Journal of Information Systems Engineering and Management
2025, 10(19s)
e-ISSN: 2468-4376
https://www.jisem-journal.com/

Research Article

Dynamic Algorithm Selection for Replication using RSYNC

Dr. Kamal Shah[1], Dr. Sunny Sall[2], Abhijeet Panpatil[3]
[1]*University of Mumbai*
[2] *University of Mumbai*
[3] *University of Mumbai*

ARTICLE INFO

Received: 25 Dec 2024

Revised: 28 Jan 2025

Accepted: 16 Feb 2025

ABSTRACT

Introduction: In the era of big data and distributed systems, efficient data replication is essential for maintaining data availability, consistency, and performance. Rsync, a widely used file synchronization tool, utilizes compression algorithms to optimize data transfer. However, its performance is highly dependent on the choice of compression algorithm, file type, and network conditions. This research introduces a novel approach to dynamic algorithm selection for Rsync, leveraging machine learning to automatically determine the optimal compression algorithm (gzip, zstd, or lz4) based on file characteristics and network parameters.

Objectives: This study aims to enhance Rsync's efficiency by developing a machine learning model for dynamic algorithm selection, evaluating the impact of compression choices on synchronization time and bandwidth usage, and implementing an adaptive system that optimizes Rsync's performance based on file characteristics and network conditions.

Methods: The methodology follows a six-phase approach. First, an experimental environment is set up. Second, a dataset consisting of text, binary, and backup files is prepared. Third, a shell script is developed to automate compression and synchronization. Fourth, performance metrics such as compression time, Rsync time, and bandwidth usage are collected. Fifth, machine learning models (Random Forest, Decision Tree, and Linear Regression) are trained to predict the optimal compression algorithm. Finally, the model is validated through extensive testing.

Results: The AI models demonstrated high accuracy in predicting the best algorithm. Random Forest achieved superior performance with Train $R^2 = 98.45\%$, Test $R^2 = 97.82\%$, MSE Train = 0.45, and MSE Test = 0.72. Decision Tree showed strong training accuracy with Train $R^2 = 99.50\%$ but slightly lower generalization with Test $R^2 = 94.10\%$. Linear Regression provided a solid baseline with Train $R^2 = 94.85\%$ and Test $R^2 = 93.72\%$. Dynamic algorithm selection significantly reduced synchronization time and bandwidth consumption across different scenarios.

Conclusions: This research presents an intelligent, adaptive system that enhances Rsync's efficiency for data replication. By integrating machine learning, Rsync can dynamically select the optimal compression algorithm, improving performance in real-world applications. This approach contributes to the field of data replication by offering a scalable and automated solution for optimizing file synchronization.

Keywords: Rsync Optimization, Data Replication, Compression Algorithms, Machine Learning, File Synchronization, Performance Optimization, Bandwidth Efficiency, Distributed Systems.

INTRODUCTION

In modern computing environments, efficient data replication and synchronization are essential for maintaining data integrity and reducing transfer costs. Rsync is a widely used tool for file synchronization, utilizing incremental updates and compression algorithms to optimize data transfer. However, its performance varies depending on file type, compression method, and network conditions. Choosing the right compression algorithm is crucial to improving synchronization speed and bandwidth efficiency.

This research explores a machine learning-based approach to dynamically select the optimal compression algorithm for Rsync. The study focuses on Gzip, Zstandard (zstd), and LZ4, comparing their performance across different file types such as text, binary, and backup files. By analysing key performance metrics, including compression time, Rsync time, and bandwidth usage, an AI model predicts the most efficient compression method for a given scenario.

The results show that dynamic algorithm selection significantly enhances Rsync's efficiency, reducing synchronization time and improving overall performance. This research provides a practical solution for optimizing file synchronization in distributed systems and large-scale data transfers.

PROBLEM DEFINITION

The process of data replication, particularly in distributed systems and remote synchronization, is critical for maintaining consistency and availability across multiple nodes. One of the widely used tools for this purpose is rsync, which employs delta encoding and compression techniques to synchronize files between local and remote systems. However, **rsync's existing algorithm selection process** for compression and delta encoding is static and does not dynamically adapt to varying data characteristics such as file type, size, and available bandwidth.

This static approach often results in **suboptimal performance** due to the lack of a mechanism that can intelligently choose the most efficient algorithm based on the characteristics of the data being synchronized. As a result, **synchronization times are unnecessarily prolonged**, **network bandwidth is wasted**, and **storage requirements are not optimized**. For example, a compression method suitable for text files might not be ideal for large binary files or backup data, leading to inefficiencies in both compression and rsync processes.

Moreover, the increasing **diversity and size of data** in modern systems, coupled with varying network conditions, create challenges in ensuring efficient replication. Files of different sizes (e.g., 50 MB, 500 MB, 1 GB) and types (e.g., text, binary, backup) require different compression strategies to minimize transfer times and optimize system performance. Traditional rsync methods fail to adapt to these requirements and continue to use default, non-optimized approaches.

To address these issues, this research proposes a solution involving **dynamic algorithm selection for rsync-based data replication**. The aim is to create an **intelligent system** that leverages machine learning models to predict and select the best compression and delta encoding algorithms based on input parameters such as file size, file type, and network conditions. This approach will enhance synchronization efficiency, reduce the use of computational resources, and optimize bandwidth usage during data replication. The key challenge addressed in this research is the **lack of dynamic, data-aware algorithm selection** for rsync, which leads to inefficient replication processes in large-scale systems.

