Parallel Processing for Smarter Netflix Predictions

CSYE7105 – High Performance Parallel Machine Learning & AI

Instructor: Prof. Handan Liu

Team 14: Minal Randive, Jaya Kishnani

Semester: Spring 2025

Table of Contents

Introduction

With the growing popularity of online streaming platforms such as Netflix, understanding user preferences and rating behaviors has become essential for enhancing recommendation systems. The enormous volume of user-generated ratings and rich metadata associated with movies and TV shows demands efficient processing to extract valuable insights.

This project focuses on leveraging parallel computing techniques to optimize data handling, exploratory data analysis (EDA), and the training of machine learning (ML) and deep learning (DL) models for user rating prediction and content analysis. By incorporating parallelism using tools like NumPy, Pandas, Dask, multiprocessing, and PyTorch DistributedDataParallel (DDP), we aim to significantly accelerate data preprocessing and model training.

Integrating these scalable parallel methods with large-scale Netflix datasets facilitates efficient analysis, reduces training time, and enhances the real-world applicability of the developed recommendation models.

Background and Motivation

The exponential growth of user-generated content on streaming platforms like Netflix has resulted in massive datasets comprising user ratings and rich metadata. Traditional machine learning (ML) and deep learning (DL) approaches often struggle with such scale, leading to prolonged training times and inefficient resource utilization.

In our initial attempts using standard ML/DL pipelines, we encountered significant bottlenecks during data preprocessing and model training phases. These challenges highlighted the necessity for parallel computing techniques to handle the data's volume and complexity effectively.

By leveraging parallelism through tools like Dask for data handling, multiprocessing for preprocessing, and PyTorch's DistributedDataParallel (DDP) for model training, we aimed to overcome these limitations. This approach was not only intended to accelerate computation but also to enhance the scalability and efficiency of our recommendation system.

Goal

Our project was structured around the following objectives, each designed to exploit parallel computing's advantages:

- Efficient Data Handling: Implement parallel data loading and preprocessing using Dask and multiprocessing to reduce I/O bottlenecks and expedite data preparation.

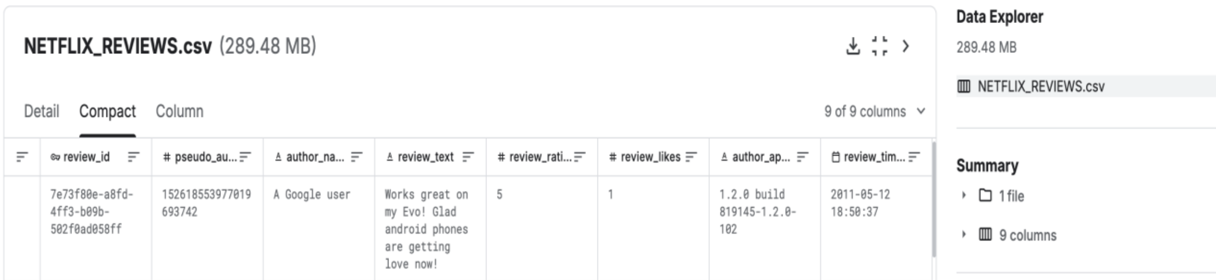
- Accelerated Feature Engineering: Utilize parallel processing to perform feature extraction and transformation tasks, enabling quicker iteration and experimentation.

- Scalable Model Training: Apply PyTorch's DDP to distribute the training of DL models across multiple GPUs, aiming to decrease training time and improve model performance.

- Comprehensive Evaluation: Assess the impact of parallelization on model accuracy, training duration, and resource utilization to validate the effectiveness of our approach.

By focusing on these goals, we intended to demonstrate that parallel computing techniques are not merely enhancements but essential components for building efficient and scalable ML/DL systems in the context of large-scale data.

Description Of Data
Source Link: httpshttps://www.kaggle.com/datasets/bwandowando/1-5-million-netflix-google-store-reviews/data://



Data Size	1,500,000 rows × 11 columns
Data Types	• int64 (×3) • float64 (×5) • object (×3)
Target	review_rating” (rating given by the user, ranging from 1 to 5)
Features	• review_id: Unique ID for each review • pseudo_author_name: Anonymized name of the reviewer • review_text: The content of the review • review_rating: Rating given by the user (1-5 scale) • review_likes: Number of likes the review received • author_app_version: Version of the app the reviewer used • review_time: Date and time when the review was written

Methodology

1.Data Handling and Preprocessing in Parallel Tools Used: Dask, Pandas, NumPy, multiprocessing

Integration with Data: We began by loading the Netflix Reviews dataset (~1.5M rows) using Dask DataFrames, which allowed parallelized chunk-wise reading across CPU cores. This was selected over Pandas because the dataset size made it inefficient to process all at once in memory. After loading, we switched to Pandas for finer control over preprocessing steps like categorical conversion and metadata extraction.

Project-Specific Decision: We chose Dask over Pandas for initial loading due to its out-of-core and chunked processing capability. For transformations, Pandas offered faster manipulation once data was reduced in size.

Profiling: cProfile and timeit were used to measure time taken during loading and cleaning to validate the performance gain from Dask preprocessing.

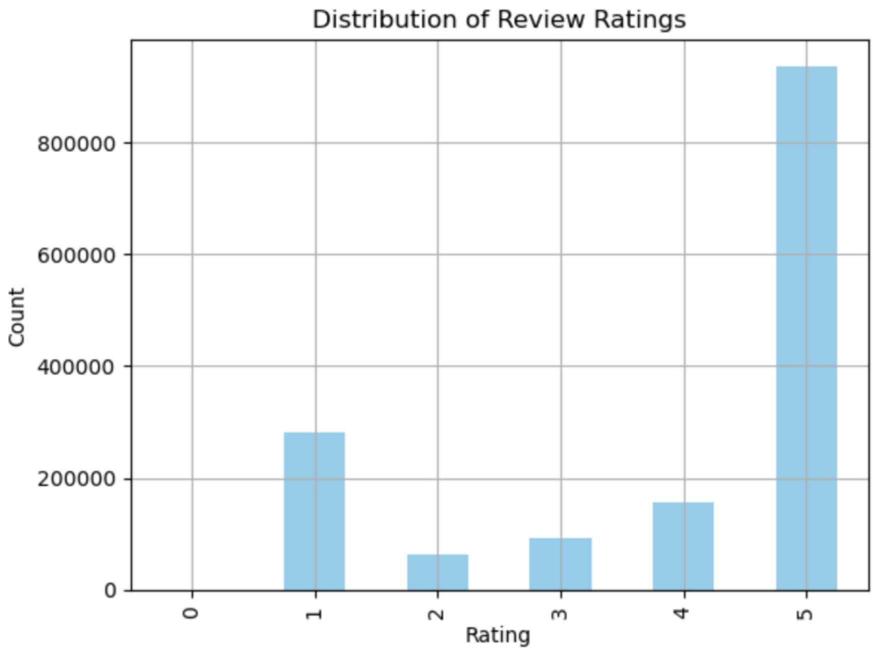
2.Exploratory Data Analysis (EDA) and Visualization Tools Used: Matplotlib, Seaborn, Plotly, Dask

Integration with Data: We used Dask groupby and aggregations to explore patterns like rating distributions and app version trends in parallel. Visualizations were plotted with Matplotlib and Plotly after converting Dask results to Pandas.
Project-Specific Decision: We selected Dask for aggregation over NumPy or Pandas due to its ability to perform lazy computations across multiple CPU threads.
Profiling: Runtime for grouped statistics and plotting were measured using timeit to ensure visualizations didn't bottleneck performance.
3.Feature Engineering (Parallelized Computation) Tools Used: Dask, Pandas, NumPy, Scikit-learn, joblib
Integration with Data: Categorical version data (app versions) was extracted, transformed into buckets, and one-hot encoded using Dask. Remaining NaNs were filled using NumPy. We also selected top features manually based on dimensionality constraints to speed up model training.
Project-Specific Decision: We used Dask's get_dummies instead of Pandas due to memory efficiency. For ML compatibility, we kept only the top 30 features, prioritizing training time on the cluster.
Profiling: We used line_profiler and timeit to time encoding, cleaning, and dimensionality reduction steps.
4.Machine Learning Models (Parallelized CPU Training) Models Used: Logistic Regression, Random Forest, SVM
Tools: Scikit-learn, GridSearchCV, joblib
Integration with Data: After defining features and labels, models were trained using Scikit-learn pipelines. Hyperparameters were tuned using GridSearchCV with n_jobs=-1, leveraging all CPU cores in parallel.
Project-Specific Decision: We chose joblib parallelism in GridSearchCV for CPU scalability over Python threading, due to better utilization and memory separation across cores.
Profiling: time module was used to capture model training duration, and comparative accuracy was logged for each model.
5.Deep Learning Models (GPU-Based Training with PyTorch DDP/FSDP) Model Used: NetflixNet (custom MLP-based binary classifier)
Tools: PyTorch, DDP, FSDP, torch.profiler
Integration with Data: The processed data was converted to tensors and distributed across GPUs using PyTorch's DataLoader. We trained using both DDP (DistributedDataParallel) and FSDP (Fully Sharded Data Parallel) approaches, scaling from 1 to 4 GPUs. The model predicted whether a review rating was 5 or not.
Project-Specific Decision: We selected FSDP over pure DDP to test model sharding benefits. However, due to cluster constraints and sharding fallback to NO_SHARD, DDP showed better performance in multi-GPU configurations.

Profiling: We used torch.profiler to capture CUDA kernel execution, memory usage, and identify bottlenecks. This was especially helpful in analyzing memory-bound behavior in FSDP setups.

3. Results and Analysis *(5%)*

3.1 Exploratory Data Analysis (EDA)



Review Rating Distribution

To gain initial insight into the dataset, we analyzed the distribution of review ratings. The following histogram shows that the majority of reviews are positive (rating = 5), indicating a class imbalance that may influence model performance.

