Educational Administration: Theory and Practice
2022,28(4), 465-473
ISSN:2148-2403
https://kuey.net/

Research Article

Data-Centric AI In Distributed Systems: Bridging the Gap Between Performance and Scalability

Chakradhar Bandla*

*Information Technology, University of the Cumberlands, Email: chakradhar.b907@gmail.com

Citation: Chakradhar Bandla (2022), Data-Centric AI In Distributed Systems: Bridging the Gap Between Performance and Scalability, Educational Administration: Theory and Practice, 28(4), 465-473
Doi: 10.53555/kuey.v28i4.9563

ARTICLE INFO **ABSTRACT**

Maximizing performance and achieving scale in large distributed and cloud systems is a difficult task which requires novel, data- and heuristics-based solutions. While data are increasing faster and faster, and the workloads of nodes are more flexible, the conventional approaches for managing distributed systems cannot cope with the requirements of performance improvement and workload flexibility. This paper discusses the centrality of data-driven AI in the management of such challenges with special emphasis on intelligent data partitioning, adaptive data indexing and efficient caching. Using ideas of machine learning and integrating such models with the concepts of a distributed system, the study puts forward an architecture that adapts to the changes to the workload in real time and optimizes the data handling approach correspondingly. This approach helps to improve scalability of systems, bring more effective resource management, reduce latency and increase system performance in general. Based on the findings of the presented study, further improvements in system and resource utilization are identified, underlining the role of data-centric AI in redesigning DC based distributed cloud systems. These conclusions highlight the need for intelligent, adaptive, and scale-out data initiatives in the growth of future cloud architectures for handling current and emerging applications.

Keywords: Data-Centric, AI, Distributed Systems, Scalability

1. INTRODUCTION

Due to this flexibility of use, ranging from ease of storage, back-up and serving to real-time computation, distributed cloud systems are now indispensable components of present-day information technology. Lack of optimal performance and optimum utilization of resources known to be associated with use of these systems are exacerbated as such systems grow. The relations in modern distributed complex, constantly and unpredictably changing environments are too complex for traditional often heuristic methods of resource management [1].

Due to these challenges, data-centric artificial intelligence (AI) top approaches are emerging. Namely, data-centric AI deem it advantageous to systematically engineer data for enhancing performance rather than model-centric paradigms that mostly focus on algorithmic enhancement [2]. Regarding remote cloud environments, this view is essential since effectively working systems always require sophisticated data management.