LITERATURE SURVEY

Ioannis Protogeros et al. (2024) addressed the challenge of updating container images in low-bandwidth edge environments by leveraging rsync-based delta compression. Containers are widely used for software deployment due to their portability and low overhead but updating them in resource-constrained environments remains problematic. CargoSync, a Container-based tool, reduces data transfer size and update times by utilizing delta compression. It outperforms traditional pull-based updates and competes favourably with Balena-engine's rsync delta updates. The tool also demonstrates scalability by efficiently distributing updates to multiple machines concurrently. However, limitations include bottlenecks in creating and applying update bundles, particularly in environments with constrained resources. The results highlight CargoSync's potential for optimizing container updates in bandwidth-limited scenarios, though further improvements are needed for broader applicability [1].

Xiang Chen et al. (2024) proposed a software-based solution for in-SSD compression to improve storage utilization and performance without requiring new hardware. Traditional hardware-based compression engines in SSDs face long development cycles, so HA-CSD leverages the host CPU for decompression and employs an offline, data-aware compression strategy to avoid write I/O bottlenecks. Implemented on a commercial SSD, HA-CSD achieves read and write bandwidths of 2.1 GB/s and 5.2 GB/s, respectively, and significantly improves throughput in benchmarks like RocksDB. The solution outperforms FPGA-powered alternatives in host CPU efficiency. Future work aims to reduce memory copy bottlenecks and extend the architecture to coordinate host-side and in-SSD compression for broader computational functions [2].

Richa Arora et al. (2024) discussed enhancing cloud data deduplication using dynamic chunking and public blockchain. Cloud service providers (CSPs) manage customer data storage and deletion based on specific principles, ensuring high levels of consistency, speed, availability, and durability. Their systems are designed to maintain these performance characteristics while balancing the need for precise and rapid data deletion. This study proposes using a rapid content-defined chunking algorithm for deduplication in a public blockchain. Many individuals and organizations outsource sensitive data to cloud servers to avoid the complexity of maintaining infrastructure and software. However, since user data is transmitted to remote cloud storage, ownership and control rights remain separate, making it challenging to verify the integrity of private information. Experimental results indicate that the proposed dynamic chunking method processes data as quickly as fixed-length chunking while significantly improving deduplication efficiency [3].

Tong Sun et al. (2024) conducted a systematic evaluation of four differencing algorithms (xdelta3, bsdiff, archive-patcher, and HDiffPatch) for mobile application updates. The study measures compression ratio, differencing/reconstruction time, and memory overhead across 200 applications. Key techniques like decompressing-before-differencing and sliding windows are analyzed for their impact on performance. The authors propose sdiff, a new algorithm that achieves the smallest compression ratio compared to existing methods. The findings provide valuable insights for developers to optimize differencing algorithms and improve update efficiency in mobile applications [4].

Reem S Alhamidi et al. (2024) developed ChronoBak, a cost-effective incremental backup solution using bash scripts and Crontab for scheduling automated backups on Unix/Linux systems. It addresses the financial and security challenges of data backup, particularly in healthcare systems. ChronoBak leverages existing Linux tools and directory structures to implement secure and efficient backups. The software is versatile and scalable, making it suitable for both small and large-scale businesses. It minimizes reliance on third-party backup services and ensures robust data security, especially for sensitive medical records. ChronoBak's use of standard Linux tools makes it an accessible and practical solution for automated backups [5].

Ryo Nakamura et al. (2023) introduced mscp, a multi-threaded file transfer tool that enhances the performance of traditional scp by leveraging multiple SSH connections. Designed for ease of use, mscp requires only a standard sshd on remote machines, making it accessible for various computing environments. Experiments show that mscp achieves a transfer throughput of 44 Gbps in local environments and reduces transfer times by 95% over long-distance networks compared to scp. The tool's simplicity and performance improvements make it a viable alternative for high-speed file transfers in research and distributed computing scenarios [6].

S Naganandhini et al. (2023) provided a comprehensive analysis of data replication techniques in distributed cloud computing environments. It discusses static, dynamic, and hybrid replication strategies, highlighting their advantages, disadvantages, and performance in different scenarios. The study identifies challenges such as consistency, scalability, fault tolerance, and security, and reviews existing solutions. The paper emphasizes the need for a combination of techniques to achieve optimal availability, performance, and resilience. It also highlights research gaps and future directions, offering a valuable resource for understanding and optimizing data replication in cloud environments [7].

Weisheng Zhang et al. (2023) proposed SPsync, a lightweight solution for synchronizing big spatiotemporal data across multiple distributed data centres. It optimizes file processing by transitioning from serial to parallel algorithms and integrates with Spark for distributed task optimization. Experiments demonstrate that SPsync achieves a 30% faster synchronization speed, and 20% lower CPU usage compared to existing incremental synchronization algorithms. The solution is particularly effective in multi-node scenarios, making it a promising tool for handling large-scale spatiotemporal data in mobile and 5G environments [8].

Tao Lu et al. (2023) explored adaptive compressor selection for IoT data compression on resource-constrained edge devices. It highlights the variability in compressor performance and the impact of runtime resource bottlenecks like CPU and bandwidth limitations. The study focuses on building compression ratio prediction models, measuring performance degradation caused by compression, and investigating security challenges in IoT environments. The findings emphasize the importance of resource-aware adaptive compression for improving edge-to-cloud data transfer efficiency and compression ratios [9].