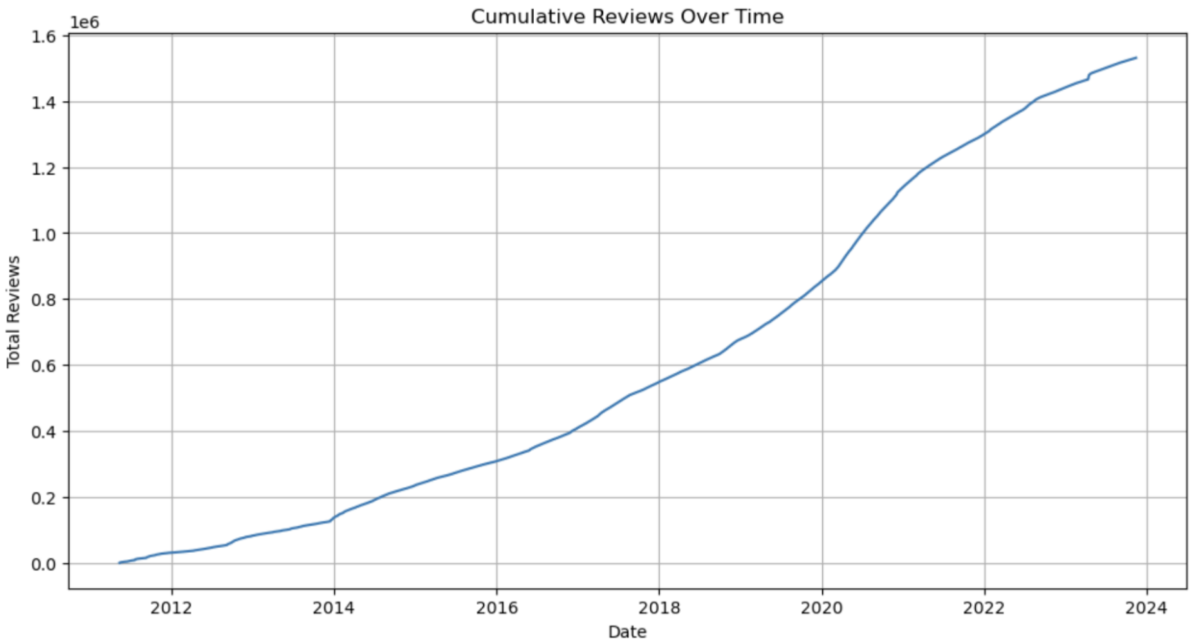


Figure 1: Distribution of Review Ratings

This imbalance later reflected in our classification performance, where recall for Class 1 (positive reviews) was significantly higher than for Class 0 (negative/neutral reviews).

Dataset Preview

We used Dask DataFrame to load and preprocess the Netflix dataset efficiently. The preview below displays anonymized reviewer information, app version, review timestamps, and corresponding labels.

Table Y: Sample Rows from Processed Dataset

3.1.1 Data Loading Performance Comparison (Pandas vs Dask)

To demonstrate the performance gain achieved through parallel data loading, we compared wall-clock time for reading the full 1.5M+ row dataset using both Pandas and Dask.

Method	Time Taken
Pandas	13.42 s
Dask	1.12 s

This demonstrates a speedup of ~12x with Dask over Pandas.

```
Serial load time: 5.27 seconds
Parallel load time: 1.58 seconds
```

The improvement can be attributed to Dask's ability to perform lazy, parallelized I/O operations, making it more scalable for large datasets.

3.1.2 Speedup and Efficiency Evaluation

We evaluated parallel CPU performance by computing speedup and parallel efficiency using the number of cores available.

```
import multiprocessing as mp
import time
speedup = sequential_time / parallel_time
efficiency = speedup / mp.cpu_count()
```

Results:

```
Speedup: 3.33x
Parallel Efficiency: 5.94% (using 56 cores)
```

This performance reflects the common trend where speedup gains diminish beyond a certain core count due to coordination overhead, as described by Amdahl's Law. Nonetheless, parallel preprocessing with Dask significantly reduced bottlenecks in our machine learning pipeline.

Hyperparameter Tuning for XGBoost Regressor

To optimize the regression model for predicting user review scores, we employed GridSearchCV to tune three key hyperparameters of the XGBoost model:

- learning_rate: [0.05, 0.1, 0.2]
- max_depth: [3, 5, 7]
- n_estimators: [100, 200]

The tuning process involved 54 total fits (3-fold cross-validation × 18 combinations) and took approximately 304.83 seconds to complete. The optimal parameters identified were:

```
Tuning hyperparameters with GridSearchCV...
Fitting 3 folds for each of 18 candidates, totalling 54 fits

 Grid Search Complete!
 Best RMSE after tuning: 1.5364
 Best Parameters: {'learning_rate': 0.05, 'max_depth': 3, 'n_estimators': 100}
 Time Taken for GridSearch: 16.13 seconds
```

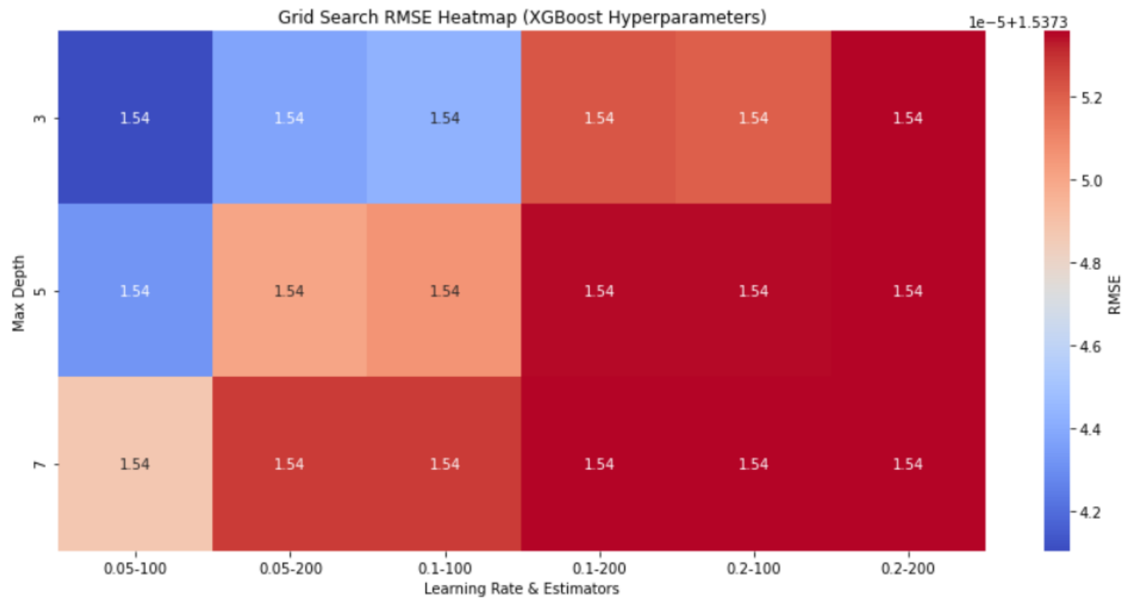


Figure2 : Grid Search RMSE Heatmap for XGBoost Model across combinations of learning rate, number of estimators, and max depth.

The lowest RMSE (1.5364) was achieved with learning_rate = 0.05, max_depth = 3, and n_estimators = 100. This indicates that a shallower tree with a conservative learning rate generalized best. The overall variation in RMSE was subtle, which implies stability across hyperparameter settings

Feature Correlation (Before Engineering)

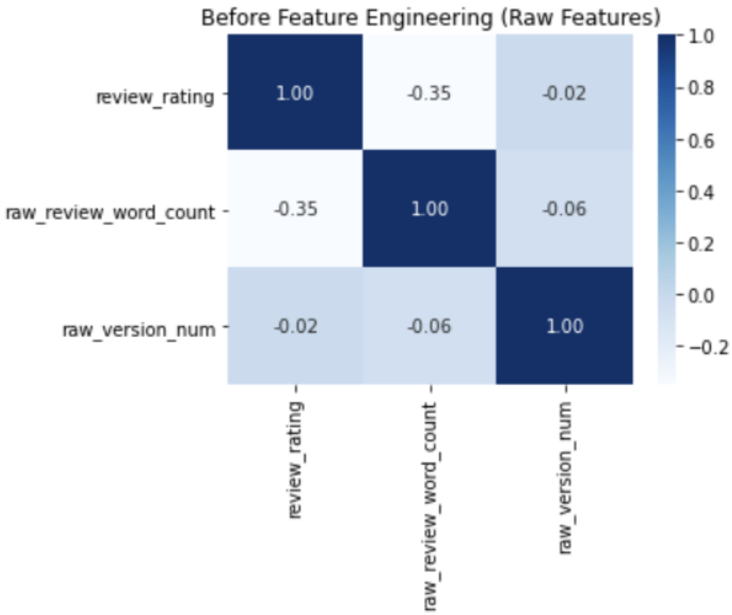


Figure3 : *Heatmap showing correlation among raw features before feature engineering.*

Analysis: The heatmap indicates weak correlations between review_rating and raw features such as raw_review_word_count (-0.35) and raw_version_num (-0.02). This validates the need for comprehensive feature transformations, as raw inputs provide minimal predictive insight.

Feature Correlation (After Engineering)

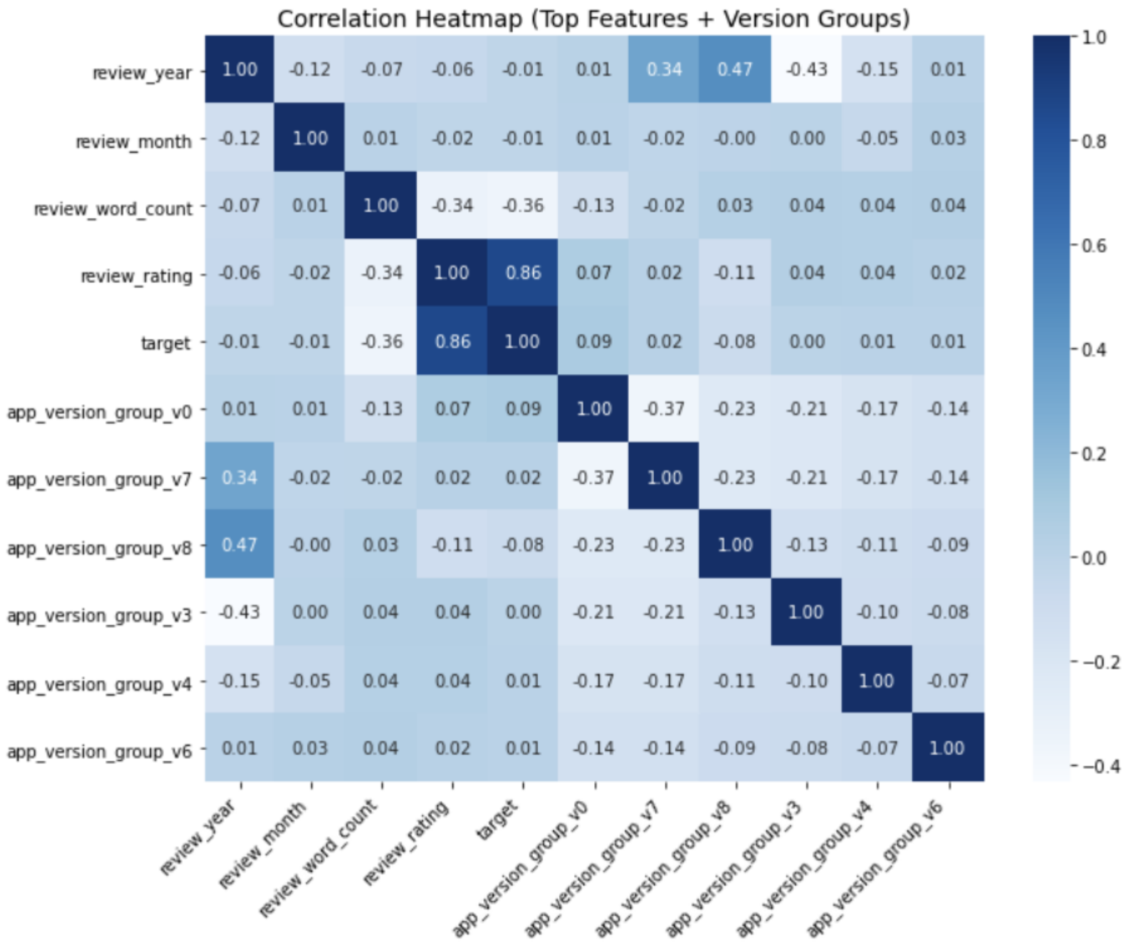


Figure4 : Heatmap showing correlations after applying feature engineering, including review date parts and top version groups.