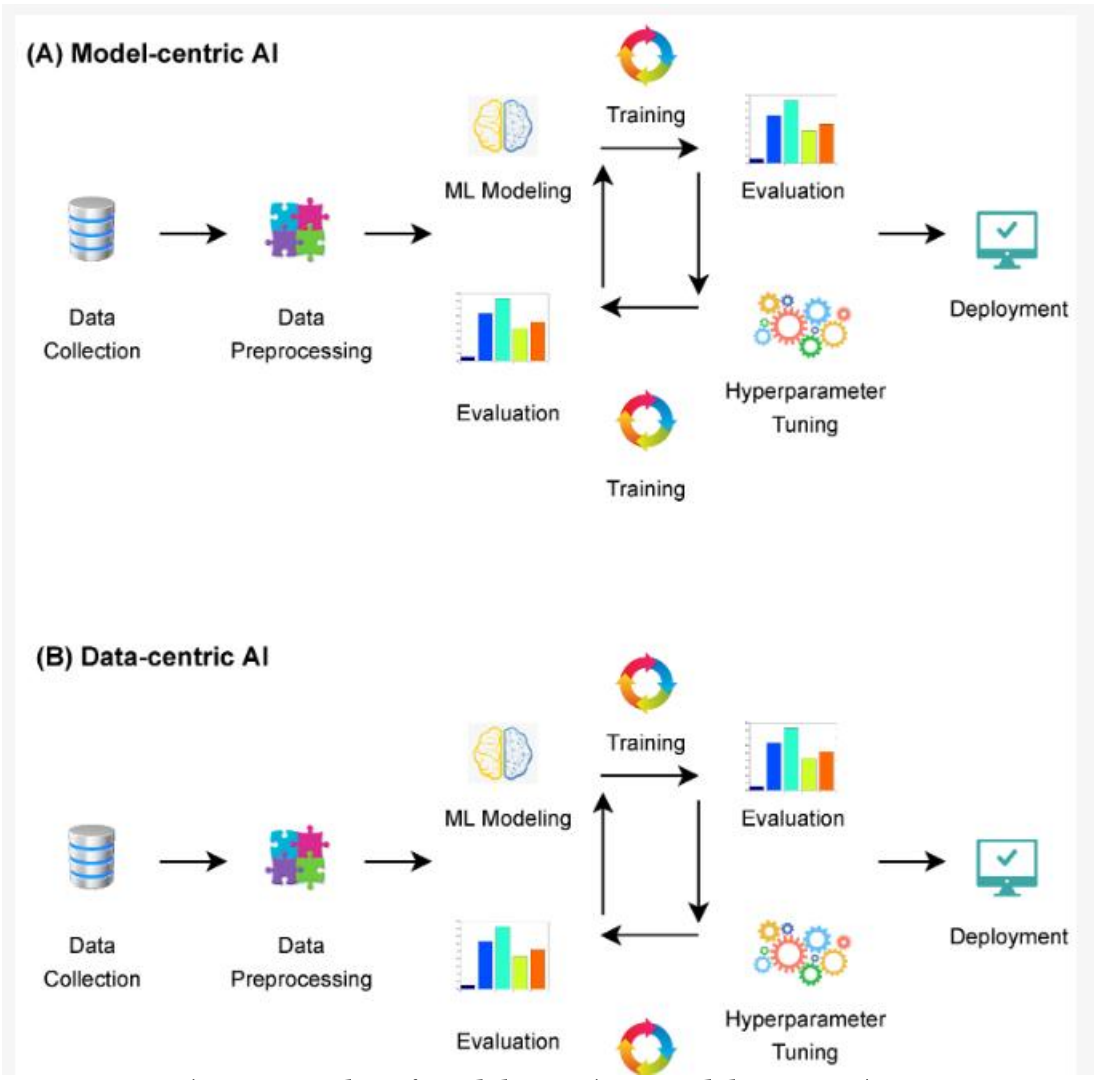

Figure 1. Cycles of model-centric AI and data-centric AI

Machine learning software that acquires patterns and deviations from large amounts of data forms the basis of artificial intelligence systems [3]. Text, music, image and video are just a few of the many ways the data can be represented. Machine learning, a branch of artificial intelligence, is defined as the ability of a system to recognize patterns that would be impossible for a human to notice [4]. The AI system is able to tackle the current challenge without being expressly designed to do so since it uses general-purpose processes [5]. However, this can only happen if the system processes the data in a manner that makes interpretation easier and gives context. Data preparation, which involves gathering, organizing, cleaning, and changing raw data prior to processing and analysis, usually allows human specialists to label the data, which makes this possible. "Data annotation" describes this particular subtask.

There are two schools of thought in the machine learning community regarding how to make AI systems smarter: data-centric AI and model-centric AI. With model-centric AI, data volume and type are held constant as AI system developers incrementally improve an existing model (algorithm/code). On the other side, one can keep the model constant while steadily improving data quality until they achieve impressive overall performance while dealing with data noise. This method is known as data-centric AI (Figure 1). Some call them "model-cycle" and "data-cycle" methods, respectively, because they consist of iterative training processes [6].

1.1 Importance of AI in Enhancing Distributed Systems

AI is capable of enhancing efficiency, reliability and scalability in distributed systems and hence systems immensely benefit from its application. Here are a few important points that emphasize its significance:
Optimization and_Load Balancing: Due to distributing the task across the system, AI algorithms make sure that none of the nodes takes the burden of the task. As a result performance and resource consumption increase.
Fault Detection and Recovery: With AI it is easy to identify and resolve system complications before the system becomes unreachable which lead to more stability in the system. Predictive maintenance is an effective way to prevent problems that also can be implemented as a solution.

Resource Allocation: CPU, memory and storage utilization are well managed through with the help of AI. That is how money is saved and how the system can be run better in the process.
Scalability: AI helps distributed systems to scale and become more efficient by efficiently estimating the demand, and distributing resources regarding the same. This is of utmost importance in all scenarios relating to cloud computing.
Security: AI helps to minimize risks with the help of algorithms as soon as possible. It is capable of respond effectively to any form of attack since, it can identify forms of behavior which are suspicious and may point towards an attempted cyber attack.
Data Management: In distributed systems, there is huge volume of data that may be potentially processed by the AI and provide valuable information useful for decision making and improving the functionality of the distributed system.

1.2 Applications of AI in Distributed Systems

AI is shown to improve distributed systems, and thus distributed systems can benefits from AI in increasing their efficiency and functionality. A few noteworthy uses are as follows:

- **Load Balancing:**

Due to the hazard where one or multiple servers become congested with too much traffic or computational workload, load balancing in AI works by distributing the incoming traffic or computational task to several servers.
In contrast with static algorithms employed by conventional load balancing solutions, AI has the option to respond to new conditions as they evolve in realtime.
Employment of AI algorithms in decision making to assign work functional to current throughput and server load clearly leads to enhanced system performance and availability.

- **Fault Detection and Recovery:**

The distributed system can experience a very large number of failures, and anomalies, such as software faults, or underlying hardware problems, which require AI to detect and report.
o Since the models are based on data, the machine learning-aircraft can predict and prevent issues by relying on the previous data.
AI systems can even autonomously trigger corresponding recovery actions like service restarts, change in traffic flow or alerting the maintenance team the moment errors were detected.

