# Democratizing Large Language Model Training in Open Science Environments

Chris Esposo
Georgia Institute of Technology
Atlanta, GA, USA
cesposo3@gatech.edu

## ABSTRACT

The exponential growth of Large Language Models (LLMs) has revolutionized artificial intelligence, enabling transformative applications across industries. However, training LLMs remains largely inaccessible due to the massive computational resources required. This research proposes a novel approach to democratizing LLM training by leveraging distributed computing across heterogeneous wide-area networks, specifically utilizing Open Science (OS) pool environments. Key challenges include efficient resource allocation, security and integrity of data during distributed training, and robust fault tolerance mechanisms. This proposal outlines methodologies to address these challenges, including adapting the Gale-Shapley algorithm for dynamic resource allocation, implementing secure multi-party computation and homomorphic encryption for security, and developing fault tolerance strategies suitable for wide-area networks.

## CCS CONCEPTS

• **Computing methodologies → Distributed algorithms**; **Neural networks**; *Machine learning*; • **Security and privacy** → *Distributed systems security*; Cryptography; • **Computer systems organization** → *Dependable and fault-tolerant systems and networks.*

## KEYWORDS

Large Language Models, Distributed Computing, Resource Allocation, Security, Fault Tolerance, Open Science Environments

## 1 INTRODUCTION AND MOTIVATION

The exponential growth of Large Language Models (LLMs) has revolutionized artificial intelligence, enabling transformative applications across industries. Models like GPT-4 and PaLM have demonstrated unprecedented capabilities in natural language understanding and generation. However, the training of LLMs remains largely inaccessible due to the massive computational resources required. While organizations like OpenAI and Google harness vast infrastructures to develop state-of-the-art models, smaller research institutions and organizations are effectively excluded, stifling innovation and limiting the diversity of contributions to this rapidly evolving field.

This research proposes a novel approach to democratizing LLM training by leveraging distributed computing across heterogeneous wide-area networks, specifically utilizing Open Science (OS) pool environments. These distributed environments, composed of volunteer computing resources and idle computational capacity from various institutions, offer a cost-effective alternative to traditional high-performance computing clusters. By harnessing these resources, we aim to enable the training of large-scale language models without the need for proprietary infrastructure.

The key challenges in this endeavor include efficient resource allocation in a dynamic and heterogeneous environment, ensuring the security and integrity of the model and data during distributed training, and implementing robust fault tolerance mechanisms to handle the volatility inherent in wide-area networks. Addressing these challenges requires innovative adaptations of existing algorithms and the development of new methodologies tailored to the unique characteristics of OS pool environments.

### Intellectual Merit

This research contributes to the field in three significant ways. First, it extends the theoretical foundations of the Gale-Shapley algorithm by adapting it for dynamic resource allocation in distributed LLM training environments. This adaptation involves developing new matching mechanisms capable of handling the variability and heterogeneity of resources, optimizing for both computational efficiency and network latency. Second, it introduces novel metrics for resource-task compatibility that account for heterogeneous computational capabilities and network conditions, facilitating optimal matching decisions in real-time. Third, this work proposes innovative security mechanisms, including secure multi-party computation and homomorphic encryption, to enable safe model training across untrusted nodes, mitigating the risks of data leakage and model theft.

### Broader Impacts

The democratization of LLM training has the potential to fundamentally transform the landscape of AI research and development. By lowering barriers to entry, this research could empower smaller organizations and underrepresented communities to contribute diverse and specialized language models, addressing needs in various domains such as low-resource languages and niche scientific fields. Beyond LLM training, the advancements in dynamic matching algorithms and distributed systems security could benefit a range of distributed computing applications, including collaborative scientific research, decentralized computing platforms, and crowd-sourced data processing.

## 2 BACKGROUND AND LITERATURE REVIEW

Training Large Language Models (LLMs) demands immense computational resources, which poses a significant barrier to entry for researchers and institutions without access to high-performance computing infrastructure. Recent efforts have aimed to democratize LLM training by exploring distributed computing strategies, optimizing resource utilization, enhancing security, and ensuring fault tolerance in heterogeneous environments. This section reviews relevant literature in these areas, highlighting the gaps that our research aims to address.

## 2.1 Distributed Training in Heterogeneous Environments

Asynchronous stochastic gradient descent (SGD) methods have been studied to leverage heterogeneous compute resources effectively. Liu et al. [4] investigated asynchronous Local-SGD for training large language models, focusing on the impact of hardware heterogeneity, model size, and optimizer choice. They identified that naive asynchronous implementations suffer from slow convergence due to stale gradients and proposed a method using delayed Nesterov momentum updates and dynamic adjustment of local training steps based on worker computation speed. This approach showed promise in matching synchronous Local-SGD performance while improving wall-clock time.

Similarly, Ryabinin et al. [10] introduced Moshpit SGD, a communication efficient decentralized training algorithm designed for heterogeneous and unreliable devices. By operating asynchronously and allowing devices to update model parameters independently without synchronization, they reduced communication overhead and maintained scalability and robustness. Their push-sum consensus mechanism ensured convergence even when devices dropped out or produced delayed updates.