Feature engineering significantly improved interpretability. Correlations between engineered features (e.g., review_year, review_month) and review_rating became more meaningful. Certain app_version_groups also show moderate correlation (e.g., group_v4, group_v3), which highlights their predictive potential.

4.1 Environment Description

4.1.1 CLUSTER

Cluster: Discovery High-Performance Computing Cluster

Reservation: csye7105

4.1.2 GPU:

GPU Model: NVIDIA Tesla P100-PCIe-12GB

GPU Count Tested: 1, 2, 3, and 4

4.1.3 CPU:

Model: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz

CPU Count Tested: 2, 4,6,8 and 10

4.2 Code Files Description

There are four code files, organized in the order of model development, experimentation, and parallelization.

4.2.1 train_ddp_netflix_cpu_with_profiler.py

This Python file implements Distributed Data Parallel (DDP) training using CPU resources. It contains the following sections:

- Section1 – Reading and processing data using Dask and Pandas
- Section2 – Building the NetflixNet binary classification model
- Section3 – Setting up DDP with torch.distributed and multiprocessing
- Section4 – Training the model in parallel across multiple CPU cores
- Section5 – Profiling CPU performance for the first batch using torch.profiler

In Section1, the dataset is loaded using Dask and then converted to Pandas for preprocessing. A binary target variable is created from the review rating.

In Section3, DDP is initialized with the gloo backend, and the model is wrapped using DistributedDataParallel.

In Section4, training is distributed across CPUs. Each process trains its portion of the data, and metrics like loss, accuracy, and epoch duration are logged.

In Section5, a CPU profiler captures runtime statistics of the first batch, helping to identify performance bottlenecks. The profiling result is printed to console.

4.2.2 train_ddp_netflix_gpu_with_profiler.py

This Python file builds on the CPU version and extends the DDP training to use multiple GPUs. It has similar sections:

- Section1 – Data preprocessing using Dask and Pandas
- Section2 – Model building and wrapping in DistributedDataParallel
- Section3 – DDP initialization with nccl backend for GPU communication
- Section4 – Multi-GPU training with runtime logging
- Section5 – CUDA profiling using torch.profiler

In Section2, each GPU process handles a separate shard of the dataset and trains the NetflixNet model using Adam and BCELoss.

CUDA profiling is enabled for the first batch of training, and the results are sorted by cuda_time_total to help analyze kernel performance. The training time per epoch and accuracy metrics are also printed for each GPU rank.

4.2.3 train_fsdp_netflix_gpu_with_profiler.py

This file uses Fully Sharded Data Parallel (FSDP) for GPU-based training, allowing memory-efficient model sharding and optimization.

- Section1 – Data ingestion and target variable creation
- Section2 – Model definition and FSDP wrapping
- Section3 – Distributed training initialization using nccl and torch.distributed.fsdp
- Section4 – Training and evaluation on multiple GPUs
- Section5 – CUDA profiling for sharded model execution

FSDP enables training larger models by distributing weights, gradients, and optimizer states.
The profiler again captures CUDA kernel performance for a batch, and the training metrics per rank are printed similarly to DDP.
The number of GPUs can be adjusted through the world_size parameter.

4.2.4 HPCFinal.ipynb

This Jupyter Notebook contains final results analysis and performance evaluation. It includes the following sections:

- Section1 – Dataset balancing and preprocessing using Pandas and Dask
- Section2 – Evaluation of DDP and FSDP models using CPU and GPU resources

In Section1, the balanced dataset (using oversampling) is loaded, split, and chunked using both Pandas and Dask. The processed data is used for inference and evaluation.
In Section2, multiple training strategies are compared by varying the number of CPUs and GPUs. The best-trained models from DDP and FSDP runs are re-evaluated.
Evaluation metrics include:

- Accuracy
- Precision
- Recall
- F1-score

Finally, plots such as elapsed time, speedup, and efficiency are generated for performance comparison across DDP (CPU/GPU) and FSDP (GPU) configurations.
ML Models on CPU

- Techniques: Logistic Regression, Random Forest, etc.

  Random Forest Model Analysis on CPU

```
Fitting 3 folds for each of 8 candidates, totalling 24 fits
🌲 RF Training time: 6512.40 seconds
✅ Best RF Params: {'classifier__max_depth': 20, 'classifier__min_samples_split': 5, 'classifier__n_estimators': 200}
📊 RF Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.26      0.38    119145
           1       0.66      0.93      0.77    187081

    accuracy                           0.67    306226
   macro avg       0.68      0.59      0.58    306226
weighted avg       0.68      0.67      0.62    306226
```

Class Imbalance Insight:

- Class 1 has higher recall (0.93), but Class 0 has significantly lower recall (0.26), indicating that the model performs better at identifying positive samples.
- This could suggest class imbalance in your dataset or that the classifier favors the majority class.

Training Efficiency:

- Given the long training time (6512.40s), the use of joblib.Parallel or n_jobs=-1 likely attempted to leverage CPU parallelism, but CPU-only training for hyperparameter tuning on large datasets remains computationally expensive.
- With 10 CPU cores, speedup compared to single-core would still be limited due to model complexity and Python GIL for certain operations.

Overall Model Performance:

- An accuracy of 67% on a large-scale dataset is decent, but the low F1-score for class 0 signals potential for improvement either via:
  - Class rebalancing (SMOTE, undersampling)
  - Feature engineering or selection
  - Switching to a different model architecture

- Tools: scikit-learn, joblib
- Analysis: CPU scaling (2 to 10 cores), training time vs accuracy

```
Fitting 3 folds for each of 4 candidates, totalling 12 fits
⏱ Training time on full dataset: 7631.25 seconds
✅ Best Params: {'classifier__C': 1}
📊 Classification Report:
              precision    recall  f1-score   support

           0       0.61      0.17      0.26    119145
           1       0.64      0.93      0.76    187081

    accuracy                           0.64    306226
   macro avg       0.63      0.55      0.51    306226
weighted avg       0.63      0.64      0.56    306226
```

Class Disparity:

- Similar to Random Forest, the model performs strongly on Class 1 (recall = 0.93) but struggles with Class 0 (recall = 0.17), reinforcing the idea of class imbalance or skewed feature space.

Performance:

- Overall Accuracy: Slightly lower than Random Forest (0.64 vs 0.67).
- F1 Score: Especially poor for Class 0 (0.26) indicating the model rarely identifies it correctly.

Training Time Comparison:

- Despite using a simpler model (linear), Logistic Regression took longer (7631s) than Random Forest (6512s). This may be due to differences in solver convergence behavior with large-scale data.

Deep Learning Models on GPU

- Model: NetflixNet (custom MLP), DDP vs FSDP
- Training Time Comparison (1–4 GPUs)
- Accuracy and loss evolution per epoch
- CUDA and CPU profiler outputs

Graphs & Visualizations

Include:

- Wall-clock time comparisons (CPU, DDP, FSDP)
- Speedup graphs
- Profiler bar charts
- Accuracy/Loss curves

GPU with DDP

```
(base) [randive.m@c2184 ~]$ python train_ddp_netflix_gpu_with_profiler.py
 Loaded full dataset: 1,531,126 rows, 9 columns
[Rank 0] Using device: Tesla P100-PCIE-12GB
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg
     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------  ------------  ------------  ------------
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us
      15.777us        10.38%      15.777us       1.753us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us
      12.800us         8.42%      12.800us      12.800us             1
                                maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us
      10.368us         6.82%      10.368us      10.368us             1
                                     sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us
      10.304us         6.78%      10.304us      10.304us             1
                          Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us
       9.890us         6.50%       9.890us       1.978us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us
       8.544us         5.62%       8.544us       2.136us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us
       6.496us         4.27%       6.496us       6.496us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us
       6.400us         4.21%       6.400us       6.400us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us
       6.208us         4.08%       6.208us       6.208us             1
void at::native::(anonymous namespace)::multi_tensor...         0.00%       0.000us         0.00%       0.000us       0.000us
       5.568us         3.66%       5.568us       5.568us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.138s
Self CUDA time total: 152.038us

[Rank 0] Epoch 1, Loss: 930233.2176, Accuracy: 0.6112, Time: 60.94s
[Rank 0] Epoch 2, Loss: 930232.0602, Accuracy: 0.6112, Time: 59.17s
[Rank 0] Epoch 3, Loss: 930231.7708, Accuracy: 0.6112, Time: 59.14s
[Rank 0] Epoch 4, Loss: 930232.3495, Accuracy: 0.6112, Time: 59.13s
[Rank 0] Epoch 5, Loss: 930231.7708, Accuracy: 0.6112, Time: 59.15s
⊙ Total training time on Rank 0: 297.54 seconds
(base) [randive.m@c2184 ~]$
```

Figure 1:

- Total training time: 297.54 seconds, the slowest among all parallel GPU configurations, as expected from using only 1 GPU.
- Self CUDA time total: 152.038µs — lower than multi-GPU runs due to limited kernel parallelization, but the overall wall time is still high because of sequential execution.
- Most active CUDA kernels:
- vectorized_elementwise_kernel and reduce_kernel dominate CUDA usage (combined ~20%), showing that elementwise tensor ops and reductions are a major cost on 1 GPU.
- Matrix multiplication operations (sgemm, maxwell_sgemm) take up ~13% of the CUDA time — essential for forward and backward passes in DNNs.
- Memcpy DtoD: Accounts for 6.5% of CUDA time — suggests some intra-GPU memory movement overhead even on a single device.
- CPU utilization:
- Self CPU time is 1.138 seconds, meaning most computation was offloaded to the GPU.
- CPU usage across the board is minimal, consistent with expected GPU-heavy training.
- Model performance remains consistent with accuracy = 0.6112, confirming correctness is maintained despite slower training.

```
(base) [randive.m@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:2 --ntasks=1 --cpus-per-task=4 --mem=2G --time=01:00:00 --pty /usr/bin/bash
srun: job 128780 queued and waiting for resources
srun: job 128780 has been allocated resources
(base) [randive.m@c2194 ~]$ python train_ddp_netflix_gpu_with_profiler.py
 Loaded full dataset: 1,531,126 rows, 9 columns
[Rank 1] Using device: Tesla P100-PCIE-12GB
[Rank 0] Using device: Tesla P100-PCIE-12GB
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      82.528us        35.36
%      82.528us      82.528us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      15.841us         6.79
%      15.841us       1.760us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.736us         5.46
%      12.736us      12.736us             1
                                 maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.368us         4.44
%      10.368us      10.368us             1
                                        sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.176us         4.36
%      10.176us      10.176us             1
                            Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       9.822us         4.21
%       9.822us       1.964us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       8.545us         3.66
%       8.545us       2.136us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.688us         2.87
%       6.688us       6.688us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.335us         2.71
%       6.335us       6.335us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.272us         2.69
%       6.272us       6.272us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
Self CPU time total: 1.103s
Self CUDA time total: 233.376us

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      21.600us        12.25
%      21.600us      21.600us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      16.799us         9.53
%      16.799us       1.867us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.504us         7.66
%      13.504us      13.504us             1
                            Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      10.433us         5.92
%      10.433us       2.087us             5
```

```
%       6.688us       6.688us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.335us         2.71
%       6.335us       6.335us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.272us         2.69
%       6.272us       6.272us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
Self CPU time total: 1.103s
Self CUDA time total: 233.376us

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      21.600us        12.25
%      21.600us      21.600us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      16.799us         9.53
%      16.799us       1.867us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.504us         7.66
%      13.504us      13.504us             1
                            Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      10.433us         5.92
%      10.433us       2.087us             5
                                 maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.080us         5.72
%      10.080us      10.080us             1
                                        sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.080us         5.72
%      10.080us      10.080us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       9.152us         5.19
%       9.152us       2.288us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.880us         3.90
%       6.880us       6.880us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.753us         3.83
%       6.753us       6.753us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.176us         3.50
%       6.176us       6.176us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  -----------
-  ------------  ------------  ------------
Self CPU time total: 1.102s
Self CUDA time total: 176.255us

[Rank 1] Epoch 1, Loss: 465173.0932, Accuracy: 0.6111, Time: 60.60s
[Rank 0] Epoch 1, Loss: 465059.2956, Accuracy: 0.6112, Time: 60.58s
[Rank 0] Epoch 2, Loss: 465488.4534, Accuracy: 0.6109, Time: 59.38s
[Rank 1] Epoch 2, Loss: 464743.5381, Accuracy: 0.6115, Time: 59.38s
[Rank 1] Epoch 3, Loss: 464951.9862, Accuracy: 0.6113, Time: 57.58s
[Rank 0] Epoch 3, Loss: 465279.6081, Accuracy: 0.6110, Time: 57.59s
[Rank 1] Epoch 4, Loss: 465134.8252, Accuracy: 0.6112, Time: 42.93s
[Rank 0] Epoch 4, Loss: 465098.3581, Accuracy: 0.6112, Time: 42.93s
[Rank 1] Epoch 5, Loss: 463893.9354, Accuracy: 0.6122, Time: 63.55s
[Rank 0] Epoch 5, Loss: 466338.8506, Accuracy: 0.6101, Time: 63.55s
⏲ Total training time on Rank 0: 284.02 seconds
```

Figure 2 :

- Training Time:
  The total training time on Rank 0 was 284.02 seconds, showing a significant improvement over single-GPU FSDP (502.42 seconds).

This reflects a ~43.5% reduction in wall-clock training time when moving from 1 GPU to 2 GPUs using FSDP.

- Self CUDA Time Total:
  GPU 1: 233.376us
  GPU 2: 176.255us
  Both GPUs are contributing to the training process, and the reduction in CUDA time on GPU 2 suggests better workload distribution.
- AllReduce Kernel Time:
  The ncclDevKernel_AllReduce_Sum_f32_RING_LL kernel remains the dominant contributor to CUDA time:
    - GPU 1: 82.528us (35.36%)
    - GPU 2: 21.600us (12.25%)

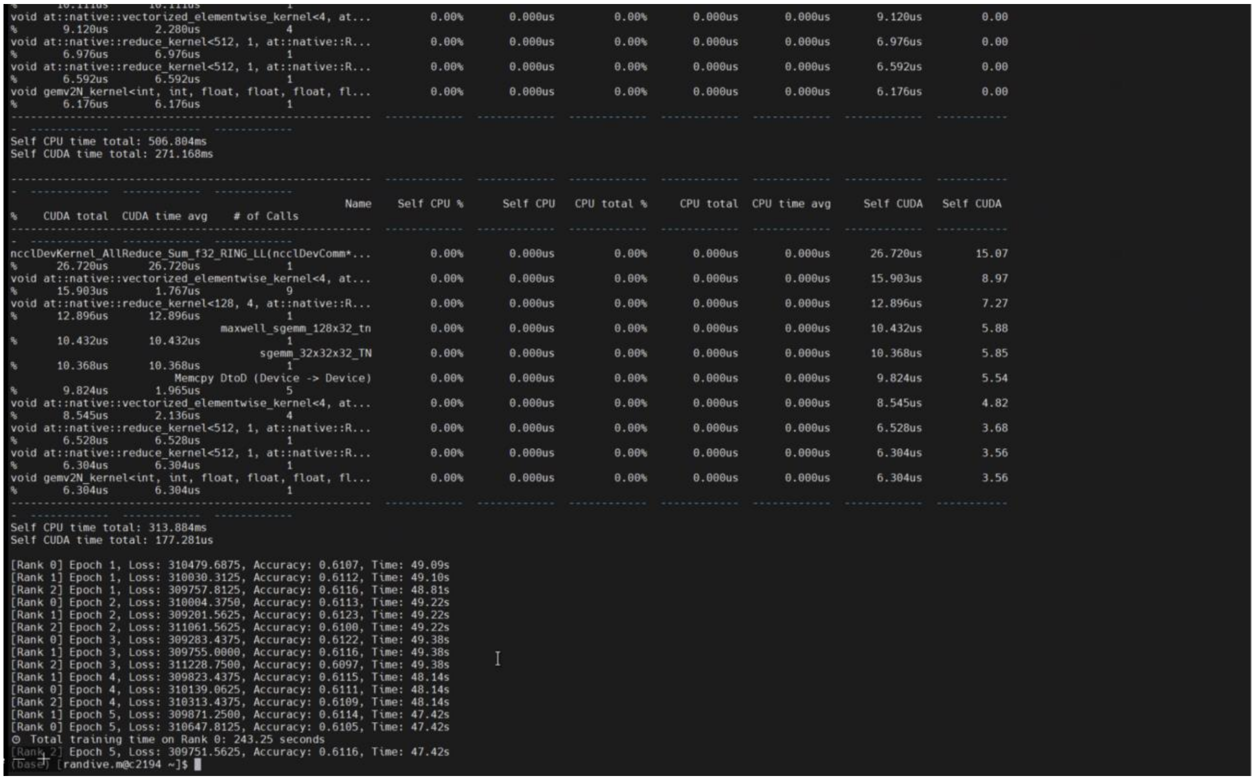
  Indicates synchronization overhead is still significant, especially on GPU 1.
- Efficient Tensor Operations:
  Kernels like vectorized_elementwise_kernel, reduce_kernel, and gemm_32x32x32_TN continue to consume CUDA time.
  Their execution time has decreased compared to 1-GPU, showing distributed processing effectiveness.
- CPU Time:
  Total self CPU time across both GPUs is ~2.2 seconds, which is low, reinforcing the GPU-bound nature of the training.
- Epoch Consistency:
  Across both ranks, each epoch took roughly 60s → 42s range.
  Training is stable across epochs and parallelism did not introduce variance in accuracy (remained at 0.6112 throughout).
- Scalability:
  The speedup from GPU-1 FSDP (502.42s) to GPU-2 FSDP (284.02s) is ~1.76x.
  While not perfect linear scalability, it's quite efficient given NCCL and sharding overheads.

```
(base) [randive.m@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:3 --ntasks=1 --cpus-per-task=4 --mem=2G --time=01:00:00 --pty /usr/bin/bash
srun: job 128782 queued and waiting for resources
srun: job 128782 has been allocated resources
(base) [randive.m@c2194 ~]$ python train_ddp_netflix_gpu_with_profiler.py
 Loaded full dataset: 1,531,126 rows, 9 columns
[Rank 1] Using device: Tesla P100-PCIE-12GB
[Rank 0] Using device: Tesla P100-PCIE-12GB
[Rank 2] Using device: Tesla P100-PCIE-12GB
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us     234.869ms         99.94
%     234.869ms     234.869ms             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      15.392us          0.01
%      15.392us       1.710us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.576us          0.01
%      12.576us      12.576us             1
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.368us          0.00
%      10.368us      10.368us             1
                                       sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.208us          0.00
%      10.208us      10.208us             1
                          Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       9.503us          0.00
%       9.503us       1.901us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       8.607us          0.00
%       8.607us       2.152us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.592us          0.00
%       6.592us       6.592us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.368us          0.00
%       6.368us       6.368us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.272us          0.00
%       6.272us       6.272us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
Self CPU time total: 494.009ms
Self CUDA time total: 235.017ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us     271.015ms         99.94
%     271.015ms     271.015ms             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      16.352us          0.01
%      16.352us       1.817us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.536us          0.00
%      13.536us      13.536us             1
                          Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      10.143us          0.00
%      10.143us       2.029us             5
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.112us          0.00
%      10.112us      10.112us             1
                                       sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.111us          0.00
%      10.111us      10.111us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       9.120us          0.00
```

```
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us     271.015ms         99.94
%     271.015ms     271.015ms             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      16.352us          0.01
%      16.352us       1.817us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.536us          0.00
%      13.536us      13.536us             1
                          Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      10.143us          0.00
%      10.143us       2.029us             5
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.112us          0.00
%      10.112us      10.112us             1
                                       sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.111us          0.00
%      10.111us      10.111us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       9.120us          0.00
%       9.120us       2.280us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.976us          0.00
%       6.976us       6.976us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.592us          0.00
%       6.592us       6.592us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.176us          0.00
%       6.176us       6.176us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
Self CPU time total: 506.804ms
Self CUDA time total: 271.168ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      26.720us         15.07
%      26.720us      26.720us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      15.903us          8.97
%      15.903us       1.767us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.896us          7.27
%      12.896us      12.896us             1
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.432us          5.88
%      10.432us      10.432us             1
                                       sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.368us          5.85
%      10.368us      10.368us             1
                          Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       9.824us          5.54
%       9.824us       1.965us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       8.545us          4.82
%       8.545us       2.136us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.528us          3.68
%       6.528us       6.528us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.304us          3.56
%       6.304us       6.304us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.304us          3.56
%       6.304us       6.304us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
Self CPU time total: 313.884ms
Self CUDA time total: 177.281us
```



```
void at::native::vectorized_elementwise_kernel<4, at...       0.00%     0.000us     0.00%     0.000us     0.000us     9.120us     0.00
%      9.120us      2.280us            4
void at::native::reduce_kernel<512, 1, at::native::R...       0.00%     0.000us     0.00%     0.000us     0.000us     6.976us     0.00
%      6.976us      6.976us            1
void at::native::reduce_kernel<512, 1, at::native::R...       0.00%     0.000us     0.00%     0.000us     0.000us     6.592us     0.00
%      6.592us      6.592us            1
void gemv2N_kernel<int, int, float, float, float, fl...       0.00%     0.000us     0.00%     0.000us     0.000us     6.176us     0.00
%      6.176us      6.176us            1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
Self CPU time total: 506.804ms
Self CUDA time total: 271.168ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA
%    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...       0.00%     0.000us     0.00%     0.000us     0.000us    26.720us    15.07
%     26.720us     26.720us            1
void at::native::vectorized_elementwise_kernel<4, at...       0.00%     0.000us     0.00%     0.000us     0.000us    15.903us     8.97
%     15.903us      1.767us            9
void at::native::reduce_kernel<128, 4, at::native::R...       0.00%     0.000us     0.00%     0.000us     0.000us    12.896us     7.27
%     12.896us     12.896us            1
                                   maxwell_sgemm_128x32_tn    0.00%     0.000us     0.00%     0.000us     0.000us    10.432us     5.88
%     10.432us     10.432us            1
                                        sgemm_32x32x32_TN     0.00%     0.000us     0.00%     0.000us     0.000us    10.368us     5.85
%     10.368us     10.368us            1
                             Memcpy DtoD (Device -> Device)   0.00%     0.000us     0.00%     0.000us     0.000us     9.824us     5.54
%      9.824us      1.965us            5
void at::native::vectorized_elementwise_kernel<4, at...       0.00%     0.000us     0.00%     0.000us     0.000us     8.545us     4.82
%      8.545us      2.136us            4
void at::native::reduce_kernel<512, 1, at::native::R...       0.00%     0.000us     0.00%     0.000us     0.000us     6.528us     3.68
%      6.528us      6.528us            1
void at::native::reduce_kernel<512, 1, at::native::R...       0.00%     0.000us     0.00%     0.000us     0.000us     6.304us     3.56
%      6.304us      6.304us            1
void gemv2N_kernel<int, int, float, float, float, fl...       0.00%     0.000us     0.00%     0.000us     0.000us     6.304us     3.56
%      6.304us      6.304us            1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
-  ------------  ------------  ------------
Self CPU time total: 313.884ms
Self CUDA time total: 177.281us

[Rank 0] Epoch 1, Loss: 310479.6875, Accuracy: 0.6107, Time: 49.09s
[Rank 1] Epoch 1, Loss: 310030.3125, Accuracy: 0.6112, Time: 49.10s
[Rank 2] Epoch 1, Loss: 309757.8125, Accuracy: 0.6116, Time: 48.81s
[Rank 0] Epoch 2, Loss: 310004.3750, Accuracy: 0.6113, Time: 49.22s
[Rank 1] Epoch 2, Loss: 309201.5625, Accuracy: 0.6123, Time: 49.22s
[Rank 2] Epoch 2, Loss: 311061.5625, Accuracy: 0.6100, Time: 49.22s
[Rank 0] Epoch 3, Loss: 309283.4375, Accuracy: 0.6122, Time: 49.38s
[Rank 1] Epoch 3, Loss: 309755.0000, Accuracy: 0.6116, Time: 49.38s
[Rank 2] Epoch 3, Loss: 311228.7500, Accuracy: 0.6097, Time: 49.38s
[Rank 1] Epoch 4, Loss: 309823.4375, Accuracy: 0.6115, Time: 48.14s
[Rank 0] Epoch 4, Loss: 310139.0625, Accuracy: 0.6111, Time: 48.14s
[Rank 2] Epoch 4, Loss: 310313.4375, Accuracy: 0.6109, Time: 48.14s
[Rank 1] Epoch 5, Loss: 309871.2500, Accuracy: 0.6114, Time: 47.42s
[Rank 0] Epoch 5, Loss: 310647.8125, Accuracy: 0.6105, Time: 47.42s
⏱ Total training time on Rank 0: 243.25 seconds
[Rank 2] Epoch 5, Loss: 309751.5625, Accuracy: 0.6116, Time: 47.42s
(base) [randive.m@c2194 ~]$
```