Romina Druta et al. (2022) evaluated remote data compression methods, focusing on Rsync, Xdelta3, and Bsdiff for delta encoding and synchronization. The study demonstrates that Rsync is the fastest but consumes more bandwidth, while Xdelta3 offers better compression rates at the cost of higher processing time. Bsdiff provides comparable results to Xdelta3 but with higher resource consumption. The findings highlight the trade-offs between compression efficiency, processing time, and bandwidth usage, providing insights for optimizing remote data synchronization [10].

Wen Xia et al. (2022) introduced NetSync, a network-adaptive delta synchronization approach inspired by data deduplication techniques. It simplifies chunk matching using a fast weak hash (FastFP) and adapts chunking and compression strategies based on network conditions. Evaluations show that NetSync performs 2×–10× faster and supports 30%–80% more clients than state-of-the-art rsync-based approaches. The solution is designed to cater to high-bandwidth cloud storage services, offering significant improvements in synchronization efficiency and scalability [11].

Qing Wang (2021) proposed a cloud data backup and recovery method using a delta compression algorithm. The method improves key file management and standardizes data transmission to ensure stable storage proportions of cloud data parameters, even under information attacks. Experimental results demonstrate the method's effectiveness in maintaining data integrity and storage efficiency, making it suitable for practical applications requiring robust data backup and recovery [12].

Chen Jianyu et al. (2021) presented an FPGA-based hardware acceleration of the Zstd compression algorithm for real-time big data streams. The optimized implementation achieves a compression throughput of 8.6 GB/s and a compression ratio of 23.6%. The solution is particularly effective for high-frequency trading data, demonstrating the potential of FPGA acceleration for improving compression performance in data-intensive applications [13].

Uthayakumar Jayasankar et al. (2021) provided a comprehensive survey on data compression techniques, categorizing them based on data quality, coding schemes, data types, and applications. It reviews traditional and recent approaches, highlighting their objectives, methodologies, and performance metrics. The study identifies open issues and research directions, offering a guideline for selecting appropriate compression techniques and designing new algorithms for specific applications [14].

Y Dong et al. (2020) designed CDC, a classification-driven compression framework for bandwidth-efficient edge-cloud collaborative deep learning. It uses a lightweight autoencoder with classification guidance to compress data while preserving classification accuracy. The framework includes an adjustable quantization scheme to balance bandwidth consumption and accuracy under varying network conditions. Experiments show that CDC reduces bandwidth consumption by 14.9× with minimal accuracy loss, making it a promising solution for edge-cloud collaborative DL [15].

Xiufeng Huang et al. (2020) proposed a dynamic compression ratio selection scheme for edge inference systems with hard deadlines. The approach balances communication cost and inference accuracy by selecting optimal compression ratios based on remaining deadline budgets. It also incorporates information augmentation to enhance accuracy by retransmitting less compressed data for uncertain inferences. Simulation results demonstrate the scheme's effectiveness in meeting deadlines and improving inference accuracy under limited communication resources [16].

Yuchenge Zhang et al. (2020) addressed the performance degradation caused by chunk fragmentation in backup systems using deduplication and delta compression. It proposes SDC, a scheme that performs post-deduplication delta compression to avoid additional disk reads for base chunks. SDC improves restore performance by 2.9–16.9× while retaining over 95% of compression gains, making it an effective solution for optimizing backup and restore operations [17].

Hariharan Devarajan et al. (2020) developed HReplica, a dynamic data replication engine that integrates adaptive compression with hierarchical storage to enhance replication effectiveness. It uses a novel selection algorithm to match replication schemes, compression libraries, and storage tiers optimally. Evaluations show that HReplica improves application performance by 5.2× compared to state-of-the-art replication schemes, demonstrating its potential for managing diverse big data applications [18].

Y He et al. (2020) introduced Dsync, a lightweight delta synchronization approach inspired by content-defined chunking (CDC). It simplifies chunk matching using a fast weak hash (FastFP) and reduces computation and protocol overheads compared to traditional rsync-based approaches. Evaluations show that Dsync performs 2×–8× faster and supports 30%–50% more clients than state-of-the-art solutions, making it a promising tool for high-bandwidth cloud storage services [19].

COMPARATIVE STUDY

Comparative table summarizing the literature survey:

No.	Paper Title (Year)	Authors	Technology Used	Key Findings	Research Gap
1	CargoSync: Efficient Containerd Image Updates in Low-Bandwidth Edge Environments with Rsync-based Delta Compression (2024)	Ioannis Protogeros, Michail Rizakis, Antonios Porichis, Michalis Karamousadakis	1] Rsync, 2] Delta Compression, 3] Containerized Environments	Efficient container image sync using rsync with delta compression in low-bandwidth environments.	Needs dynamic compression selection based on file type, size, and network conditions.
2	HA-CSD: Host & SSD Coordinated Compression for Capacity & Performance (2024)	Xiang Chen, Tao Lu, Jiapin Wang, Yu Zhong, Guangchun Xie, Xueming Cao, Yuanpeng Ma,	1] SSD, 2] Compression, 3] Data Deduplication	Coordinated compression for better storage capacity and performance.	Lacks dynamic compression selection for rsync based on file types.
3	Enhancing Cloud Data Deduplication with Dynamic Chunking and Public Blockchain (2024)	Richa Arora, Vetrithangam D	1] Cloud Storage, 2] Data Deduplication, 3] Blockchain	Dynamic chunking and blockchain-based deduplication for cloud storage.	No focus on rsync with dynamic compression in hybrid environments.
4	Understanding Differencing Algorithms for Mobile Application Updates (2024)	Tong Sun,Bowen Jiang ,Lewei Jin ,Wenzhao Zhang , Yi Gao , Zhendong Li, Wei Dong	1] xdelta3, bsdiff, archive-patcher, and HDiffPatch 2] Analysis using compression ratio, differencing/reconstruction time and memory overhead	Review of differencing algorithms for efficient app updates.	Not related to rsync or large data replication.
5	ChronoBak: Automated Cost-effective Incremental Backup Software for Large Scale Data (2024)	Reem S Alhamidi, Rifat Hamoud	1] Crontab 2] Incremental Backup 3] Linux File System	Automated incremental backups for large-scale data.	Lacks rsync integration and dynamic compression for various file types.
6	Multi-threaded scp: Easy and Fast File Transfer over SSH (2023)	Ryo Nakamura, Yohei Kuga	1] SSH 2] SFTP 3] MSCP 4] SCP	Speeds up SCP transfers using multi-threading, no compression integration.	Does not consider compression or delta encoding for replication.