These works highlight the potential of asynchronous and decentralized training methods in heterogeneous environments. However, they primarily focus on optimizing SGD algorithms without addressing resource allocation strategies that can handle dynamic and heterogeneous resource availability.

## 2.2 Resource Allocation and Parallelism Strategies

Efficient resource allocation is crucial for scaling LLM training across distributed systems. Ren et al. [8] presented ZeRO-Offload, a system that enables training models with over 13 billion parameters on a single GPU by offloading data and computation to CPU memory. This method minimizes data movement and CPU compute time, demonstrating significant performance improvements over baseline approaches.

Narayanan et al. [5] introduced PipeDream-2BW, a system for memory-efficient pipeline-parallel training of deep neural networks. Their approach combines novel pipelining and weight gradient coalescing strategies, achieving high throughput and low memory footprint. PipeDream-2BW effectively utilizes heterogeneous GPU clusters and can be combined with other parallelism techniques.

Oyama et al. [7] proposed $\mu$-cuDNN, a wrapper library that automatically divides mini-batch computations into smaller micro-batches to optimize performance within memory constraints. This technique enhances distributed training on heterogeneous hardware by improving utilization of fast convolution algorithms.

These strategies focus on optimizing resource utilization within homogeneous or controlled environments. Our work extends this by adapting the Gale-Shapley algorithm for dynamic resource allocation in heterogeneous OS pool environments, addressing the variability and unpredictability of resource availability.

## 2.3 Security Mechanisms in Distributed Training

Securing distributed training processes is essential, especially when involving untrusted nodes. Li et al. [3] introduced TransLinkGuard, a protection mechanism for transformer models deployed on edge devices. By integrating a lightweight authorization module with a Trusted Execution Environment (TEE), they enforced request-level model access and protected against runtime reverse engineering, significantly reducing the risk of model theft with negligible runtime overhead.

Tramèr et al. [11] presented SLALOM, a framework for executing neural networks securely within TEEs while outsourcing linear computations to an untrusted co-processor. By using verification techniques like Freivalds' algorithm, SLALOM achieved significant speedups for neural network inference without compromising security.

Duan et al. [2] introduced SSNet, a lightweight multi-party computation (MPC) scheme that leverages Shamir's Secret Sharing for practical privacy-preserving machine learning services in the cloud. Their approach reduced communication overhead and demonstrated scalability and robustness in distributed settings.

These works provide valuable insights into securing distributed machine learning systems. However, they often incur significant computational overhead or are designed for inference rather than training. Our research aims to integrate efficient security mechanisms like secure multi-party computation and homomorphic encryption into the training process, minimizing performance degradation.

## 2.4 Fault Tolerance in Wide-Area Networks

Ensuring fault tolerance in distributed training over wide-area networks is challenging due to node failures and unreliable communication. Niu et al. [6] introduced Hogwild!, a lock-free approach to parallelizing SGD by allowing asynchronous updates in shared memory without locking mechanisms. While effective for multi-core environments, its applicability to wide-area networks with heterogeneous nodes is limited.

Ren et al. [1] explored distributed online learning with adversarial participants in adversarial environments. They demonstrated that Byzantine-robust gradient descent algorithms with momentum can achieve sublinear regret under certain conditions, highlighting the difficulty of maintaining robust learning when participants are untrustworthy.

These studies address aspects of fault tolerance and robustness but do not provide comprehensive solutions for the dynamic and heterogeneous nature of OS pool environments. Our work aims to develop fault tolerance mechanisms that quickly detect and recover from node failures, reallocate tasks, and maintain consistent model updates without significantly impacting training performance.

## 2.5 Secure Multi-Party Computation in Machine Learning

Several frameworks have been proposed to enhance privacy-preserving machine learning through secure multi-party computation. Zhou et al. [12] provided a comprehensive survey of SMPC applied to machine learning, evaluating existing protocols and highlighting the need for better incentive structures and operational efficiency.

Ren et al. [9] introduced PrivDNN, which enhances privacy-preserving deep learning by partially encrypting critical neurons in a DNN, reducing computational overhead while maintaining model accuracy.

These advancements inform our approach to integrating security into distributed training. By selectively applying encryption and optimizing SMPC protocols, we aim to protect data and model integrity with minimal performance trade-offs.

### 2.6 Gaps and Motivation

While existing literature offers valuable strategies for distributed training, resource optimization, security, and fault tolerance, there is a lack of comprehensive solutions that address all these challenges in the context of heterogeneous and dynamic OS pool environments. Our research fills this gap by:

- Adapting the Gale-Shapley algorithm for dynamic resource allocation, considering real-time resource availability and task requirements.
- Integrating efficient security mechanisms into the training process, protecting against data leakage and model theft.
- Developing fault tolerance strategies tailored to wide-area networks, ensuring robust and efficient LLM training despite resource volatility.