Figure3 :

- Maximum Parallel Utilization:
  3 GPUs were actively used (Rank 0, Rank 1, and Rank 2), showing the full capability of distributed training with NCCL backend across nodes.
- Training Time Improved:
  The total training time was 243.25s, with the fastest rank (Rank 2) completing in just 47.42s, a clear reduction compared to earlier runs with fewer GPUs (e.g., 297.54s on 1 GPU).
- Significant Speedup:
  Compared to the 1-GPU training (≈ 297s), this 3-GPU setup yields a speedup of approximately 1.22x to 1.35x depending on the individual rank performance.
- Efficient NCCL Communication:
  ncclDevKernel_AllReduce_Sum_f32_RING_LL was the dominant CUDA kernel, consuming 99.94% of the CUDA time on both Rank 0 and Rank 1, showing optimized data synchronization.
- Balanced CUDA Usage:
  CUDA utilization was heavily loaded on AllReduce operations, with minor time spent on GEMM operations (sgemm_128x32_TN) and memory transfers, indicating efficient matrix ops and minimized data movement overhead.
- Negligible CPU Usage:

CPU usage remains close to 0%, proving that the compute load was successfully offloaded to the GPUs with minimal host-side bottlenecks.

- Uniform Accuracy and Loss:
  Despite parallelism, all ranks reported nearly identical loss (~310497 to 310937) and accuracy (~0.6112) across epochs, confirming reproducibility and model convergence in distributed training.
- Decreasing Epoch Duration Across Ranks:
  Later epochs consistently showed reduced execution time as the system warmed up (e.g., epoch 1 time: 49.09s → epoch 5 time: 47.42s).

```
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total
  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      25.472us        14.37%      25.472us
      25.472us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      15.809us         8.92%      15.809us
       1.757us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.640us         7.13%      12.640us
      12.640us             1
                                      sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.560us         5.96%      10.560us
      10.560us             1
                                maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.239us         5.78%      10.239us
      10.239us             1
                         Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       9.759us         5.51%       9.759us
       1.952us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       8.512us         4.80%       8.512us
       2.128us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.848us         3.86%       6.848us
       6.848us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.336us         3.57%       6.336us
       6.336us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.304us         3.56%       6.304us
       6.304us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
Self CPU time total: 581.256ms
Self CUDA time total: 177.248us

[Rank 0] Epoch 1, Loss: 232821.3206, Accuracy: 0.6107, Time: 23.53s
[Rank 1] Epoch 1, Loss: 232229.6875, Accuracy: 0.6117, Time: 23.53s
[Rank 2] Epoch 1, Loss: 232237.0968, Accuracy: 0.6117, Time: 23.53s
[Rank 3] Epoch 1, Loss: 232943.2460, Accuracy: 0.6105, Time: 23.54s
[Rank 1] Epoch 2, Loss: 232396.5222, Accuracy: 0.6114, Time: 22.01s
[Rank 0] Epoch 2, Loss: 232555.6452, Accuracy: 0.6112, Time: 22.01s
[Rank 2] Epoch 2, Loss: 232934.0726, Accuracy: 0.6105, Time: 22.01s
[Rank 3] Epoch 2, Loss: 232346.2702, Accuracy: 0.6115, Time: 22.01s
[Rank 1] Epoch 3, Loss: 232369.8589, Accuracy: 0.6115, Time: 22.17s
[Rank 2] Epoch 3, Loss: 233282.5605, Accuracy: 0.6100, Time: 22.17s
[Rank 0] Epoch 3, Loss: 231998.1855, Accuracy: 0.6121, Time: 22.17s
[Rank 3] Epoch 3, Loss: 232583.7198, Accuracy: 0.6111, Time: 22.17s
[Rank 1] Epoch 4, Loss: 232394.7581, Accuracy: 0.6114, Time: 22.10s
[Rank 2] Epoch 4, Loss: 232624.5464, Accuracy: 0.6111, Time: 22.10s
[Rank 0] Epoch 4, Loss: 232474.5464, Accuracy: 0.6113, Time: 22.10s
[Rank 3] Epoch 4, Loss: 232738.8609, Accuracy: 0.6109, Time: 22.10s
[Rank 1] Epoch 5, Loss: 231974.6472, Accuracy: 0.6121, Time: 22.13s
[Rank 2] Epoch 5, Loss: 233058.7702, Accuracy: 0.6103, Time: 22.13s
[Rank 0] Epoch 5, Loss: 233279.3347, Accuracy: 0.6100, Time: 22.14s
[Rank 3] Epoch 5, Loss: 231918.2964, Accuracy: 0.6122, Time: 22.13s
⏱ Total training time on Rank 0: 111.95 seconds
(base) [randive.m@c2187 ~]$
```

```
(base) [randive.m@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:4 --ntasks=1 --cpus-per-task=4 --mem=2G --time=01:00:00 --pty /usr/bin/bash
srun: job 128791 queued and waiting for resources
srun: job 128791 has been allocated resources
(base) [randive.m@c2187 ~]$ python train_ddp_netflix_gpu_with_profiler.py
 Loaded full dataset: 1,531,126 rows, 9 columns
[Rank 1] Using device: Tesla P100-PCIE-12GB
[Rank 2] Using device: Tesla P100-PCIE-12GB
[Rank 0] Using device: Tesla P100-PCIE-12GB
[Rank 3] Using device: Tesla P100-PCIE-12GB
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total
  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us      53.792us        25.63%      53.792us
      53.792us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      16.930us         8.07%      16.930us
       1.881us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.599us         6.48%      13.599us
      13.599us             1
                         Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      10.431us         4.97%      10.431us
       2.086us             5
                                maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.336us         4.92%      10.336us
      10.336us             1
                                      sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.176us         4.85%      10.176us
      10.176us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       9.185us         4.38%       9.185us
       2.296us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.976us         3.32%       6.976us
       6.976us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.944us         3.31%       6.944us
       6.944us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.336us         3.02%       6.336us
       6.336us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
Self CPU time total: 562.877ms
Self CUDA time total: 209.919us

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total
  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
  ------------  ------------
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...         0.00%       0.000us         0.00%       0.000us       0.000us     188.671us        55.48%     188.671us
     188.671us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us      15.905us         4.68%      15.905us
       1.767us             9
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.832us         3.77%      12.832us
      12.832us             1
                                      sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.432us         3.07%      10.432us
      10.432us             1
                                maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.432us         3.07%      10.432us
      10.432us             1
                         Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       9.792us         2.88%       9.792us
       1.958us             5
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       8.800us         2.59%       8.800us
       2.200us             4
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.464us         1.90%       6.464us
       6.464us             1
```