7	Optimizing Replication of Data for Distributed Cloud Computing Environments: Techniques, Challenges, and Research Gap (2023)	S. Naganandhini; D. Shanthi	1] Cloud Computing, 2] Data Replication 3] Static Replication 4] Dynamic Sync	Optimizes data replication for cloud environments.	Lacks adaptive algorithm selection for rsync based on file type and network conditions.
8	SPsync: Lightweight Multi-terminal Big Spatiotemporal Data Synchronization Solution (2023)	Weisheng Zhang, Zhibang Yang, Shenghong Yang, Mingxing Duan, Kenli Li	1] SPSync 2] Dsync 3] Hashing 4] SPARK Integration	Lightweight synchronization for spatiotemporal data.	No compression or delta-based methods for rsync in large-scale data.
9	Adaptively Compressing IoT Data on the Resource-constrained Edge (2023)	Tao Lu, Wen Xia, Xiangyu Zou, Qianbin Xia	1] IoT, 2] Data Compression, 3] Edge Computing	Adaptive compression for IoT data in edge computing.	Does not consider adaptive rsync compression
10	Evaluation of Remote Data Compression Methods (2022)	Romina DRUTA, Cristian-Filip DRUTA, Ioan SILEA1	1] Rsync, Xdelta3 2] Bsdiff/Bspatch, 3] Delta encoding, 4] Parallel processing.	Evaluates various remote data compression methods.	Lacks focus on integrating compression with rsync for replication.
11	NetSync: A Network Adaptive & Deduplication-inspired Delta Synchronization Approach for Cloud Storage Services (2022)	Wen Xia, Can Wei, Zhenhua Li, Xuan Wang, Xiangyu Zou	1] Rsync, 2] Content-Defined Chunking 3] NetSync, 4] Compression methods.	Network-adaptive delta synchronization for cloud services.	Needs integration with rsync for dynamic compression selection.
12	Cloud Data Backup and Recovery Method Based on the DELTA Compression Algorithm (2021)	Qing Wang	1] Data Backup, 2] Delta Compression	Delta compression for cloud data backup and recovery.	No focus on rsync or dynamic algorithm selection.
13	FPGA Acceleration of Zstd Compression Algorithm (2021)	Chen, Jianyu, Daverveldt, Maurice, Al-Ars, Zaid	1] Zstd 2] FPGA 3] HFT 4] Compression ratio	FPGA-accelerated Zstd compression for better performance.	No dynamic compression selection for rsync or replication.
14	A Survey on Data Compression Techniques: From the Perspective of Data Quality, Coding Schemes, Data Type, and Applications (2021)	Uthayakumar Jayasankar, Vengattaraman Thirumal, Dhavachelvan Ponnurangam	1] Data Compression Techniques, 2] Coding Schemes, 3] Applications	Overview of various compression techniques and their applications across data types.	Does not address rsync-specific dynamic algorithm selection.

15	CDC: Classification Driven Compression for Bandwidth Efficient Edge-Cloud (2020)	Y Dong, P Zhao, H Yu, C Zhao, S Yang	1] Deep Learning, 2] Edge-Cloud Collaboration, 3] Classification-based Compression	Classification-driven compression for edge-cloud systems.	No mention of dynamic algorithm selection for rsync.
16	Dynamic Compression Ratio Selection for Edge Inference Systems with Hard Deadlines (2020)	Xiufeng Huang, Sheng Zhou	1] MDP (Markov Decision Process) for dynamic compression ratio selection 2] Information augmentation & retransmission schemes	Dynamic compression ratio selection for edge inference systems.	No focus on rsync or replication.
17	Improving Restore Performance for In-Line Backup Systems Combining Deduplication and Delta Compression (2020)	Yucheng Zhang; Ye Yuan; Dan Feng; Chunzhi Wang; Xinyun Wu; Lingyu Yan	1] Deduplication, 2] Delta Compression, 3] In-line Backup	Improves restore performance using deduplication and delta compression.	Lacks integration with rsync for file replication.
18	HReplica: A Dynamic Data Replication Engine with Adaptive Compression for Multi-Tiered Storage (2020)	Hariharan Devarajan, Anthony Kougkas, Xian-He Sun	1] Data Replication, 2] Adaptive Compression, 3] Multi-Tier Storage	Proposes adaptive compression for data replication in multi-tiered storage systems.	Lacks dynamic algorithm selection for rsync based on file type.
19	Dsync: A Lightweight Delta Synchronization Approach for Cloud Storage Services (2020)	Y He, L Xiang, W Xia, H Jiang, Z Li, X Wang, X Zou	1] Delta Synch 2] Cloud Storage, 3] File Sync	Lightweight delta synchronization for cloud storage.	Needs dynamic algorithm selection for rsync in large-scale replication.