By building upon and extending the methodologies reviewed, our work contributes to democratizing LLM training, making it accessible to a broader range of researchers and institutions.

## 3 RESEARCH QUESTIONS

The following research questions guide this work:

### RQ1: How can the Gale-Shapley algorithm be effectively adapted for dynamic resource allocation in heterogeneous distributed LLM training environments while maintaining matching stability and efficiency?

**Critical Reflection:**

The novelty of this research lies in extending the well-studied Gale-Shapley algorithm, traditionally applied to static allocation problems, to dynamic and heterogeneous distributed computing environments. This extension seeks to accommodate the variability and unpredictability inherent in OS pool resources, an area that is relatively unexplored. Specifically, the focus is on adapting the algorithm to match computational tasks, such as training sub-tasks, with available resources in real-time, accounting for heterogeneous hardware capabilities and network conditions. Success will be evaluated through measurable metrics, including matching stability, resource utilization efficiency, and the overall impact on training time and model performance. The feasibility of the proposed adaptation is supported by the potential for testing through simulations and pilot implementations in OS pool environments, leveraging existing distributed computing platforms. Ethically, this research adheres to standards that promote equitable access to computational resources without compromising individual privacy or security.

**Hypothesized Results:**

We hypothesize that the adapted Gale-Shapley algorithm will result in improved matching stability and resource utilization efficiency compared to baseline allocation methods. This improvement is expected to reduce the overall training time and enhance model performance.

### RQ2: What security mechanisms can enable safe and efficient model training across untrusted nodes while minimizing the impact on training performance?

**Critical Reflection:**

Combining principles of distributed systems security with LLM training presents a relatively uncharted area, especially within heterogeneous and untrusted environments. This research specifically aims to develop cryptographic protocols and system-level defenses to protect against data leakage, model inversion attacks, and unauthorized access, ensuring the confidentiality and integrity of the training process. Security effectiveness will be measured through resistance to various attack vectors, while the impact on training efficiency will be assessed using performance benchmarks and overhead analysis. Feasibility is ensured by leveraging and adapting existing cryptographic techniques such as secure multi-party computation and homomorphic encryption, making practical implementation within distributed environments achievable. Enhancing security and privacy aligns with ethical principles, promoting trust and safeguarding sensitive data.

**Hypothesized Results:**

We hypothesize that implementing security mechanisms such as secure multi-party computation (SMPC) and homomorphic encryption will prevent unauthorized access and data leakage without causing significant degradation in training performance.

### RQ3: How can fault tolerance mechanisms be implemented in wide-area network environments to ensure robust and efficient LLM training despite the volatility of resources?

**Critical Reflection:**

Addressing fault tolerance in the context of distributed LLM training over heterogeneous wide-area networks introduces unique challenges that have not been fully explored. The research aims to develop mechanisms that can quickly detect and recover from node failures, reallocate tasks, and maintain consistent model updates without significantly impacting training performance. Effectiveness will be evaluated through metrics such as recovery time, system availability, training throughput, and the impact on model convergence. Adapting existing fault tolerance strategies, such as checkpointing and redundant computations, within the context of OS pool environments is practical and can be validated through real-world testing. Ensuring system reliability and robustness supports the ethical goal of providing equitable access to reliable AI training resources.

**Hypothesized Results:**

We hypothesize that implementing fault tolerance mechanisms will significantly reduce recovery time after node failures and maintain training efficiency, resulting in minimal impact on model convergence.

## 4 METHODOLOGY

To address the research questions outlined, this study will involve a combination of theoretical algorithm development, system implementation, and experimental evaluation. The methodology comprises several key components.

### 4.1 Adapting the Gale-Shapley Algorithm for Dynamic Resource Allocation

*4.1.1 Algorithm Development.* The first step involves extending the Gale-Shapley stable matching algorithm to accommodate the dynamic and heterogeneous nature of OS pool environments. This will involve developing metrics that quantify the suitability of resources for specific training tasks, considering factors such as computational capabilities (CPU, GPU, memory), network bandwidth, latency, and current load. The traditional preference list structures will be modified to account for real-time resource availability and task requirements, with both resources (nodes) and tasks generating preference lists based on these compatibility metrics. Additionally, the matching process will be adapted to handle dynamic changes in resource availability, including nodes joining or leaving the network and fluctuating resource capacities.

Once resources and tasks are matched, the system must efficiently coordinate the distributed training process across these heterogeneous nodes. This necessitates a formal approach to managing local computations and global synchronization.

*4.1.2 Training Process Formalization.* To formalize the distributed training process within the matched resource-task pairs, the research will implement an asynchronous training approach adapted for heterogeneous environments. Each node in the distributed system will perform local updates while periodically synchronizing with the global model.

