sketch method provides an efficient mechanism for summarizing feature distributions using compact data structures, enabling approximate yet accurate split selection. By discretizing continuous feature values into bins, histogram-based approaches significantly reduce both computation and memory requirements. More importantly, these summaries can be computed independently across data partitions and later merged, making them well-suited for parallel and distributed environments. This shift from exact to approximate split finding represents a key enabler for Xtreme gradient boosting without substantial loss in model accuracy.

---

5 System-Level and Distributed Optimizations

Beyond algorithmic approximation, system-oriented optimizations play a critical role in improving the performance of large-scale boosting frameworks. Research emphasizes the importance of cache-aware data layouts, column-oriented storage, and efficient memory access patterns to fully utilize modern multi-core architectures. In distributed settings, training data is partitioned across multiple workers, each computing local statistics required for tree construction. These local summaries are aggregated using collective communication operations to determine global split decisions. While this approach reduces computation time, it introduces communication and synchronization overhead, particularly as cluster size increases. Consequently, overall system performance becomes highly dependent on network bandwidth, synchronization frequency, and load balancing across workers.

---

6 Limitations and Research Gap

Despite significant advancements in Xtreme gradient boosting, existing approaches still face inherent limitations. The sequential dependency between boosting iterations restricts full parallelism, placing an upper bound on achievable speedup. Additionally, synchronous aggregation of histogram information can lead to performance degradation in the presence of stragglers, network latency, or uneven data distribution. As datasets and computing clusters continue to scale, communication overhead increasingly dominates computation time. These challenges highlight an important research gap in understanding and modeling the trade-offs between computation, communication, and system efficiency in distributed gradient boosting frameworks. Addressing these limitations motivates further investigation into performance modeling and optimization strategies, which forms the basis of the problem formulation discussed in the subsequent section.

References

1. **Chen, T., & Guestrin, C. (2016).**
   XGBoost: A Scalable Tree Boosting System.
   Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD ’16).
   https://arxiv.org/abs/1603.02754
2. **Chen, T., & Guestrin, C. (2016).**
   Weighted Quantile Sketch for Efficient Data Analysis.
   https://arxiv.org/abs/1602.02304
3. **Chen, T., He, T., Benesty, M., Khotilovich, V., Tang, Y., & Cho, H. (2017).**
   GPU Acceleration for Large-Scale Tree Boosting.
   https://arxiv.org/abs/1706.08359

A2 – Problem Formulation

1 Problem Definition

The objective of this work is to analyze the performance scalability of distributed XGBoost training in large-scale data processing environments. Although XGBoost introduces several system-level optimizations to enable parallel and distributed gradient boosting, its overall performance is influenced by multiple interacting factors, including computation cost, communication overhead, and synchronization constraints. The problem addressed in this section is to formally characterize how these factors affect training time and speedup as the number of worker nodes and dataset size increase. In particular, the focus is on understanding the computation–communication trade-offs that arise during histogram-based tree construction in distributed settings and identifying the conditions under which scalability deviates from ideal linear behavior.

---

2 System Model and Assumptions

We consider a distributed XGBoost training environment consisting of P homogeneous worker nodes connected through a shared communication network. The training dataset contains N data samples with D features and is horizontally partitioned across the workers, such that each worker processes approximately N/P samples. Feature values are discretized into B histogram bins to enable approximate split finding. Tree construction follows a level-wise growth strategy, and a fixed number of boosting iterations T is assumed. During training, each worker independently computes local histogram statistics for candidate splits, after which these statistics are aggregated synchronously across workers to determine global split decisions. The cost of local computation, network communication, and synchronization is assumed to be non-negligible and contributes to the overall training time.

---

3 Performance Metrics

To evaluate the scalability and efficiency of distributed XGBoost training, the following performance metrics are considered. **Training time** is defined as the total wall-clock time required to complete a fixed number of boosting iterations. **Speedup** is measured as the ratio of the training time on a single worker to the training time on P workers, indicating the effectiveness of parallelization. **Scalability** reflects the ability of the system to reduce training time as additional workers are added, while maintaining a constant workload. Together, these metrics provide a quantitative basis for analyzing how computation and communication costs influence the performance of distributed XGBoost training.

4 Bottleneck Identification and Analysis Scope

Despite the parallelization strategies employed by XGBoost, several bottlenecks limit its scalability in distributed environments. First, the sequential dependency between boosting iterations restricts parallel execution across trees, imposing an upper bound on achievable speedup. Second, the synchronous aggregation of histogram statistics introduces communication overhead that grows with the number of workers, particularly in network-constrained settings. Third, load imbalance and straggling workers can delay synchronization, further reducing parallel efficiency. These bottlenecks suggest that training performance is increasingly dominated by communication and synchronization costs as system scale grows. Consequently, this work focuses on modeling the relative contributions of computation and communication to total training time, forming the basis for the cost analysis and performance modeling presented in the subsequent section.