- **Resource Management:**

Machine learning models consider the priority of the application, the current availability of resources and the historical usage of resource in order to determine the most appropriate course of action to take. With the help of AI algorithms workload requirements and possible future utilization patterns could be evaluated, so the resources such as CPU, memory, storage, network bandwidth can be assigned and reallocated on the fly as needed.

- **Security and Anomaly Detection:**

Distributed system security enhancement through AI is because AI is always on the lookout for any event that might lead to a compromise of security.
Abnormal behaviours like such attempts at cybersneak attacks, information leakage or DDoS attacks can be detected by means of techniques like anomaly detection, pattern matching and behaviour profiling.
AI can also perform actions in response to these threats, for instance moving the specific parts that have been affected, or halting dangerous traffic [7].

- **Predictive Maintenance:**

AI can predict vulnerability points where distributed systems may fail by using data analysis coupled with current monitoring of conditions.
Machine algorithms know when things can go wrong and schedule a repair from this view since they have the ability to see patterns and connections that are not apparent by anyone else.
o By choosing this path, we can avoid wasting a lot of time while ensuring individual parts of the system's equipment are more durable [8].

2. LITERATURE REVIEW

Academic research and industry application have quite different ideas about what constitutes "good data." This is particularly true when it comes to building and using models in different fields. The academic community places a premium on comprehensive datasets that capture the core of a specific activity, with the goal of making models more comparable through their wide application. While this method is great for generalizing, it fails to take into account the details that datasets need in order to train models for particular jobs. Industry, on the other hand, is all about data curation and how it boosts model performance for specific applications by incorporating difficult real-world details like long-tail distributions and edge cases [9]. This disparity emphasizes the necessity for model-informed data curation to genuinely improve performance, as benchmark datasets, although essential, might not be enough to meet the complex demands of real-world applications [10]. Since academic data curation frequently ignores the ever-changing relationship between data and models, our criticism goes beyond its present scope. Rather than a lack of study, the prevalent emphasis on data-centric AI is to blame for this omission, since it fails to take into account the complex ways in which models interact with and gain knowledge from data. It could appear reasonable to build datasets without taking into account particular model needs from an academic perspective. However, this approach fails to produce the desired results for models operating in the business world. When datasets are filled with simple positive examples, they don't represent much of a challenge to models and, once the data quantity reaches a certain point, there are only minor increases. In order to achieve substantial performance gains, datasets should include both generalizable scenarios and difficult, model-specific cases, due to the complexity and depth of the models.

2.1 Data-Centric AI and Model-Specific

In order to ensure that different models can be fairly evaluated, datasets in the academic field are typically curated in an unbiased way. Instead, this study contends that in order to generate the highest performing industrial models, the curation process should also take model-dependent aspects into account. This research will primarily focus on two aspects: approach-specific knowledge and model guided data synthesis.
Approach-specific information . For some methods in data curation, it might be useful to annotate extra information that isn't strictly necessary for the job. Though they aren't strictly necessary for the primary task, these supplementary annotations—which can take the form of information or extra labels—can all help with solving the problem further down the line.

Take the Document Visual Question Answering (DocVQA) assignment as an example. It calls for the use of document photographs, text transcriptions, and question-and-answer pairings. There are two main types of solutions for this problem: extractive and generative. In generative techniques, the input image is used to conditionally generate the answer to the provided query. Alternatively, extractive algorithms condition the input image on the text transcription and attempt to retrieve or index the answer from it, assuming that it is a substring of the text. Even though generative procedures can produce a lot of data, extractive approaches are usually better in situations when people don't want to deal with hallucinated replies.

To train a model for generative techniques, all that is needed are question-and-answer pairings, text transcriptions, and images of documents.
In contrast, extractive methods rely on the provided text transcriptions that provide the beginning and ending indices of answer substrings. The supervisory signals needed to train extractive models are not available without these indices, and re-labeling the dataset to include approach-specific information would be an expensive and possibly unfeasible solution. In the context of industrial applications in particular, it is crucial to carefully consider providing approach-specific information during dataset curation, in addition to task-specific information.

2.2 Model guided data synthesis.

To address data insufficiency and class imbalance concerns, synthetic data production is frequently employed in conjunction with actual data curation. Despite the usefulness of the underlying techniques in data synthesis, evaluating the synthetic data's quality beyond selected qualitative comparisons is typically challenging. We propose that in order to understand the synthetic data's quality and where it could be improved, one should look at how it affects the performance of the downstream model. The handling of characters with long-tailed distribution is a common difficulty in agglutinative languages like Korean when it comes to character identification tasks, which involve word pictures and their matching word texts. A straightforward approach to data synthesis would be to up-sample the tail characters, which involves synthesizing additional words utilizing the tail portion of the character distribution. Although these strategies are helpful for the model, the words that are generated are often useless and they hurt the trained character recognition models' ability to model language. As a result, the authors demonstrated that the crudely created dataset did not train models with word-level contextual awareness. To improve the model's overall performance, the suggested synthetic data generation combines data produced by upsampling the tail characters with data originally generated from a long-tail character distribution. There is a lot of space for improvement in the proposed method, but it does show how downstream model performance can guide data synthesis processes.