For each node $i$ in the distributed system, the local training process follows:

$$\theta_i^{(t+1)} = \theta_i^{(t)} - \eta \nabla L_i(\theta_i^{(t)}; D_i) \tag{1}$$

Where:

- $\theta_i^{(t)}$ represents the model parameters at step $t$ for node $i$
- $\eta$ is the adaptive learning rate, adjusted based on node capabilities
- $\nabla L_i$ is the gradient computed on the local data subset $D_i$

After $E_i$ local epochs, determined dynamically based on node characteristics, synchronization occurs according to:

$$\theta^{(t+1)} = \text{Aggregate}(\{\theta_i^{(t+1)}\}_{i=1}^{N}, \theta^{(t)}) \tag{2}$$

This formalization enables the system to:

- Accommodate heterogeneous computational capabilities through adaptive local update frequencies
- Minimize communication overhead by reducing synchronization frequency
- Maintain training stability through careful aggregation of local updates

The parameters $E_i$ and $\eta$ will be tuned dynamically based on the resource metrics collected through the Gale-Shapley matching process, ensuring efficient utilization of the matched resources while maintaining model convergence.

*4.1.3 Resource Allocation Optimization.* To complement the matching algorithm, we formalize the resource allocation process as an optimization problem. This formalization provides a quantitative framework for evaluating allocation decisions and guides the development of preference metrics used in the matching process.

The resource allocation process will be formalized as an optimization problem to ensure efficient distribution of computational tasks. Let:

- $N$ represent the set of available nodes
- $J$ represent the set of computational jobs
- $d_{ij}$ represent the network delay between node $i$ and job $j$
- $p_{ij}$ represent the processing time of job $j$ on node $i$
- $x_{ij}$ be a binary variable indicating assignment

The objective function aims to minimize the overall completion time:

$$\min \sum_{i \in N} \sum_{j \in J} (p_{ij} + d_{ij}) x_{ij} \tag{3}$$

Subject to the following constraints:

$$\sum_{i \in N} x_{ij} = 1, \quad \forall j \in J \tag{4}$$

$$\sum_{j \in J} p_{ij} x_{ij} \leq C_i, \quad \forall i \in N \tag{5}$$

Where $C_i$ represents the capacity of node $i$. This formalization provides a concrete framework for evaluating and optimizing resource a

These mathematical formulations provide the theoretical foundation for our implementation. To validate their practical effectiveness, we proceed with simulation and testing in realistic environments.

*4.1.4 Integration of Mathematical Frameworks.* The training process formalization and resource allocation optimization frameworks operate in complementary fashion to ensure efficient distributed training. The Gale-Shapley matching results inform the parameters of the training process through:

$$\eta_i = f(C_i, d_{ij}, p_{ij}) \tag{6}$$

Where $f$ represents a mapping function that determines the learning rate $\eta_i$ based on node capacity $C_i$, network delay $d_{ij}$, and processing time $p_{ij}$. Similarly, the local update frequency $E_i$ is determined by:

$$E_i = \left\lceil \frac{\min(C_i, \max_j p_{ij})}{\text{avg}_j p_{ij}} \right\rceil \tag{7}$$

This integration ensures that nodes with higher capacity and better network conditions perform more local updates before synchronization.

*4.1.5 Simulation, Prototyping, and Implementation.* After developing the algorithm, it will be implemented and tested in a simulated OS pool environment. A network simulation tool such as NS-3 will be used to model the behavior of heterogeneous nodes and network conditions. The adapted Gale-Shapley algorithm will be integrated within a distributed training framework, possibly using existing platforms like Horovod or a custom-built solution, to manage task-resource allocation. Various testing scenarios will be created to evaluate the algorithm's performance, including varying node availability, resource heterogeneity levels, and network conditions. Upon success of the simulation, we will proceed to testing it on our local cluster to see how the system works in real run-time. We will report on these and the results. then implement it on OS Pool.

*4.1.6 Evaluation Metrics.* The effectiveness of the adapted algorithm will be evaluated using metrics such as matching stability, measuring the stability of the matching over time and the satisfaction of both tasks and resources. Resource utilization efficiency will be assessed to determine how effectively computational resources are utilized, aiming for high utilization rates without overloading nodes. The impact on overall training time and model convergence will be analyzed compared to baseline allocation methods to understand the impact of training performance.

*4.1.7 Quantitative Success Criteria.* We do not yet know what the true values of these metrics will be, but set these as initial seeds to be changed as we run the system and collect statistics:

- **Resource Utilization**:
  - Minimum 80% average CPU/GPU utilization
  - Maximum 15% node idle time
  - Less than 10% resource allocation conflicts
- **Security Performance**:
  - Maximum 20% computational overhead from security measures
  - Zero successful unauthorized access incidents
  - Less than 5% false positive rate in threat detection
- **Fault Tolerance**:
  - Recovery time under 30 seconds for single node failures
  - 99.9% system availability
  - Maximum 5% performance degradation during recovery
- **Training Performance**:
  - Minimum 25% improvement in training time over baseline
  - Maximum 10% deviation in model accuracy
  - Convergence within 1.2x epochs compared to baseline

*4.1.8 Security Protocol Design.* To secure the distributed training process across untrusted nodes, the research will begin with threat modeling to identify potential security threats, including data leakage, model inversion attacks, and unauthorized access to model weights. Appropriate cryptographic methods such as secure multi-party computation (SMPC), homomorphic encryption, and differential privacy techniques will be selected. Protocols that integrate these cryptographic methods into the distributed training workflow will be developed, ensuring minimal overhead.