Design of Distributed XGBoost Training

The distributed design of XGBoost follows a data-parallel architecture in which the training dataset is horizontally partitioned across multiple worker nodes. Each worker independently processes its local data partition to construct histogram-based summaries for candidate feature splits. These local histograms are then aggregated using synchronous collective communication operations to determine globally optimal split decisions at each tree node.

Tree construction proceeds in a level-wise manner, ensuring consistency across workers while enabling parallel computation within each level. Although computation is distributed, synchronization is required at every split decision, introducing communication overhead and limiting full parallelism. This design enables scalability for moderate cluster sizes but introduces performance trade-offs as system scale increases. The cost model developed in the subsequent section maps directly to this design by quantifying computation, communication, and synchronization components.

A3 – Cost Model and Performance Analysis

1 Overview of Cost Modeling

The performance of distributed XGBoost training is determined by the combined cost of local computation, inter-worker communication, and synchronization overhead. Although XGBoost enables parallel computation during tree construction, the presence of sequential boosting iterations and synchronous aggregation operations introduces inherent limits to scalability. In this section, a cost model is developed to analyze the total training time as a function of dataset size, number of workers, and system parameters. The objective is to identify how computation and communication costs interact and to explain why speedup deviates from ideal linear scaling in distributed environments.

---

2 Computation Cost

During each boosting iteration, XGBoost constructs decision trees by evaluating candidate splits using histogram-based methods. For a dataset of size *N* with *D* features and *B* histogram bins, the dominant computation cost arises from histogram construction at each tree node. In a distributed setting with *P* workers, each worker processes approximately *N/P* samples. As a result, the local computation cost per worker per boosting iteration can be approximated as proportional to:

$$T\_comp \propto (N / P) \times D \times B$$

This formulation captures the fact that histogram computation scales linearly with the number of samples processed locally and benefits from data parallelism across workers. As *P* increases, the computation cost per worker decreases proportionally, enabling faster local processing.

---

3 Communication Cost

In addition to local computation, distributed XGBoost requires workers to exchange information in order to determine global split decisions. After computing local histograms, workers participate in synchronous aggregation operations to combine histogram statistics across all workers. The communication cost depends on the size of histogram data exchanged, the number of workers, and network characteristics. This cost can be modeled as:

$$T\_comm \propto D \times B \times \log(P)$$

where the logarithmic factor reflects the use of collective communication operations, such as AllReduce, commonly employed in distributed training frameworks. Unlike computation cost, communication cost does not decrease proportionally with additional workers and can become a dominant factor as *P* increases.

---

4 Synchronization and Sequential Overhead

Beyond computation and communication, XGBoost training includes synchronization and sequential components that limit parallel efficiency. Boosting iterations are inherently sequential, as each tree depends on the residuals produced by previous trees. Additionally, synchronization is required at each tree level to ensure that all workers agree on split decisions before proceeding. These factors introduce a sequential overhead that cannot be parallelized and can be represented as a constant or slowly varying term:

$$T\_seq = \alpha$$

where *α* captures synchronization delays, straggler effects, and other non-parallelizable components of training.

---

5 Total Training Time Model

Combining computation, communication, and sequential overhead, the total training time for *T* boosting iterations in a distributed XGBoost setup can be expressed as:

$$T\_total = T \times (T\_comp + T\_comm) + T\_seq$$

This model highlights the trade-off between computation and communication. While increasing the number of workers reduces computation time, it also increases communication and synchronization costs, leading to diminishing returns in speedup.

---

6 Speedup and Scalability Analysis

Speedup is defined as the ratio of training time on a single worker to training time on *P* workers:

$$\text{Speedup}(P) = T\_total(1) / T\_total(P)$$

In the ideal case, speedup increases linearly with *P*. However, due to communication overhead and sequential components, the observed speedup of XGBoost deviates from linear scaling as *P* increases. When communication and synchronization costs dominate computation, adding more workers yields limited performance gains. This behavior aligns with classical parallel performance models, where non-parallelizable components impose an upper bound on achievable speedup.

---

7 Implications and Discussion

The proposed cost model explains why distributed XGBoost achieves near-linear speedup at small to moderate cluster sizes but experiences diminishing returns at larger scales. It also highlights the importance of balancing computation and communication when designing scalable training systems. Factors such as efficient histogram compression, optimized collective communication, and load balancing play a crucial role in improving scalability. This analysis provides a foundation for evaluating optimization strategies and system configurations.

Conclusion

This report presented a structured analysis of XGBoost from a machine learning systems perspective, focusing on its scalability in distributed environments. Through a literature survey, key algorithmic and system-level techniques enabling scalable gradient boosting were identified. A formal problem formulation was then developed to characterize performance bottlenecks, followed by a cost model that captures the interaction between computation, communication, and synchronization overheads.

The analysis demonstrates that while XGBoost achieves near-linear speedup at small to moderate scales, communication and sequential dependencies impose fundamental limits on scalability as cluster size increases. These findings highlight the importance of balancing computation and communication in the design of scalable machine learning systems. Future work may explore asynchronous training strategies, communication-efficient aggregation methods, or hardware-accelerated implementations to further improve scalability.