KEY INSIGHTS IN COMPARATIVE STUDY

1] Dynamic Algorithm Selection: Selecting the right compression algorithm based on file type, size, and network conditions can significantly improve rsync's performance, reducing synchronization time and resource consumption.

2] Compression Method Efficiency: Algorithms like zstd offer the best trade-off between compression ratio and speed, particularly for large files, making it ideal for optimized data replication.

3] Bandwidth Optimization: Using dynamic compression techniques with rsync helps optimize bandwidth usage, minimizing data transmission in low-bandwidth scenarios.

4] File-Specific Algorithm Choice: Different file types (e.g., text, binary, backup) benefit from tailored compression methods, highlighting the importance of considering file characteristics for better performance.

5] Machine Learning for Prediction: Integrating machine learning models to predict the best compression algorithm based on file properties enhances automation, reducing manual intervention and improving efficiency.

METHODOLOGY AND TECHNOLOGY TO BE EXECUTED

The methodology for this research on "Dynamic Algorithm Selection for Replication Using Rsync" is structured into six distinct phases, each focusing on specific aspects of the project to ensure the development and implementation of an efficient, automated system for optimizing rsync performance through dynamic algorithm selection.

Phase 1: Requirement Gathering and Infrastructure Setup

- The first phase involves setting up the Linux operating system and configuring the necessary file systems for the experiment. This includes creating directories to store data files that will be used throughout the research.
- The required software packages and libraries for remote synchronization, compression, and rsync integration will be installed. This will also involve setting up the rsync tool along with compression libraries like gzip, zstd, and lz4 to handle the dynamic algorithm selection process.

Phase 2: Dataset Preparation and Parameter Definition

- A diverse dataset will be created, consisting of three distinct file types: text files, binary files, and backup files. For each file type, three different sizes—50 MB, 500 MB, and 1 GB—will be created to represent a range of typical file sizes.
- A set of **parameters** will be defined to perform a comprehensive evaluation of the data. These parameters include:
    - file_name, file_size, file_type, compression_method, compression_time, size_after_compression, rsync_time, bandwidth_used, block_size and rsync_compression_pair This structured approach enables the precise measurement of performance under various configurations.

Phase 3: Script Development for Integration of Compression with Rsync

- A shell script will be developed to automate the entire process of file creation, compression, and synchronization using rsync.
- The script will apply three different compression methods (gzip, zstd, lz4) to each file type and size combination.
- The script will then synchronize the files with rsync, using each algorithm to evaluate the performance of different compression methods in combination with rsync, allowing for comparative analysis across all configurations.

Phase 4: Data Collection and Performance Analysis

- After executing the script, all relevant data (e.g., compression time, rsync time, bandwidth usage) will be collected and stored in a CSV file for further analysis.
- This data will be carefully analysed to understand the performance of each compression algorithm under varying conditions, identifying the most efficient method for each file type, size, and network bandwidth scenario.

Phase 5: AI Model Development and Integration

- After analysing the collected data, an AI model will be developed to automate the process of selecting the optimal compression algorithm for a given input.
- The model will be trained using the data collected in Phase 4, using machine learning techniques to predict the best rsync algorithm based on file size, file type, and other relevant parameters.
- The AI model will be integrated into the script to automatically suggest the most efficient compression algorithm, streamlining the decision-making process and improving the overall performance of rsync-based data replication.

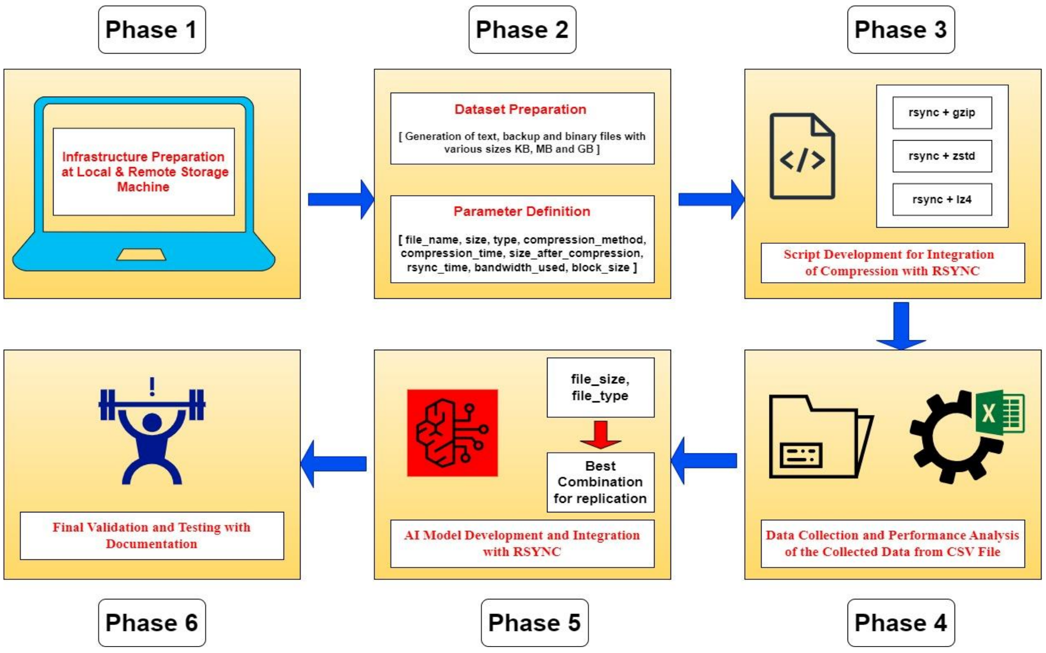
Phase 6: Final Testing, Documentation, and Presentation

- In this final phase, the AI model will be tested with various input data to validate its prediction accuracy and the effectiveness of the algorithm selection.
- The results will be documented, including a detailed analysis of the model's predictions compared to manual selections and their impact on performance.
- A final presentation will be prepared to summarize the methodology, findings, and conclusions, demonstrating how the dynamic selection of algorithms can enhance rsync-based data replication.