*4.1.9 Security Guarantees and Overhead Bounds.* The security mechanisms must satisfy the following formal guarantees:

$$P(\text{breach} \mid \text{SMPC}) \leq \epsilon \tag{8}$$

Where $\epsilon$ represents the maximum acceptable probability of security breach. The computational overhead is bounded by:

$$O_{\text{security}}(n) \leq \alpha \cdot t_{\text{baseline}} \tag{9}$$

Where $\alpha$ is the maximum acceptable overhead factor (typically 1.2-1.5) and $t_{\text{baseline}}$ is the baseline training time.

*4.1.10 Implementation and Integration.* The security protocols will be implemented within the distributed training framework, possibly extending existing libraries like TensorFlow Federated or PySyft. Performance optimization will be crucial to reduce computational overhead and latency introduced by encryption and secure computation. Compatibility testing will be conducted to ensure integration with the resource allocation mechanism and overall system architecture.

*4.1.11 Security Evaluation.* The security mechanisms will be evaluated through attack simulations, testing the effectiveness of the protocols in preventing data leakage and model theft. Performance metrics will be collected to measure the impact of security mechanisms on training performance, including computational overhead and training time. A trade-off analysis will be performed to balance the level of security with training efficiency, aiming to find optimal configurations.

## 4.2 Developing Fault Tolerance Mechanisms

*4.2.1 Mechanism Design.* Fault tolerance strategies suitable for wide-area network environments will be designed. Mechanisms to detect node failures and communication breakdowns promptly will be implemented, along with methods for periodic saving of model states and efficient recovery in case of failures. Strategies to reassign tasks from failed nodes to active ones without disrupting the training process will also be developed.

*4.2.2 Implementation and Testing.* These mechanisms will be incorporated into the distributed training system. Stress testing will be conducted by simulating high node failure rates and network instability to test the robustness of the fault tolerance mechanisms. Performance monitoring will track the impact on training throughput and model convergence during fault conditions.

*4.2.3 Evaluation Metrics.* The effectiveness of fault tolerance mechanisms will be assessed through metrics such as recovery time, measuring the time taken to detect failures and recover from them. The availability of the system will be evaluated to determine the overall availability of the system despite node failures. The impact on training performance will be analyzed to identify any delays or degradations due to fault tolerance processes.

## 4.3 Experimental Evaluation with Real-world LLM Training

*4.3.1 Resources and Datasets.* To validate the proposed system, we will conduct real-world LLM training using GPT-Neo, an open-source language model based on GPT-2, available in the Hugging-Face repository. Building upon the reasoning LLM project to which we contributed previously, the goal is to efficiently train a larger model that enhances the logical reasoning capabilities demonstrated in our prior work. Currently, on our home-built cluster, we cannot train models beyond 100–120 million parameters without deploying data or model parallelization. Although we have exceeded this size using Georgia Tech's HPC infrastructure, these require leveraging expensive and in-demand GPU units. Learning how to leverage less in-demand and less expensive assets provides options to control the cost curves of the project.

The model is designed to perform logical reasoning tasks, specifically focusing on solving Boolean satisfiability (SAT) problems using Chain-of-Thought (CoT) reasoning. In our previous work, we developed a decoder-only Transformer capable of solving SAT problems through backtracking and deduction, implementing an algorithm equivalent to the DPLL SAT-solving algorithm. By leveraging the distributed training capabilities of the proposed system in the OS Pool environment, we aim to scale up this model, potentially improving its reasoning abilities and generalization to more complex logical tasks.

The training data consists of SAT instances encoded in the DIMACS format, encompassing varying numbers of variables and clauses to challenge the model's reasoning capabilities. These datasets include both randomly generated 3-SAT formulas and specially constructed instances designed to test the model's ability to generalize and perform deductive reasoning. The model will be trained on algorithmic traces ("reasoning paths") of the DPLL algorithm, providing the Chain-of-Thought sequences that guide the learning process.

*4.3.2 Training Procedure.* The model and training process will be configured to run across multiple heterogeneous nodes using the developed resource allocation and fault tolerance mechanisms. Currently, the ability to model or data parallelize is already implemented by the OS Pool itself. The training will involve multiple epochs over the dataset, and checkpoints will be saved periodically to allow for recovery in case of node failures.

*4.3.3 Data Collection Instruments.* Data will be collected using system logs via the OS Pool's internal API, which will provide automated logging of resource allocation decisions, node statuses, and fault events. Performance metrics will be gathered using monitoring tools to track training throughput, model convergence rates, and computational overhead. Security monitoring tools will be employed to detect and record any security incidents or attempted attacks.