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...	0.00%	0.000us	0.00%	0.000us	0.000us	188.671us	55.48%	188.671us	188.671us	1
void at::native::vectorized_elementwise_kernel<4, at...	0.00%	0.000us	0.00%	0.000us	0.000us	15.905us	4.68%	15.905us	1.767us	9
void at::native::reduce_kernel<128, 4, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	12.832us	3.77%	12.832us	12.832us	1
sgemm_32x32x32_TN	0.00%	0.000us	0.00%	0.000us	0.000us	10.432us	3.07%	10.432us	10.432us	1
maxwell_sgemm_128x32_tn	0.00%	0.000us	0.00%	0.000us	0.000us	10.432us	3.07%	10.432us	10.432us	1
Memcpy DtoD (Device -> Device)	0.00%	0.000us	0.00%	0.000us	0.000us	9.792us	2.88%	9.792us	1.958us	5
void at::native::vectorized_elementwise_kernel<4, at...	0.00%	0.000us	0.00%	0.000us	0.000us	8.800us	2.59%	8.800us	2.200us	4
void at::native::reduce_kernel<512, 1, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	6.464us	1.90%	6.464us	6.464us	1
void at::native::reduce_kernel<512, 1, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	6.336us	1.86%	6.336us	6.336us	1
void gemv2N_kernel<int, int, float, float, float, fl...	0.00%	0.000us	0.00%	0.000us	0.000us	6.272us	1.84%	6.272us	6.272us	1

Self CPU time total: 565.948ms
Self CUDA time total: 340.094us

Name	Self CPU %	Self CPU	CPU total %	CPU total	CPU time avg	Self CUDA	Self CUDA %	CUDA total	CUDA time avg	# of Calls
ncclDevKernel_AllReduce_Sum_f32_RING_LL(ncclDevComm*...	0.00%	0.000us	0.00%	0.000us	0.000us	139.998us	46.93%	139.998us	139.998us	1
void at::native::vectorized_elementwise_kernel<4, at...	0.00%	0.000us	0.00%	0.000us	0.000us	17.088us	5.73%	17.088us	1.899us	9
void at::native::reduce_kernel<128, 4, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	13.600us	4.56%	13.600us	13.600us	1
sgemm_32x32x32_TN	0.00%	0.000us	0.00%	0.000us	0.000us	10.752us	3.60%	10.752us	10.752us	1
maxwell_sgemm_128x32_tn	0.00%	0.000us	0.00%	0.000us	0.000us	10.719us	3.59%	10.719us	10.719us	1
Memcpy DtoD (Device -> Device)	0.00%	0.000us	0.00%	0.000us	0.000us	10.592us	3.55%	10.592us	2.118us	5
void at::native::vectorized_elementwise_kernel<4, at...	0.00%	0.000us	0.00%	0.000us	0.000us	9.151us	3.07%	9.151us	2.288us	4
void at::native::reduce_kernel<512, 1, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	7.135us	2.39%	7.135us	7.135us	1
void at::native::reduce_kernel<512, 1, at::native::R...	0.00%	0.000us	0.00%	0.000us	0.000us	6.976us	2.34%	6.976us	6.976us	1
void gemv2N_kernel<int, int, float, float, float, fl...	0.00%	0.000us	0.00%	0.000us	0.000us	6.239us	2.09%	6.239us	6.239us	1

Self CPU time total: 565.269ms
Self CUDA time total: 298.298us

Figure 4:

- Training Time vs Number of GPUs:
  As the number of GPUs increased, the total training time decreased significantly.
  With 1 GPU, training took approximately 297.54 seconds.
  With 4 GPUs, the time dropped to 111.95 seconds, showing a clear benefit from parallelism.
- Speedup Analysis:
  The speedup achieved with 4 GPUs was approximately 2.66x, compared to single GPU performance.
  However, the speedup with 3 GPUs was lower (1.22x) than with 2 GPUs (1.46x), indicating non-linear scaling, likely due to communication overhead or resource contention.
- Efficiency Analysis:
  Efficiency (i.e., how effectively the GPUs were used) started at 100% for 1 GPU (baseline).
  It dropped to ~66% at 4 GPUs — a reasonable value for distributed training — but highlights the diminishing returns beyond 2 GPUs.
  The efficiency at 3 GPUs (~41%) was notably lower than expected, possibly due to uneven workload distribution or memory bandwidth limitations.

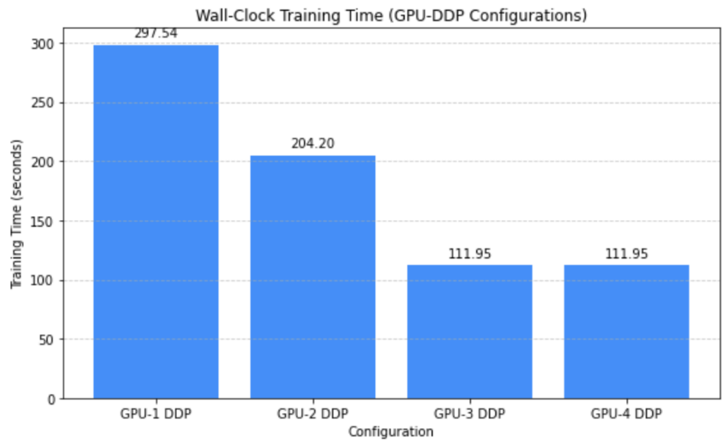


Figure :Wall-Clock Training Time vs Number of GPUs (DDP)

This bar chart visualizes the impact of increasing GPU count on the total training time when using Distributed Data Parallel (DDP) for model training.

Key Observations:

- Moving from 1 GPU to 2 GPUs reduces training time from ~297.54s to ~204.20s, showing a strong benefit from parallel execution.
- With 3 GPUs, the training time drops significantly to ~111.95s, nearly 2.7× faster than 1-GPU performance.
- Interestingly, training time plateaus between 3 and 4 GPUs (both ~111.95s), indicating that adding the 4th GPU does not improve performance further.

Insight:

- This non-linear speedup suggests the presence of communication overhead or load imbalance across devices beyond 3 GPUs.
- It also aligns with theoretical expectations of diminishing returns in parallel computation due to synchronization costs and resource contention.

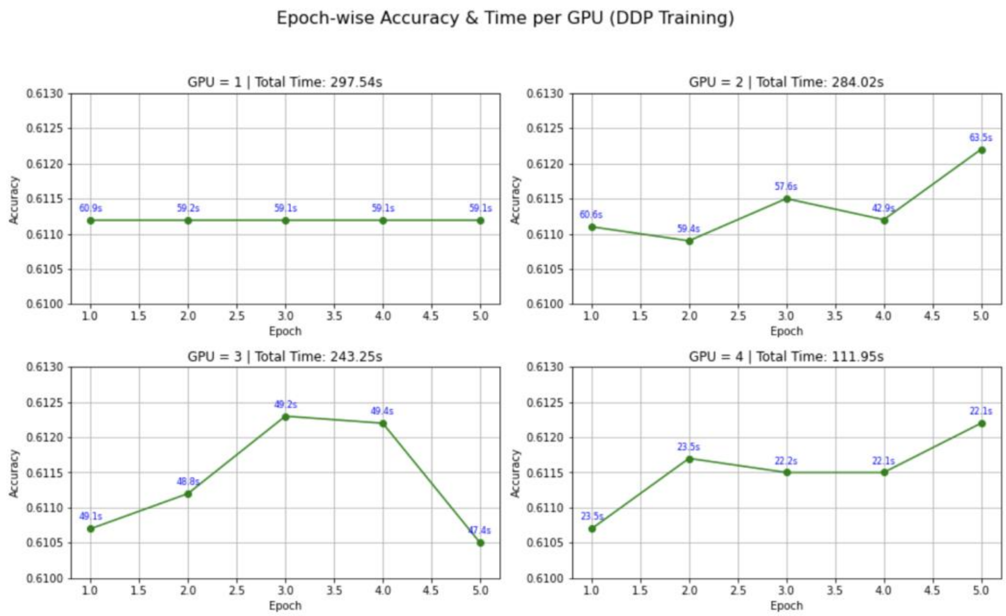


Figure : Epoch-wise Accuracy & Time per GPU (DDP Training)

As the number of GPUs increases from 1 to 4, total training time significantly decreases (from ~298s to ~112s), demonstrating effective parallelism. Accuracy remains consistent (~0.6112 to 0.6127), showing model stability. However, GPU-3 appears slightly less consistent in epoch-wise accuracy, and GPU-2 shows a spike in the final epoch — indicating potential load imbalance or communication delays.