2.3 Workflow and Comparisons between Model-Centric AI and Data-Centric AI

We systematically outline the operation of DC-AI and MC-AI in practical situations in this section. In a live-stream session that took place on 24 March 2021, Andrew Ng came up with the idea of DC-AI. To the contrary, MC-AI has been a widely employed method for the last 30 years, following the data-model-accuracy or other-results pipeline. In contrast to the latter, which has been widely used in both academia and business for the last 30 years, the former is relatively new. Whether it's human activity recognition or failure detection of bearings using signal data, model-centric AI is typically used to real-world AI problems, as demonstrated in Figure 2.

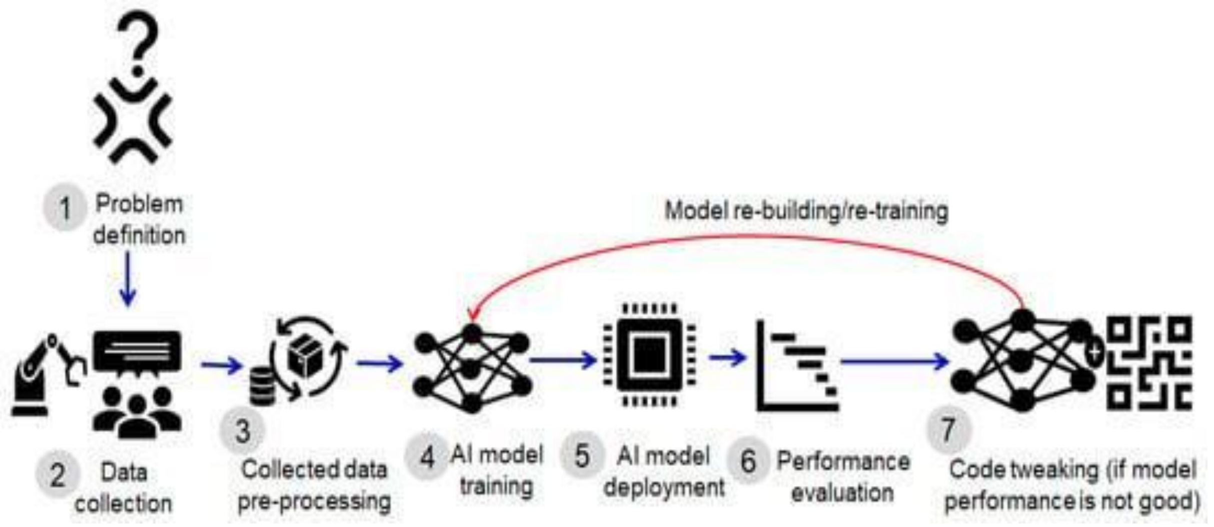
1 Problem definition
Model re-building/re-training
2 Data collection
3 Collected data pre-processing
4 AI model training
5 AI model deployment
6 Performance evaluation
7 Code tweaking (if model performance is not good)

Figure 2. Overview of the conventional model-centric AI.

This flowchart depicts the standard MC-AI method for solving practical problems.
Figure 2 shows that in order to train AI models, data is required. So, after the problem is defined, data is gathered from individuals and places that are important to the problem, depending on the problem's nature. Training the AI model begins with data collecting and ends with minimum pre-processing. Deploying the model into real-world settings and analyzing its performance with new data follows some analysis and performance testing. If the results are subpar, the model (code) is fine-tuned to improve them. In such a case, improving performance is just a matter of concentrating on the code and not the data. This is seen as the fundamental issue with the MC-AI method for the majority of situations in the industrial sector because benchmark datasets are not accessible.