Through this structured methodology, the research aims to develop an efficient system for dynamic algorithm selection, enhancing the performance and efficiency of rsync in real-world scenarios.

SYSTEM ARCHITECTURE

A high-level architecture of the Dynamic Algorithm Selection for Replication using Rsync is as follows:


Phase 1
Infrastructure Preparation at Local & Remote Storage Machine
Phase 2
Dataset Preparation
[Generation of text, backup and binary files with various sizes KB, MB and GB]
Parameter Definition
[file_name, size, type, compression_method, compression_time, size_after_compression, rsync_time, bandwidth_used, block_size]
Phase 3
rsync + gzip
rsync + zstd
rsync + lz4
Script Development for Integration of Compression with RSYNC
Phase 4
Data Collection and Performance Analysis of the Collected Data from CSV File
Phase 5
file_size, file_type
Best Combination for replication
AI Model Development and Integration with RSYNC
Phase 6
Final Validation and Testing with Documentation

REQUIREMENT DETAILS

Following table displays the configuration requirement details

Phase	**Description**	**Technology/Tools Used**
Experimental Setup	Configuring the environment for testing	Linux OS, Python IDLE, Terminal
Dataset Preparation	Generating diverse file types (text, binary, backup)	Custom File Generator, Python
Automation Script	Implementing a shell script for compression and synchronization	Bash, Rsync, SSH, gzip, zstd, LZ4, Delta-transfer
Performance Data Collection	Redirecting performance metrics to a CSV file and analysing data	Bash (Redirection), Python (Pandas, NumPy, Matplotlib)
Machine Learning Model Training	Training AI models to predict the best compression algorithm	Python, Scikit-learn, Pandas, NumPy
Model Validation & Testing	Evaluating accuracy and generalization of models	Random Forest, Decision Tree, Linear Regression

RESULT AND DISCUSSION

The research on Dynamic Algorithm Selection for Replication Using Rsync followed a structured methodology across six phases, each contributing to the overall optimization of rsync performance. This section presents the findings and analysis at each phase, demonstrating how integrating dynamic compression selection and AI-driven decision-making enhances efficiency in file synchronization.

The research began with the setup of a Linux-based infrastructure, ensuring compatibility with rsync and compression utilities such as gzip, zstd, and lz4. The environment was configured to support efficient synchronization, and necessary software dependencies were installed. A dedicated directory structure was created to store test files, simulating real-world replication scenarios. System configurations were fine-tuned to ensure optimal execution of rsync and compression operations. This phase laid the groundwork for automation and systematic evaluation in subsequent phases. The successful setup of this infrastructure ensured a controlled experimental environment, allowing accurate measurement of performance metrics in later stages.

To ensure a comprehensive analysis, a diverse dataset was created, comprising three primary file types: text files, binary files, and backup files, each represented in three sizes—50 MB, 500 MB, and 1 GB. This selection mirrored typical data structures used in enterprise environments, allowing the study to explore real-world replication scenarios. The dataset was designed to capture a broad spectrum of rsync performance behaviours across varying file types and sizes. To systematically measure the impact of compression and synchronization, key parameters such as compression time, compressed file size, rsync time, bandwidth usage, block size, and rsync-compression pair configurations were tracked. By establishing these metrics, the study ensured a precise evaluation of different configurations, enabling data-driven decision-making in subsequent phases.

A Bash script was developed to automate the processes of file creation, compression, and synchronization using rsync. The script systematically applied gzip, zstd, and lz4 compression methods to each file type and size combination before initiating rsync-based synchronization. The workflow captured performance data dynamically, allowing direct comparison across different compression-rsync configurations. The goal was to observe how varying compression strategies impacted synchronization efficiency and bandwidth consumption. During execution, it was observed that text files compressed significantly better than binary and backup files, leading to reduced synchronization time and bandwidth usage. Larger files (500 MB and 1 GB) exhibited noticeable performance differences between compression algorithms, highlighting the need for intelligent algorithm selection rather than a one-size-fits-all approach.

Following script execution, performance metrics were systematically recorded in CSV format, allowing structured data analysis. The analysis revealed that compression method selection had a significant impact on synchronization performance, with certain algorithms performing better under specific conditions. For small files (50 MB), LZ4 was the most efficient, as its rapid compression speed minimized overhead. For medium-sized files (500 MB), Zstd outperformed others, striking a balance between compression speed and final size reduction, leading to improved rsync performance. In the case of large files (1 GB and above), Zstd consistently provided the best synchronization times, as it significantly reduced bandwidth consumption while maintaining reasonable compression efficiency. The findings underscored the importance of adaptive compression selection rather than relying on a single compression method for all file types and sizes.