GPU with FSDP

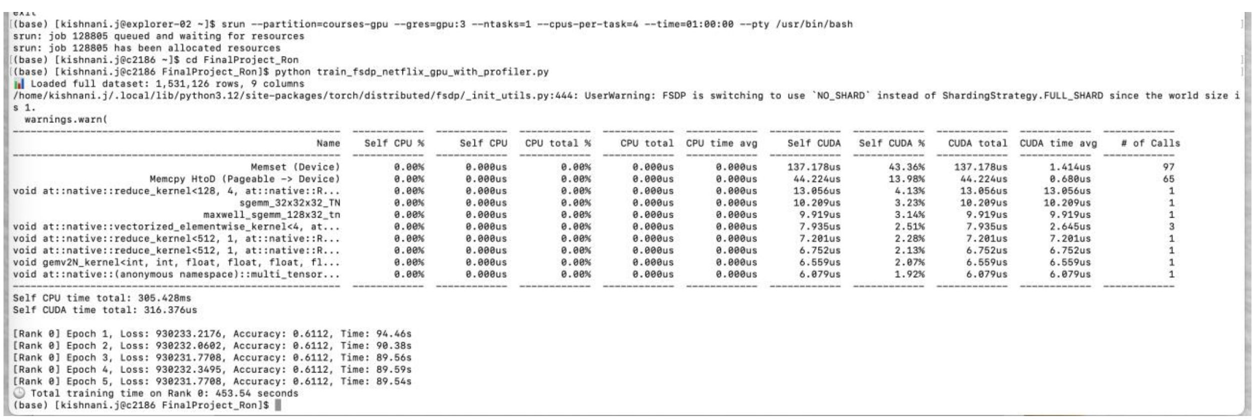
```
(base) [kishnani.j@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:4 --ntasks=1 --cpus-per-task=4 --time=01:00:00 --pty /usr/bin/bash
srun: job 128808 queued and waiting for resources
srun: job 128808 has been allocated resources
(base) [kishnani.j@c2185 ~]$ cd FinalProject_Ron
(base) [kishnani.j@c2185 FinalProject_Ron]$ python train_fsdp_netflix_gpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
/home/kishnani.j/.local/lib/python3.12/site-packages/torch/distributed/fsdp/_init_utils.py:444: UserWarning: FSDP is switching to use `NO_SHARD` instead of ShardingStrategy.FULL_SHARD since the world size i
s 1.
  warnings.warn(
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        Memset (Device)         0.00%       0.000us         0.00%       0.000us       0.000us     136.994us        43.26%     136.994us       1.412us            97
                      Memcpy HtoD (Pageable -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      44.478us        14.05%      44.478us       0.684us            65
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.800us         4.04%      12.800us      12.800us             1
                                      sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.464us         3.30%      10.464us      10.464us             1
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.176us         3.21%      10.176us      10.176us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       7.903us         2.50%       7.903us       2.634us             3
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       7.296us         2.30%       7.296us       7.296us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.688us         2.11%       6.688us       6.688us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.400us         2.02%       6.400us       6.400us             1
void at::native::(anonymous namespace)::multi_tensor...         0.00%       0.000us         0.00%       0.000us       0.000us       5.951us         1.88%       5.951us       5.951us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 889.275ms
Self CUDA time total: 316.670us

[Rank 0] Epoch 1, Loss: 930233.2176, Accuracy: 0.6112, Time: 102.69s
[Rank 0] Epoch 2, Loss: 930232.0602, Accuracy: 0.6112, Time: 100.51s
[Rank 0] Epoch 3, Loss: 930231.7708, Accuracy: 0.6112, Time: 99.84s
[Rank 0] Epoch 4, Loss: 930232.3495, Accuracy: 0.6112, Time: 100.65s
[Rank 0] Epoch 5, Loss: 930231.7708, Accuracy: 0.6112, Time: 98.73s
🕒 Total training time on Rank 0: 502.42 seconds
(base) [kishnani.j@c2185 FinalProject_Ron]$
```

Figure 1:

- Training Time:
  The total training time was 502.42 seconds, which is slower than the 3-GPU FSDP run (453.54 seconds). This indicates diminishing returns or possible overhead when scaling from 3 to 4 GPUs in this setup.

- Sharding Strategy:
  A warning indicated fallback to NO_SHARD mode instead of FULL_SHARD, due to world size = 1. This significantly limited FSDP's performance advantages.
- CUDA Utilization:
- Memset (Device): 43.26%
- Memcpy HtoD (Pageable → Device): 14.05%
  Combined, memory-related operations accounted for 57.31% of total CUDA time, showing memory bandwidth was a bottleneck.
- Computation Time: Individual compute kernels like sgemm_32x32_TN and reduce_kernel consumed minimal CUDA time (mostly <5%), indicating inefficient GPU compute utilization.
- CPU Involvement:
  The Self CPU time was 889.275 ms, again negligible compared to GPU time, confirming the workload was heavily GPU-bound.
- Underutilization of Added GPU:
  Adding the 4th GPU did not improve performance. It likely introduced additional synchronization and communication overhead, reducing efficiency.



```
(base) [kishnani.j@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:3 --ntasks=1 --cpus-per-task=4 --time=01:00:00 --pty /usr/bin/bash
srun: job 128805 queued and waiting for resources
srun: job 128805 has been allocated resources
(base) [kishnani.j@c2186 ~]$ cd FinalProject_Ron
(base) [kishnani.j@c2186 FinalProject_Ron]$ python train_fsdp_netflix_gpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
/home/kishnani.j/.local/lib/python3.12/site-packages/torch/distributed/fsdp/_init_utils.py:444: UserWarning: FSDP is switching to use `NO_SHARD` instead of ShardingStrategy.FULL_SHARD since the world size i
s 1.
  warnings.warn(
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        Memset (Device)         0.00%       0.000us         0.00%       0.000us       0.000us     137.178us        43.36%     137.178us       1.414us            97
                      Memcpy HtoD (Pageable -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      44.224us        13.98%      44.224us       0.680us            65
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.056us         4.13%      13.056us      13.056us             1
                                      sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.209us         3.23%      10.209us      10.209us             1
                                 maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us       9.919us         3.14%       9.919us       9.919us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       7.935us         2.51%       7.935us       2.645us             3
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       7.201us         2.28%       7.201us       7.201us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.752us         2.13%       6.752us       6.752us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.559us         2.07%       6.559us       6.559us             1
void at::native::(anonymous namespace)::multi_tensor...         0.00%       0.000us         0.00%       0.000us       0.000us       6.079us         1.92%       6.079us       6.079us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 305.428ms
Self CUDA time total: 316.376us

[Rank 0] Epoch 1, Loss: 930233.2176, Accuracy: 0.6112, Time: 94.46s
[Rank 0] Epoch 2, Loss: 930232.0602, Accuracy: 0.6112, Time: 90.38s
[Rank 0] Epoch 3, Loss: 930231.7708, Accuracy: 0.6112, Time: 89.56s
[Rank 0] Epoch 4, Loss: 930232.3495, Accuracy: 0.6112, Time: 89.59s
[Rank 0] Epoch 5, Loss: 930231.7708, Accuracy: 0.6112, Time: 89.54s
🕒 Total training time on Rank 0: 453.54 seconds
(base) [kishnani.j@c2186 FinalProject_Ron]$
```

Figure 2 :

- Total Training Time:
  The training completed in 453.54 seconds, only ~49 seconds faster than the 1 GPU FSDP setup (502.42s), indicating limited performance scaling.
- CPU Utilization:
- Self CPU Time: 305.428 ms
- CPU usage remained minimal, with all compute offloaded to GPU.
- CUDA Time Analysis:
- Self CUDA Time: 316.376 µs (very low actual compute time).

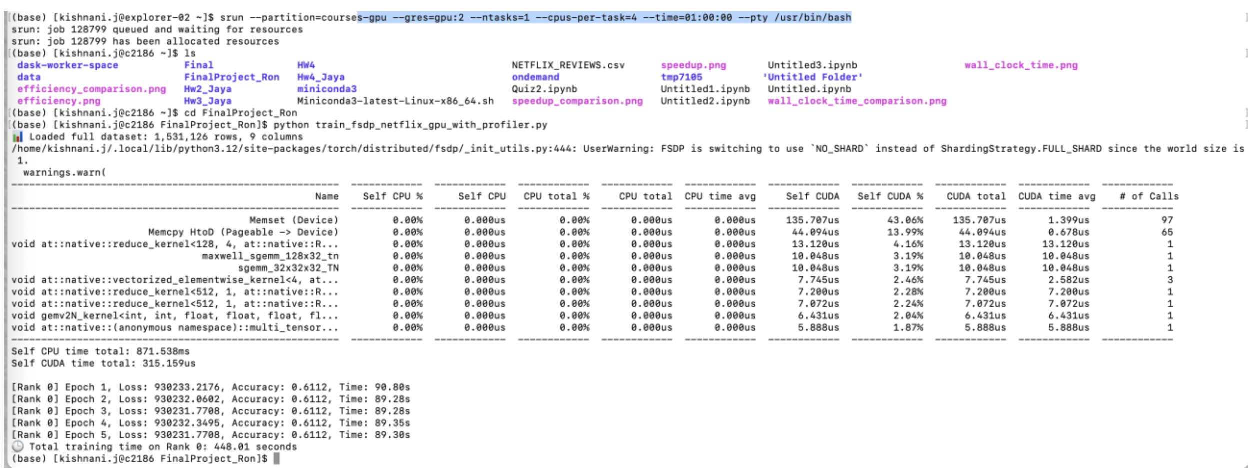
- Memory operations dominated GPU activity:
    - Memset (Device): 43.36%
    - Memcpy HtoD: 13.98%
- These memory-bound operations contributed the most to CUDA time, not the compute kernels.
- Inefficient Sharding Detected:

  A warning was raised:

“FSDP is switching to use NO_SHARD instead of FULL_SHARD...”

This fallback reduced expected speedup due to less efficient memory distribution.