## 5 DATA ANALYSIS

The primary goal of this section is to outline how the collected data will be transformed into actionable results that address each research question. For each question, the analysis will detail the data to be collected, the processing methods, the statistical analyses to be performed, and the rationale behind these choices.

### 5.1 Analysis for RQ1: Adaptation of the Gale-Shapley Algorithm

*5.1.1 Data Collection.* To evaluate the effectiveness of the adapted algorithm, data will be collected in three key areas: matching stability, resource utilization, and training performance.

Firstly, for matching stability, logs of task-resource assignments will be collected over time. This includes recording each assignment made by the algorithm, along with timestamps, to track the sequence of matches. Additionally, the study will monitor the number of rematches or reallocations that occur due to changes in resource availability, such as nodes joining or leaving the network.

Secondly, for resource utilization, detailed statistics on CPU, GPU, and memory usage will be gathered from each node participating in the distributed training. Network bandwidth and latency measurements will also be collected to understand the communication overhead and network conditions affecting the system.

Thirdly, for training performance, the total training time required to reach convergence will be recorded, as well as model convergence metrics such as the loss function values at each epoch. Training throughput, measured as the number of samples processed per second, will also be tracked to assess the efficiency of the training process.

*5.1.2 Data Processing and Statistical Analysis.* The collected data will be processed and analyzed using appropriate statistical methods to test the hypothesis.

For matching stability analysis, the study will calculate the *Matching Stability Index (MSI)*, which is defined as the ratio of stable matches to the total number of matches over the training period. Time-series analysis will be performed on the MSI to assess fluctuations over time and to identify patterns or trends in matching stability. Statistical process control charts will be used to detect any significant deviations from expected stability levels.

For resource utilization analysis, the research will compute the average and standard deviation of resource utilization metrics across all nodes. Analysis of Variance (ANOVA) will be employed to compare resource utilization efficiency between the adapted Gale-Shapley algorithm and baseline allocation methods. Regression analysis will be conducted to explore the relationship between resource utilization and training performance metrics.

For training performance analysis, the total training times achieved using the adapted algorithm will be compared versus those obtained with baseline methods. Independent samples t-tests will be used to assess whether the differences in training times are statistically significant. Model convergence rates will be analyzed by plotting loss curves over epochs for both methods.

*5.1.3 Validity Justification.* The statistical methods chosen are appropriate and valid for the types of data collected. Time-series analysis and statistical process control charts are suitable for detecting trends and anomalies over time. ANOVA is appropriate for comparing means across multiple groups. Regression analysis helps in understanding the relationship between resource utilization and training performance. Independent samples t-tests are suitable for comparing the means of two independent groups.

### 5.2 Analysis for RQ2: Security Mechanisms Effectiveness

*5.2.1 Data Collection.* To evaluate the effectiveness and impact of the security mechanisms, data will be collected related to security incidents and training performance under both secure and non-secure conditions.

For security incident data, simulated attack scenarios will be conducted, including attempts at data leakage, model inversion attacks, and unauthorized access to model parameters. The outcomes of these simulations will be recorded. Logs of any security incidents detected by monitoring tools during the training process will be collected.

For performance impact data, training times will be measured with and without the implementation of security mechanisms. Computational overhead metrics, such as CPU usage, memory consumption, and encryption/decryption times, will be recorded. Training throughput and model convergence rates will also be tracked under both secure and non-secure conditions.

*5.2.2 Data Processing and Statistical Analysis.* For security effectiveness analysis, the study will calculate the *Attack Success Rate (ASR)* for each simulated attack scenario. Chi-square tests will be used to compare the ASRs between the secure and non-secure configurations. Detection rates and false positives reported by the security monitoring tools will also be analyzed.

For performance impact analysis, the percentage increase in training time resulting from the implementation of security mechanisms will be computed. Paired t-tests will be performed to assess the statistical significance of the differences in training times, computational overhead, and training throughput between the secure and non-secure conditions. Effect size measures, such as Cohen's d, will be calculated.

*5.2.3 Validity Justification.* Chi-square tests are valid for comparing categorical data, such as the success or failure of attacks. Paired t-tests are appropriate for comparing the means of two related groups. Calculating effect sizes complements the significance testing by indicating the practical importance of the findings.

### 5.3 Analysis for RQ3: Fault Tolerance Mechanisms Efficacy

*5.3.1 Data Collection.* To assess the efficacy of the fault tolerance mechanisms, data will be collected on fault tolerance metrics, system availability, and training efficiency during both normal operation and simulated failure conditions.

For fault tolerance data, all node failures that occur during the training process will be logged, including timestamps and types of failures. Recovery times will be recorded.

For system availability data, the overall uptime percentage of the system will be calculated. Mean Time Between Failures (MTBF) and Mean Time To Recovery (MTTR) will be computed.

For training efficiency data, the training throughput during normal operation and under failure conditions will be monitored. Model convergence metrics will be tracked throughout the training process.