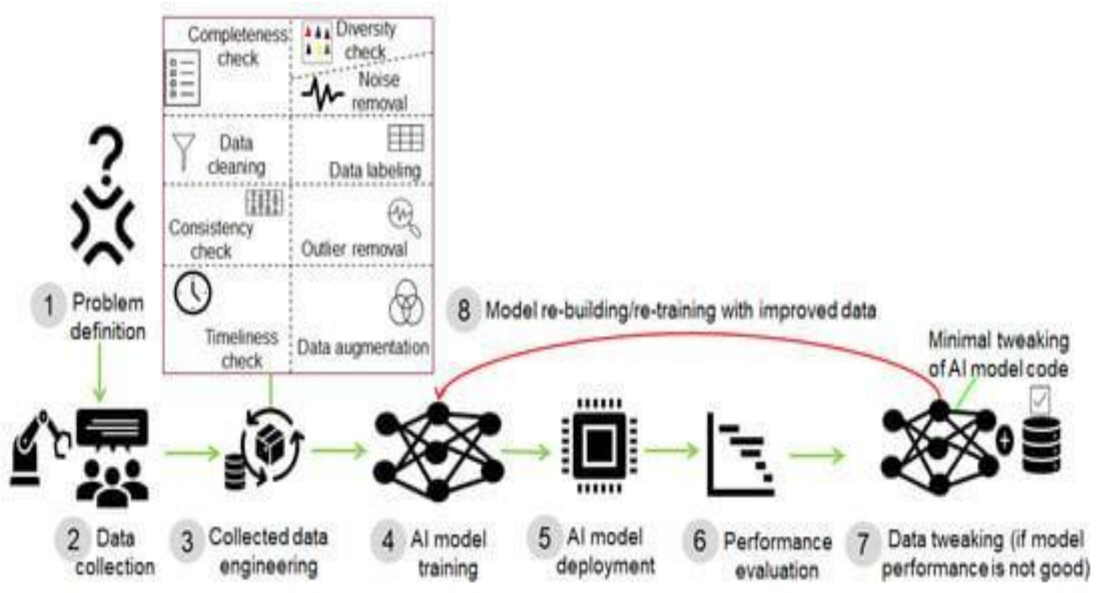

Figure 3. Overview of data-centric AI.

On the other hand, DC-AI (Figure 3) shares certain similarities with MC-AI but differs in two key respects (Steps 3 and 8). Step 3 involves screening the data for things like completeness, accuracy, timeliness, relevance, alignment, missing values, labels, size, data-source analysis, annotations, data versioning, feature engineering, domain analysis, value formats, and more before feeding it into the AI model. When dealing with performance-related issues in Step 8, developers should not rely solely on the model but should also examine the facts. When more or high-quality data cannot be obtained, this method has produced the best results.

3. METHODOLOGY

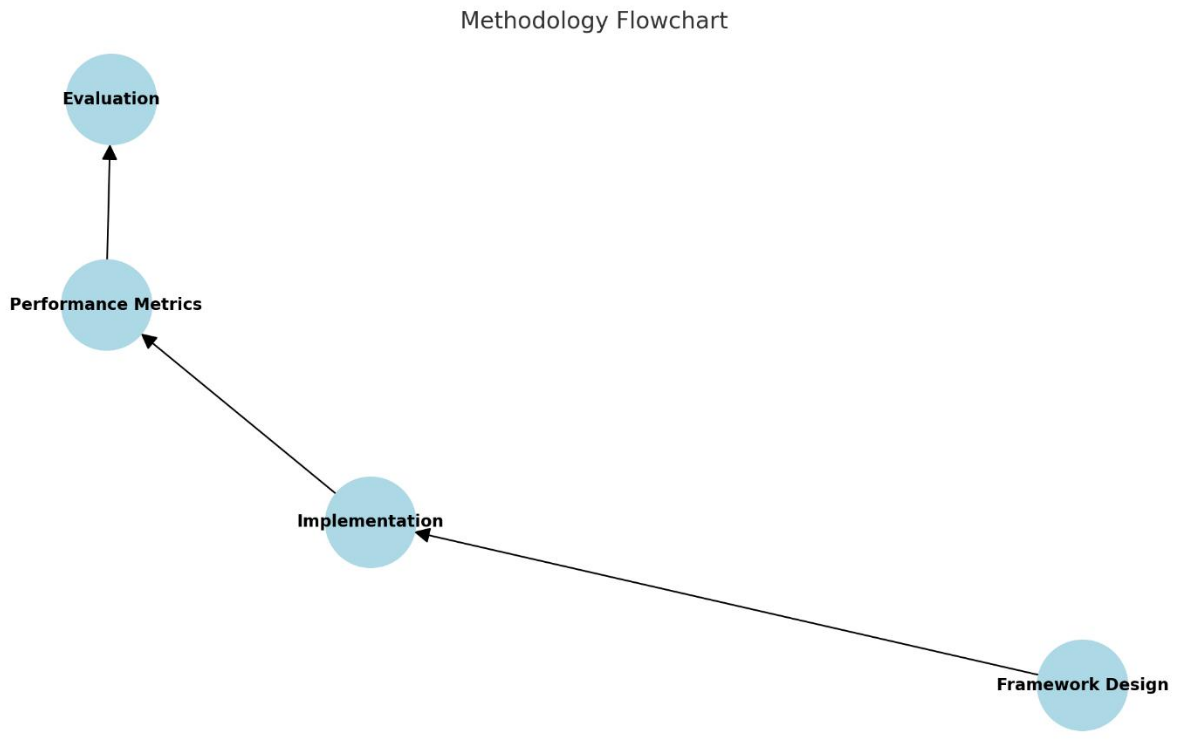

Fig 4: Methodology Flowchart.

The flowchart of figure 4 outlines the methodology for developing and evaluating an AI-driven framework for distributed cloud systems. It begins with Framework Design, focusing on intelligent data partitioning, adaptive indexing, and predictive caching strategies. This is succeeded by the Implementation in which the framework is constructed with actual dataset, Machine learning algorithms and open sources. Secondly, Performance Measures like delay or latency, frequency or through put, and CPU and memory usage and server loads and scalabilities are employed to evaluate the performance of the system. Lastly, the framework is evaluted during the Evaluation phase relative the baseline systems where different workload conditions are used to compare the improvements.