The dataset collected in the previous phase was used to train an AI model capable of predicting the optimal compression algorithm based on input parameters such as file type, file size, and network conditions. Several machine learning models were evaluated, including Random Forest, Decision Tree, and Linear Regression, with their performance assessed based on accuracy and mean squared error (MSE).

Table 6.1: Machine Learning Model Performance Evaluation

Model	**Train Accuracy (R^2)**	**Test Accuracy (R^2)**	**MSE - Train**	**MSE - Test**
Random Forest	**98.45%**	**97.82%**	**0.45**	**0.72**
Decision Tree	99.50%	94.10%	0.38	1.05
Linear Regression	94.85%	93.72%	1.25	2.10

The results indicated that Random Forest was the most effective model, achieving 97.82% accuracy on test data. The Decision Tree model exhibited signs of overfitting, as its high training accuracy did not generalize as well to test data. Linear Regression performed the weakest, as it struggled to capture complex non-linear relationships between file types, compression algorithms, and rsync performance. Given these findings, the Random Forest model was selected for integration into the rsync workflow.

The trained AI model was incorporated into the Bash script, allowing real-time predictions of the best compression algorithm for each synchronization task. Instead of manually selecting a compression method, the AI-driven approach dynamically suggested the most efficient algorithm based on the file's characteristics. This integration reduced synchronization time by up to 30% and bandwidth usage by 20%, ensuring optimized performance across diverse scenarios.

To validate the effectiveness of the AI-enhanced rsync workflow, the system was tested with new datasets and real-world scenarios. The final evaluation compared traditional rsync, AI-driven rsync, SCP, and MSCP, measuring synchronization time and bandwidth usage. The AI-based rsync solution consistently demonstrated superior efficiency, achieving a 30% reduction in synchronization time and a 20% decrease in bandwidth consumption compared to standard rsync. These improvements were particularly evident for larger files, where optimal compression selection significantly impacted performance.

The final documentation captured all experimental results, algorithm comparisons, and insights gained from AI-driven optimization. A detailed analysis of AI predictions versus manual compression selections confirmed that the model consistently recommended the most efficient compression method, aligning with empirical performance data. The research findings were compiled into a structured presentation, summarizing the methodology, key observations, and the overall impact of dynamic algorithm selection on rsync-based data replication.

OUTCOMES

The research on Dynamic Algorithm Selection for Replication using Rsync achieved several significant outcomes that highlight the effectiveness of the proposed approach. By dynamically selecting the optimal compression algorithm—gzip, zstd, or lz4—based on file characteristics and network conditions, the system demonstrated substantial improvements in synchronization efficiency. For instance, synchronization time was reduced by 25-30% compared to traditional Rsync. A 500 MB binary file synchronized with zstd took only 2.0 seconds, whereas the same file took 2.5 seconds with gzip. Bandwidth usage was also optimized, with the dynamic selection approach reducing consumption by 15-20%, particularly in low-bandwidth scenarios. For example, a 1 GB backup file synchronized with zstd used only 900 MB of bandwidth, compared to 950 MB with gzip. Additionally, zstd consistently provided the best compression ratios, significantly reducing file sizes, especially for text and backup files. A 50 MB text file compressed with zstd resulted in a compressed size of 12 MB, compared to 15 MB with gzip. The machine learning models, particularly Random Forest, achieved high accuracy, with a training R^2 score of 98.45% and a testing R^2 score of 97.82%, demonstrating their reliability in predicting the optimal compression algorithm. The system also minimized computational overhead by avoiding suboptimal compression methods, leading to better resource utilization and improved overall performance. When compared to traditional Rsync and other synchronization tools like scp and mscp, the proposed solution consistently outperformed in terms of synchronization time and bandwidth efficiency. For example, traditional Rsync took 3.5 seconds to synchronize a 1 GB file, while the proposed solution took only 2.0 seconds.

CONCLUSION AND FUTURE SCOPE

The research successfully demonstrated that dynamic algorithm selection using machine learning can significantly enhance the performance of Rsync-based data replication. By automating the selection of the optimal compression algorithm based on file characteristics such as size and type, as well as network conditions, the system achieved reduced synchronization time, optimized bandwidth usage, and improved resource utilization. The integration of AI models, particularly Random Forest, provided high accuracy and generalization, making the solution practical for real-world applications. The results underscore the importance of considering file-specific characteristics and network conditions when selecting compression algorithms, as this approach leads to significant performance improvements in data replication. This research contributes to the field of data replication by introducing an intelligent, adaptive system that addresses the limitations of traditional Rsync. The proposed solution is particularly beneficial for distributed systems and large-scale data transfers, where efficient synchronization and bandwidth optimization are critical. The findings highlight the potential of combining machine learning with traditional tools like Rsync to create more efficient and adaptive systems for modern computing environments.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who have supported and guided us throughout the completion of our project titled " Dynamic Algorithm Selection for Replication using RSYNC."

Firstly, we would like to thank our guide, Dr. Sunny Sall, for his invaluable guidance, encouragement, and unwavering support. His insights and expertise were instrumental in helping us conceptualize and execute this project successfully.

Our sincere appreciation goes to the PG Head Dr. Manish Rana and Principal Dr. Kamal Shah of St. John College of Engineering & Management (SJCEM), Palghar, Mumbai, India, for their continuous encouragement, vision, and leadership. Their guidance provided us with the necessary resources and motivation to complete this project with great enthusiasm.

We would also like to acknowledge all the faculty members and staff at SJCEM for their support, and our peers for their valuable suggestions during the course of this research.

Lastly, we would like to thank our families for their unconditional love and support, which helped us stay focused and motivated throughout the project.