*5.3.2 Data Processing and Statistical Analysis.* For recovery time analysis, the average recovery time across all recorded failures will be calculated. Survival analysis techniques, such as Kaplan-Meier estimators, will be used to model the time until recovery events occur. Hypothesis testing using the log-rank test will be performed.

For system availability analysis, availability metrics will be computed and compared to industry benchmarks. Reliability engineering methods will be employed to assess the robustness of the system.

For training efficiency and convergence analysis, time-series analysis will be used to examine the impact of node failures on training throughput. ANOVA will be used to compare the means of training throughput and convergence metrics.

*5.3.3 Validity Justification.* Survival analysis is appropriate for modeling time-to-event data. Reliability engineering methods provide a structured approach to evaluate system availability. ANOVA is suitable for comparing the means of more than two groups or conditions.

## 6 PLAN OF ATTACK

This section outlines the detailed steps and timeline for executing the proposed study over a 16-week semester, aligning with the typical 150–200 hours allocated for an independent research experience.

### 6.1 Necessary Resources

- **Computational Resources**: Access to the OS Pool computing infrastructure. The ability to model or data parallelize is already implemented by the OS Pool itself. Also, access to prototyping system to ensure we do not waste OS Pool resources. This prototype system will be a homemade cluster of 9 GPUs connected on a 10-switch.
- **Software Tools**: NS-3 for network simulation, TensorFlow Federated or PySyft for implementing security protocols, Horovod or custom-built solutions for distributed training.
- **Datasets**: SAT problem datasets in DIMACS format, including randomly generated 3-SAT formulas and specially constructed instances.
- **Models**: GPT-Neo models from the HuggingFace repository.

### 6.2 Schedule

**Week 1–2: Preparations and Resource Acquisition (15 hours)**

- Deepen understanding of existing algorithms, security mechanisms, and fault tolerance strategies.
- Configure development environment and obtain access to resources.
- Collect and preprocess datasets.

**Week 3–5: Algorithm and Mechanism Development (30 hours)**

- Develop compatibility metrics and modify preference lists.
- Design security protocols and fault tolerance mechanisms.

**Week 6–8: Implementation and Integration (35 hours)**

- **Core Algorithm Implementation (10 hours)**:
  - Gale-Shapley adaptation implementation
  - Resource allocation optimization framework
  - Basic testing and validation
- **Security Protocol Integration (10 hours)**:
  - SMPC protocol implementation
  - Encryption framework integration
  - Security testing
- **Fault Tolerance Implementation (15 hours)**:
  - Checkpointing mechanism
  - Failure detection system
  - Recovery protocol testing

**Week 9–10: Testing and Validation (25 hours)**

- Create testing scenarios and collect preliminary data.
- Deploy the system on the OS Pool environment.

**Week 11–12: Data Collection (20 hours)**

- Conduct extensive training sessions.
- Introduce controlled variables like simulated attacks and node failures.

**Week 13–14: Data Analysis (20 hours)**

- Perform statistical tests and interpret results.
- Generate visualizations to present findings.

**Week 15: Documentation and Finalization (20 hours)**

- Compile methodology, results, and analysis into a coherent document.
- Proofread and revise as necessary.

**Week 16: Presentation and Reflection (10 hours)**

- Develop presentation materials and practice delivery.
- Reflect on the research process and outcomes.

## 6.3 Intended Publication Venues

The findings of this research are intended to be submitted to leading conferences and journals such as:

- **Conference on Machine Learning and Systems (MLSys)**: Relevant for its focus on the intersection of machine learning and systems, covering topics related to distributed computing, resource optimization, and scalable machine learning.
- **ACM Symposium on Cloud Computing (SoCC)**: Appropriate for work on distributed computing systems and cloud-based solutions.

These venues are suitable because they attract audiences interested in AI, distributed computing, and resource optimization, aligning with the research's scope.

## 7 SELF-CRITIQUE ON DATA ANALYSIS AND PLAN OF ATTACK

### 7.1 Feasibility Assessment

The proposed plan is feasible for an independent study within the allocated 175 hours. The tasks are distributed evenly across the semester, allowing for flexibility in case of unforeseen challenges. Access to necessary resources is secured, including:

**Computational Resources:** The OS Pool provides the required computational infrastructure, and access has been granted following training.

**Datasets and Models:** The SAT problem datasets are generated at will from our synthetic data generator which is already written, and GPT-Neo model are publicly available and suitable for the study.

Potential challenges include the complexity of integrating multiple system components and potential learning curves associated with new tools. However, the plan includes time for familiarization and testing, mitigating these risks.

### 7.2 Validity of Data Analysis Plan

The data analysis procedures are valid and appropriate for answering the research questions:

**Adaptation of Established Methods:** The statistical tests selected (e.g., ANOVA, t-tests, Chi-square tests) are standard in the field and suitable for the types of data collected.

**Hypothesis-Driven Analysis:** Each research question is accompanied by a clear hypothesis, and the analysis methods are designed to test these hypotheses effectively.