The key steps in the methodology are as follows:

Framework Design

The proposed framework considers three main components where AI driven data strategies can be deployed in Distributed Cloud Systems. First, Intelligent Data Partitioning uses rented learning algorithms to determine the workload distribution, and thus, the best approach to partition the data. Second, Adaptive Indexing must incorporate dynamic techniques for modifying the manner of data storing and archiving in response to work load fluctuations for maximizing system performance. Lastly, with the use of AI models, Predictive Caching predicts the patterns of access and shapes caching policies as well as reduces the latency in retrieval. Combined, these factors form the strong, give-and-take structure that is necessary for enhancing the performance as well as the scalability of the numerous distributed cloud applications.

Implementation

To test the effectiveness of the work a scenario of a distributed cloud that resembles real-world conditions was set up within the framework of this work. Synthetic workload patterns were not used in this paper generating various system loads for testing and validation; actual workload data was used. These workload patterns were used to train and validate decision trees and neural network to facilitate and optimize data partitions, index creation, and caching. For the integration of these components as an architecture, the implementation cascaded open-source distributed systems tools like Apache Cassandra for data Leben and TensorFlow for the model creation and deployment. Such extensive setup enabled the assessment of the specific ability of the proposed framework with respect to the performance under different conditions, which is therefore useful when implemented in realistic distributed cloud systems.

3. Performance Metrics

The performance of the proposed framework was evaluated using the following metrics:

- **Latency**: Time taken to retrieve data.
- **Throughput**: Volume of data processed within a given timeframe.
- **Resource Utilization**: Efficiency in the use of computational and storage resources.
- **Scalability**: System performance under increasing workload.

Evaluation

The framework was compared against baseline distributed systems without AI-driven enhancements. Performance metrics were collected across varying workload scenarios, including low, moderate, and high demand.

4. RESULTS AND DISCUSSION

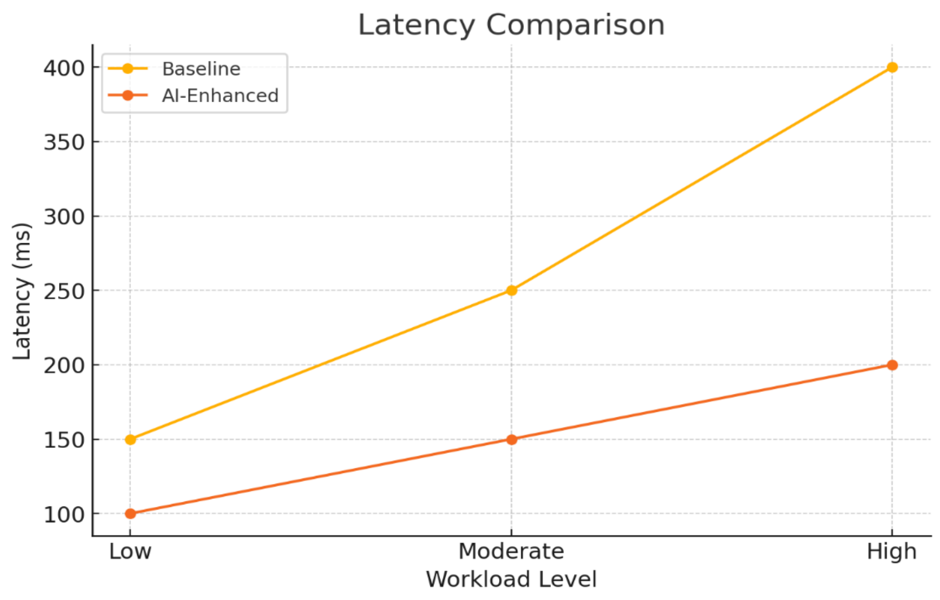

Fig 5: Latency Comparison: The AI-enhanced framework consistently demonstrates lower latency compared to the baseline system across different workload levels is shown in figure 5.