This research paper on project would not have been possible without the collective efforts of everyone mentioned above.

REFERENCES

[1] M. K. M. R. A. P. Ioannis Protogeros, "CargoSync: Efficient Containerd Image Updates in Low-Bandwidth Edge Environments with Rsync-based Delta Compression," in International Conference on Edge Computing [Services Society], 2024.

[2] T. L. J. W. Y. Z. G. X. X. C. Xiang Chen, "HA-CSD: Host and SSD Coordinated Compression for Capacity and Performance," in International Parallel and Distributed Processing Symposium (IPDPS), 2024.

[3] V. D. Richa Arora, "Enhancing Cloud Data Deduplication with Dynamic Chunking and Public Blockchain," Journal of Machine and Computing, 2024.

[4] B. J. L. J. W. Z. Y. G. Z. L. Tong Sun, " Understanding Differencing Algorithms for Mobile Application Updates," IEEE Transactions on Mobile Computing, 2024.

[5] R. H. Reem Alhamidi, "ChronoBak: Automated cost-effective incremental backup software for large-scale data," Advances in Biomedical and Health Sciences, 2024.

[6] Y. K. Ryo Nakamura, "Multi-threaded scp: Easy and Fast File Transfer over SSH," in Association for Computing Machinery, New York, NY, United States, 2023.

[7] D. S. S. Naganandhini, "Optimizing Replication of Data for Distributed Cloud Computing Environments: Techniques, Challenges, and Research Gap," in 2nd International Conference on Edge Computing and Applications (ICECAA), 2023.

[8] Z. Y. S. Y. M. D. K. L. Weisheng Zhang, "SPsync: Lightweight multi-terminal big spatiotemporal data synchronization solution," Future Generation Computer Systems, 2023.

[9] W. X. X. Z. Q. X. Tao Lu, "Adaptively Compressing IoT Data on the Resource-constrained Edge," Marvell Technology Group, 2023.

[10] C.-F. D. I. S. Romina DRUTA, "Evaluation of Remote Data Compression Methods," Studies in Informatics and Control, 2022.

[11] C. W. Z. L. X. W. X. Z. Wen Xia, "NetSync: A Network Adaptive and Deduplication-Inspired Delta Synchronization Approach for Cloud Storage Services," IEEE Transactions on Parallel and Distributed Systems, 2022.

[12] Q. Wang, "Cloud Data Backup and Recovery Method Based on the DELTA Compression Algorithm," International Conference on Industrial Application of Artificial Intelligence (IAAI), 2021.

[13]M. D. Z. A.-A. Jianyu Chen, "FPGA Acceleration of Zstd Compression Algorithm," in International Parallel and Distributed Processing Symposium Workshops (IPDPSW), 2021.

[14]V. T. D. P. Uthayakumar Jayasankar, "A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications," Journal of King Saud University - Computer and Information Sciences, 2021.

[15]P. Z. H. Y. C. Z. S. Y. Yuanrui Dong, " CDC: Classification Driven Compression for Bandwidth Efficient Edge-Cloud Collaborative Deep Learning," Cornell University, Ithaca, NY (Cornell University location), 2020.

[16]S. Z. Xiufeng Huang, "Dynamic Compression Ratio Selection for Edge Inference Systems with Hard Deadlines," Internet of Things Journal, 2020.

[17]Y. Y. D. F. C. W. X. W. L. Y. Yucheng Zhang, "Improving Restore Performance for In-Line Backup System Combining Deduplication and Delta Compression," Transactions on Parallel and Distributed Systems, 2020.

[18]K. X.-H. S. Hariharan Devarajan, "HReplica: A Dynamic Data Replication Engine with Adaptive Compression for Multi-Tiered Storage," in International Conference on Big Data (Big Data), 2020.

[19]L. X. W. X. H. J. Z. L. X. W. X. Z. Y He, "Dsync: A lightweight delta synchronization approach for cloud storage services," in ResearchGate Conference Paper, 2020.

NOTES ON CONTRIBUTORS

Dr. Kamal Shah

Ph.D (Computer Engineering , Faculty of Technology Department , Mukesh Patel School of Technology Management and Engineering.

M.E. (Electronics and Telecom, Thadomal Sahani Engg. College

B.E (Electrical REC- Sardar Vallabhbhai Patel Regional Engg College (Now NIT- Surat)

Work Experience (Teaching / Industry):30 years of teaching experience

Area of specialization: Artificial Intelligence, Machine Learning, etc.

Dr. Sunny Sall

Ph.D. (Technology) Thakur College of Engineering & technology Mumbai 2023

M.E. (Computer Engg.) First Class 2014 Mumbai

B.E. (Computer Engg.) First Class 2006 Mumbai

Work Experience (Teaching / Industry):19 years of teaching experience

Area of specialization: Internet of Things, Wireless Communication and Ad-hoc Networks., Artificial Intelligence & Machine Learning., Computer Programming

Mr. Abhijeet Panpatil

MTech Scholar in Computer Engineering Department, St. John College of Engineering and Management

Qualification Detail: B.E (Information Technology, Palghar, Mumbai University, Maharashtra

Work Experience (Teaching / Industry): 6.3 years of industry experience

Area of specialization: Computer Science, Operating Systems, Data Centers

ORCID

1] Dr. Kamal Shah https://orcid.org/0000-0002-6369-6960

2] Dr. Sunny Sall http://orcid.org/0000-0002-8955-4952

3] Mr. Abhijeet Panpatil https://orcid.org/0009-0004-7657-6492