**Threats to Validity:** Potential threats include:

- *Sample Size Limitations:* Insufficient data may affect the statistical power of the tests. This will be mitigated by ensuring ample data collection during experiments.
- *Biases in Data Collection:* To prevent biases, data will be collected systematically and objectively, with careful monitoring to ensure accuracy.
- *External Validity:* The representativeness of the OS Pool environment to other distributed systems may limit generalizability. This is acknowledged, and findings will be contextualized accordingly.

### 7.3 Replicability of the Study

The methodology provides sufficient detail for replication:

**Detailed Procedures:** Step-by-step descriptions of algorithm adaptations, implementations, and experimental setups are provided.

**Specific Tools and Frameworks:** The use of tools like NS-3, TensorFlow Federated, and specific datasets/models is specified, guiding others in replicating the study.

**Data Analysis Methods:** Statistical tests and analysis procedures are explicitly outlined, allowing for reproduction of results.

To enhance replicability further, code repositories and detailed documentation will be maintained and made available where appropriate.

## 8 REFLECTION ON MODIFICATIONS FOR PREVIOUS SECTIONS

In refining the previous sections, I focused on integrating feedback to enhance clarity, specificity, and alignment with the research objectives. Key modifications include:

**Introduction and Motivation:** Expanded the discussion on the unique challenges posed by OS Pool environments, emphasizing resource heterogeneity and dynamic availability. This provides a stronger foundation for the research questions.

**Research Questions:** Refined the questions to be more specific and measurable. For instance, RQ1 now explicitly mentions maintaining matching stability and efficiency, highlighting the dual objectives of the algorithm adaptation.

**Methodology:** Added details to the experimental evaluation section, particularly regarding the resources and datasets. Included context about the limitations of current infrastructure and the motivation for leveraging the OS Pool.

**Critical Reflections:** Enhanced the critical reflections accompanying each research question to address ethical considerations, feasibility, and potential challenges more thoroughly.

These modifications strengthen the proposal by ensuring that each component is directly tied to the overarching goals of democratizing LLM training and that the proposed methods are feasible and well-justified.

## 9 CONCLUSION

This research proposal outlines a comprehensive methodology to address the challenges of democratizing LLM training in Open Science environments. By adapting the Gale-Shapley algorithm for dynamic resource allocation, implementing robust security mechanisms, and developing fault tolerance strategies, the proposed work aims to make large-scale language model training accessible to a broader range of researchers and institutions. Through theoretical development, system implementation, and empirical evaluation, the research seeks to contribute significant advancements to the fields of distributed computing and artificial intelligence, with broader impacts that extend to collaborative scientific research and decentralized computing initiatives.

## REFERENCES

[1] Xingrong Dong and Zhaoxian Wu. 2023. Distributed Online Learning With Adversarial Participants In An Adversarial Environment. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE.

[2] Shijin Duan and Chenghong Wang. 2024. SSNet: A Lightweight Multi-Party Computation Scheme for Practical Privacy-Preserving Machine Learning Service in the Cloud. *arXiv preprint arXiv:2406.02629* (2024).

[3] Qinfeng Li and Zhiqiang Shen. 2024. TransLinkGuard: Safeguarding Transformer Models Against Model Stealing in Edge Deployment. *arXiv preprint arXiv:2404.11121* (2024).

[4] Bo Liu and Chhaparia Rachita. 2024. Asynchronous Local-SGD Training for Language Modeling. *arXiv preprint arXiv:2401.09135* (2024).

[5] Deepak Narayanan and Amar Phanishayee. 2021. Memory-Efficient Pipeline-Parallel DNN Training. In *International Conference on Machine Learning*. PMLR, 7937–7947.

[6] Feng Niu and Benjamin Recht. 2011. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. In *Advances in neural information processing systems*. 693–701.

[7] Yosuke Oyama and Tal Ben-Nun. 2018. Accelerating Deep Learning Frameworks with Micro-batches. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 478–479.

[8] Jie Ren and Samyam Rajbhandari. 2021. ZeRO-Offload: Democratizing Billion-Scale Model Training. In *USENIX Annual Technical Conference (USENIX ATC 21)*.

[9] Liangqin Ren and Zeyan Liu. 2024. PrivDNN: A Secure Multi-Party Computation Framework for Deep Learning using Partial DNN Encryption. *Proceedings on Privacy Enhancing Technologies* (2024).

[10] Max Ryabinin and Eduard Gorbunov. 2022. Moshpit SGD: Communication-Efficient Decentralized Training on Heterogeneous Unreliable Devices. *arXiv preprint arXiv:2205.13636* (2022).

[11] Florian Tramer and Dan Boneh. 2019. Slalom: Fast, verifiable and private execution of neural networks in trusted hardware. In *International Conference on Learning Representations*.

[12] Ian Zhou and Tofigh Farzad. 2024. Secure Multi-Party Computation for Machine Learning: A Survey. *IEEE Access* (2024).