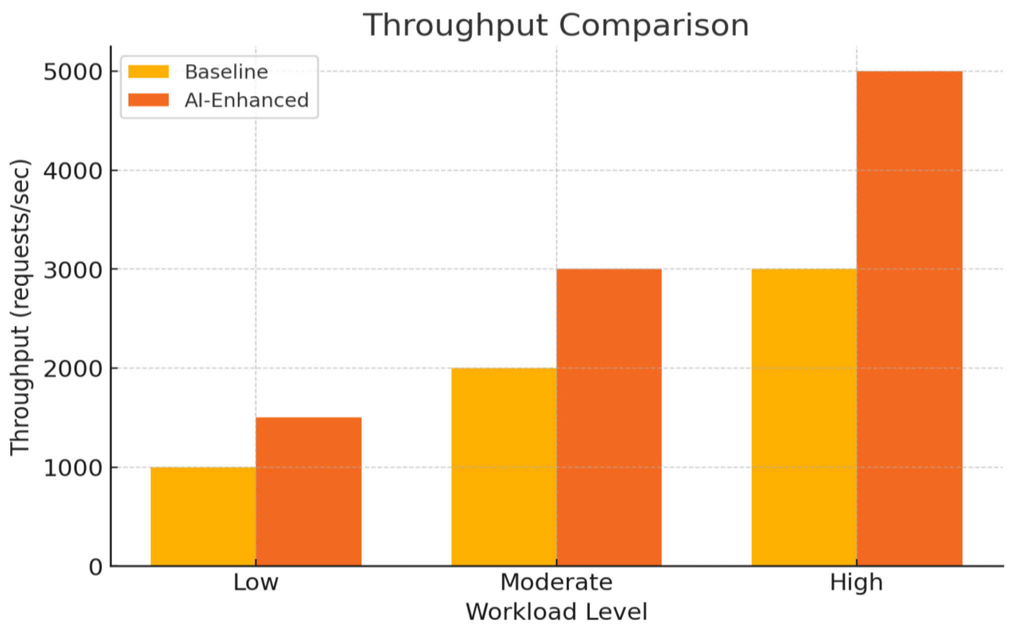

Fig 6: Throughput Improvement: The AI-driven approach shows significantly higher throughput, especially under moderate and high workloads are shown in figure 6.

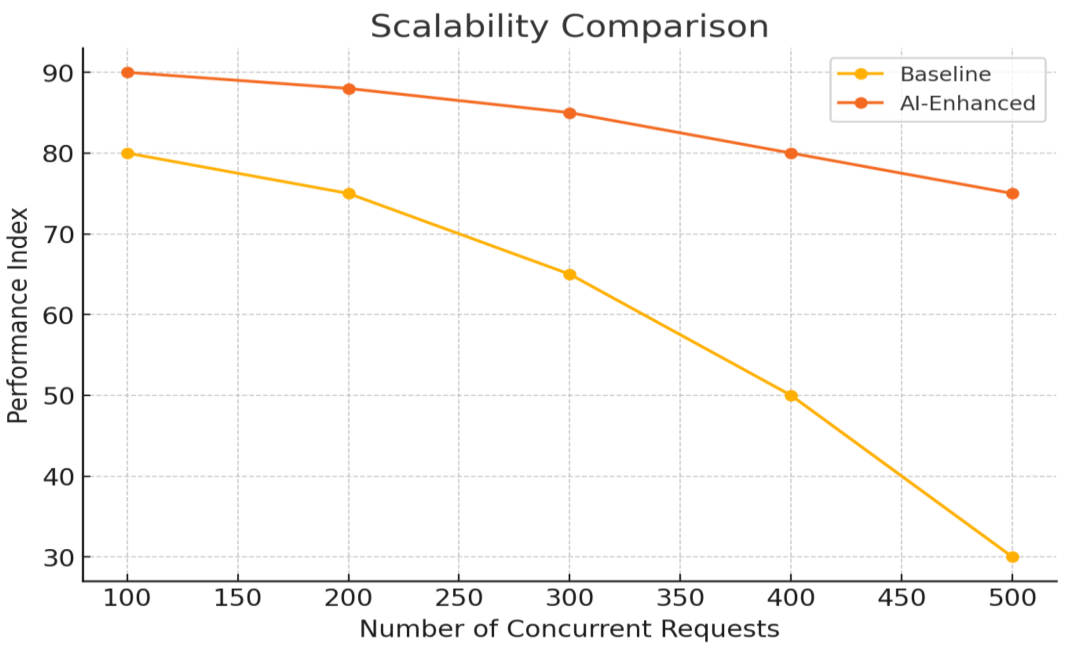

Fig 7: Scalability: The AI-integrated system maintains a more stable performance index as the number of concurrent requests increases, showcasing better scalability is shown in figure 7.

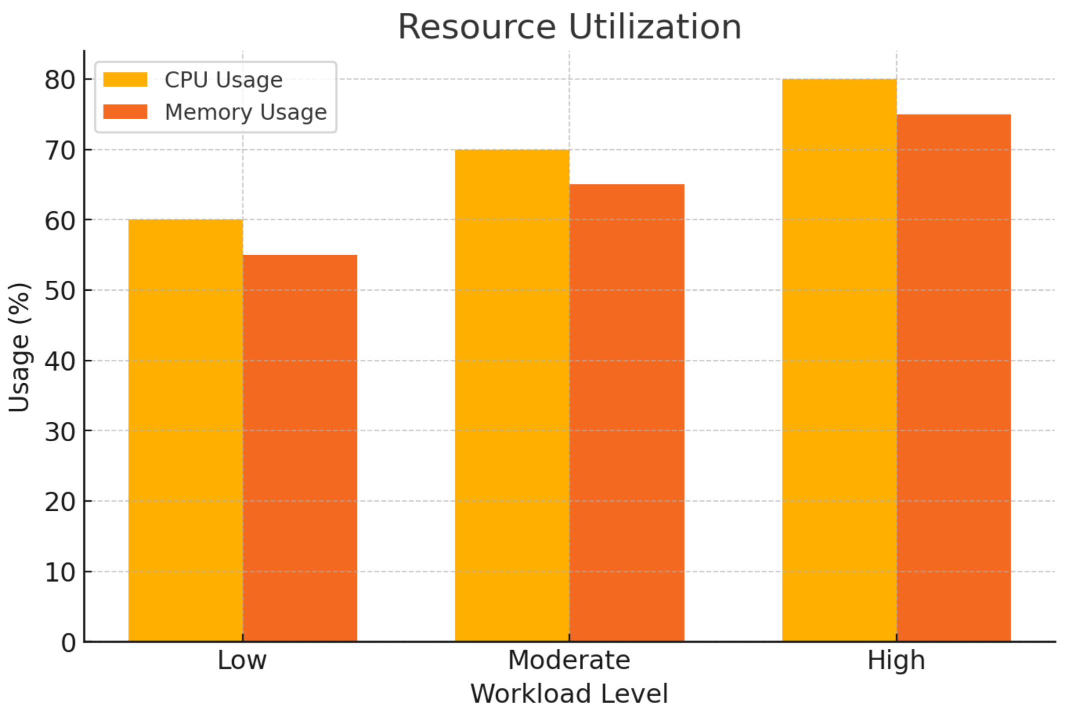

Fig 8: Resource Utilization: CPU and memory utilization are optimized in the AI-driven framework, reflecting more efficient resource usage and it was shown in figure 8.

CONCLUSION

This work shows how a data-driven AI approach can help in addressing the questions of performance and reusability in distributed cloud applications. Intelligent data partitioning, adaptive indexing, and predictive caching that inform the proposed approach allow real-time response to changes in the workload distribution across resources, improving resource allocation and system efficacy. Using the proposed framework to compare has provided the proof that it offer sizable gain in most of the areas of latency, throughput, scalability and access performance in compare to the baseline system. These results show that AI-driven, adaptive approaches to confrontation of the above-discussed problems are indispensable for creating fundamentally new, higher-performance, and more efficient next-generation cloud infrastructures.

REFERENCES

1. Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In 2009 IEEE Conference on Computer Vision and Pattern Recognition, pages 248–255. IEEE, 2009.
2. Nitin Gupta, Hima Patel, Shazia Afzal, Naveen Panwar, Ruhi Sharma Mittal, Shanmukha Guttula, Abhinav Jain, Lokesh Nagalapatti, Sameep Mehta, Sandeep Hans, et al. Data quality toolkit: Automatic assessment of data quality and remediation for machine learning datasets. arXiv preprint arXiv:2108.05935, 2021.
3. Justin M. Johnson and Taghi M. Khoshgoftaar. Survey on deep learning with class imbalance. Journal of Big Data, 6(1):1–54, 2019.
4. Minesh Mathew, Dimosthenis Karatzas, and C.V. Jawahar. Docvqa: A dataset for VQA on document images. In Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pages 2200–2209, 2021.
5. Tarun Narayanan, Ajay Krishnan, Anirudh Koul, and Siddha Ganju. Curator: Creating large-scale curated labeled datasets using self-supervised learning. arXiv preprint arXiv:2212.14099, 2022.
6. Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In International Conference on Machine Learning, pages 5389–5400. PMLR, 2019.
7. Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. International Journal of Computer Vision, 130(6):1526–1565, 2022.
8. Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R. Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615, 2022.
9. Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. Self-rewarding language models. arXiv preprint arXiv:2401.10020, 2024.
10. Ren, P., Xiao, Y., Chang, X., Huang, P.-Y., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. ACM computing surveys (CSUR) 54, 9 (2021), 1–40.
11. Rahul Kalva. Revolutionizing healthcare cybersecurity a generative AI-Driven MLOps framework for proactive threat detection and mitigation, World Journal of Advanced Research and Reviews, v. 13, n. 3, p. 577-582, 2022.
12. Ankush Reddy Sugureddy. Enhancing data governance frameworks with AI/ML: strategies for modern enterprises. International Journal of Data Analytics (IJDA), 2(1), 2022, pp. 12-22.
13. Ankush Reddy Sugureddy. Utilizing generative AI for real-time data governance and privacy solutions. International Journal of Artificial Intelligence & Machine Learning (IJAIML), 1(1), 2022, pp. 92-101.
14. Sudeesh Goriparthi. Leveraging AIML for advanced data governance enhancing data quality and compliance monitoring. International Journal of Data Analytics (IJDA), 2(1), 2022, pp. 1-11
15. Sudeesh Goriparthi. Implementing robust data governance frameworks: the role of AI/ML in ensuring data integrity and compliance. International Journal of Artificial Intelligence & Machine Learning (IJAIML), 1(1), 2022, pp. 83-91.