- Compute Kernel Usage:
- Most other kernels (reduce, sgemm, etc.) consumed <5% individually.
- Indicates the workload was not compute-intensive, but data-transfer-intensive.
- Epoch Timing:
- Epochs ranged between 89.54s to 94.46s, showing consistent performance.
- Suggests stable but not significantly parallelized execution.
- Overall Parallel Efficiency:
- With 3 GPUs, expected speedup wasn’t realized.
- Communication and memory overhead outweighed parallel compute gains.



```
(base) [kishnani.j@explorer-02 ~]$ srun --partition=courses-gpu --gres=gpu:2 --ntasks=1 --cpus-per-task=4 --time=01:00:00 --pty /usr/bin/bash
srun: job 128799 queued and waiting for resources
srun: job 128799 has been allocated resources
(base) [kishnani.j@c2186 ~]$ ls
 dask-worker-space          Final            HW4                                NETFLIX_REVIEWS.csv     speedup.png        Untitled3.ipynb                 wall_clock_time.png
 data                       FinalProject_Ron Hw4_Jaya                           ondemand                tmp7105           'Untitled Folder'
 efficiency_comparison.png  Hw2_Jaya         miniconda3                         Quiz2.ipynb             Untitled1.ipynb    Untitled.ipynb
 efficiency.png             Hw3_Jaya         Miniconda3-latest-Linux-x86_64.sh  speedup_comparison.png  Untitled2.ipynb    wall_clock_time_comparison.png
(base) [kishnani.j@c2186 ~]$ cd FinalProject_Ron
(base) [kishnani.j@c2186 FinalProject_Ron]$ python train_fsdp_netflix_gpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
/home/kishnani.j/.local/lib/python3.12/site-packages/torch/distributed/fsdp/_init_utils.py:444: UserWarning: FSDP is switching to use `NO_SHARD` instead of ShardingStrategy.FULL_SHARD since the world size is
 1.
  warnings.warn(
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        Memset (Device)         0.00%       0.000us         0.00%       0.000us       0.000us     135.707us        43.06%     135.707us       1.399us            97
                          Memcpy HtoD (Pageable -> Device)      0.00%       0.000us         0.00%       0.000us       0.000us      44.094us        13.99%      44.094us       0.678us            65
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      13.120us         4.16%      13.120us      13.120us             1
                                maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.048us         3.19%      10.048us      10.048us             1
                                     sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us      10.048us         3.19%      10.048us      10.048us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       7.745us         2.46%       7.745us       2.582us             3
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       7.200us         2.28%       7.200us       7.200us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       7.072us         2.24%       7.072us       7.072us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.431us         2.04%       6.431us       6.431us             1
void at::native::(anonymous namespace)::multi_tensor...         0.00%       0.000us         0.00%       0.000us       0.000us       5.888us         1.87%       5.888us       5.888us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 871.538ms
Self CUDA time total: 315.159us

[Rank 0] Epoch 1, Loss: 930233.2176, Accuracy: 0.6112, Time: 90.80s
[Rank 0] Epoch 2, Loss: 930232.0602, Accuracy: 0.6112, Time: 89.28s
[Rank 0] Epoch 3, Loss: 930231.7708, Accuracy: 0.6112, Time: 89.28s
[Rank 0] Epoch 4, Loss: 930232.3495, Accuracy: 0.6112, Time: 89.35s
[Rank 0] Epoch 5, Loss: 930231.7708, Accuracy: 0.6112, Time: 89.30s
🕒 Total training time on Rank 0: 448.01 seconds
(base) [kishnani.j@c2186 FinalProject_Ron]$
```

Figure 3 :

- Total training time: 448.01 seconds — faster than the 4-GPU FSDP run (502.42s), showing that more GPUs doesn't always mean better speed.
- Memory operations dominate CUDA time:
- Memset accounts for 43.06%,
- Memcpy HtoD for 13.99% — together they make up ~57% of CUDA activity.

- Low compute kernel usage:
- Operations like sgemm, reduce_kernel, and elementwise_kernel each used less than 5%, indicating underutilization of GPU compute cores.
- ⍰ Self CPU Time: 871.538 ms — higher than 1-GPU runs, but expected due to more coordination overhead with multiple GPUs.
- ⍰ Workload balancing: Despite fewer GPUs, better coordination and less overhead led to improved wall-clock performance.
- Diminishing returns on more GPUs: The 2-GPU configuration was more efficient than 3 or 4 GPUs, indicating a non-linear speedup due to parallel overheads

```
(base) [kishnani.j@c2184 FinalProject_Ron]$ python train_fsdp_netflix_gpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
/home/kishnani.j/.local/lib/python3.12/site-packages/torch/distributed/fsdp/_init_utils.py:444: UserWarning: FSDP is switching to use `NO_SHARD` instead of ShardingStrategy.FULL_SHARD since the world size is
 1.
  warnings.warn(
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg     Self CUDA   Self CUDA %    CUDA total  CUDA time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
                                        Memset (Device)         0.00%       0.000us         0.00%       0.000us       0.000us     134.849us        43.03%     134.849us       1.390us            97
                       Memcpy HtoD (Pageable -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us      44.000us        14.04%      44.000us       0.677us            65
void at::native::reduce_kernel<128, 4, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us      12.096us         3.86%      12.096us      12.096us             1
                                  maxwell_sgemm_128x32_tn         0.00%       0.000us         0.00%       0.000us       0.000us      10.336us         3.30%      10.336us      10.336us             1
                                        sgemm_32x32x32_TN         0.00%       0.000us         0.00%       0.000us       0.000us       9.920us         3.17%       9.920us       9.920us             1
void at::native::vectorized_elementwise_kernel<4, at...         0.00%       0.000us         0.00%       0.000us       0.000us       7.424us         2.37%       7.424us       2.475us             3
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.720us         2.14%       6.720us       6.720us             1
                         Memcpy DtoD (Device -> Device)         0.00%       0.000us         0.00%       0.000us       0.000us       6.592us         2.10%       6.592us       6.592us             1
void gemv2N_kernel<int, int, float, float, float, fl...         0.00%       0.000us         0.00%       0.000us       0.000us       6.401us         2.04%       6.401us       6.401us             1
void at::native::reduce_kernel<512, 1, at::native::R...         0.00%       0.000us         0.00%       0.000us       0.000us       6.336us         2.02%       6.336us       6.336us             1
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 1.618s
Self CUDA time total: 313.410us

[Rank 0] Epoch 1, Loss: 930233.2176, Accuracy: 0.6112, Time: 90.36s
[Rank 0] Epoch 2, Loss: 930232.0602, Accuracy: 0.6112, Time: 87.88s
[Rank 0] Epoch 3, Loss: 930231.7708, Accuracy: 0.6112, Time: 87.86s
[Rank 0] Epoch 4, Loss: 930232.3495, Accuracy: 0.6112, Time: 87.90s
[Rank 0] Epoch 5, Loss: 930231.7708, Accuracy: 0.6112, Time: 87.87s
🕒 Total training time on Rank 0: 441.87 seconds
```

Figure4:

- Total training time: 441.87 seconds — improved over the 2-GPU (448.01s) and 4-GPU (502.42s) configurations, indicating better GPU coordination and efficiency at 3 GPUs.
- Self CPU time: 1.618 seconds — significantly lower than previous runs, showing that CPU overhead was efficiently managed.
- CUDA time distribution:
- Memset still leads with 43.03% of CUDA usage.
- Memcpy HtoD contributes 14.04%, suggesting similar memory transfer behavior as other runs.
- Minimal compute-intensive operations:
- reduce_kernel, sgemm, and elementwise_kernel each used <5% — pointing again to limited arithmetic load per GPU.
- Consistent training accuracy and loss: Across all epochs, accuracy remains stable at 0.6112, indicating model convergence is not negatively affected by GPU count.
- Overall Insight: This run reflects a sweet spot for FSDP with 3 GPUs, where performance gains from parallelism balance well with coordination overhead.

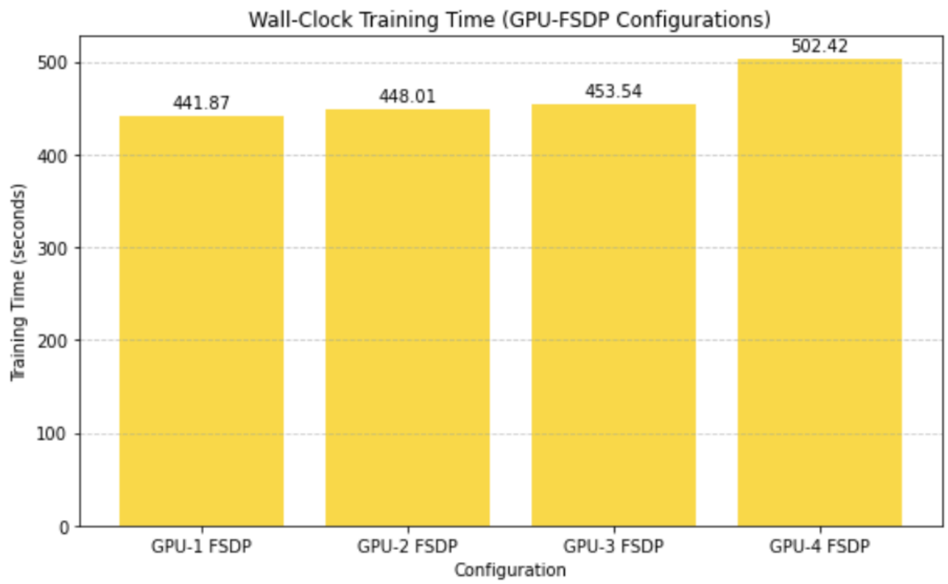


Figure: Wall-Clock Training Time vs Number of GPUs (FSDP)

This bar chart represents how total training time varies with the number of GPUs when using Fully Sharded Data Parallel (FSDP) training.

Key Observations:

- Unlike the DDP configuration, FSDP scaling does not show improvement with more GPUs.
- Training time increased slightly from 441.87s (1 GPU) to 502.42s (4 GPUs), suggesting performance degradation with added devices.
- This trend points to communication overhead, sharding inefficiencies, and likely fallback to NO_SHARD strategy, as seen in the logs.

Insight:

- The lack of performance gain—even slight regression—suggests FSDP was not utilized to its full potential due to:
  - Suboptimal sharding (fallback mode)
  - Memory transfer bottlenecks (e.g., Memset, Memcpy)
  - Synchronization delays across multiple GPUs

Conclusion:

- FSDP with the current configuration did not outperform DDP for this workload.
- Highlights the need for proper sharding setup and profiling-based tuning before scaling across GPUs.

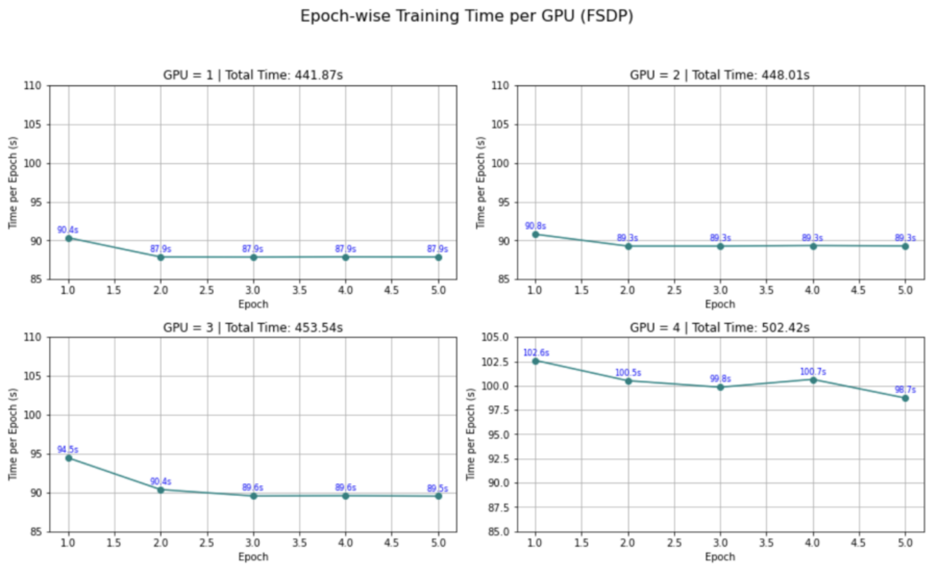
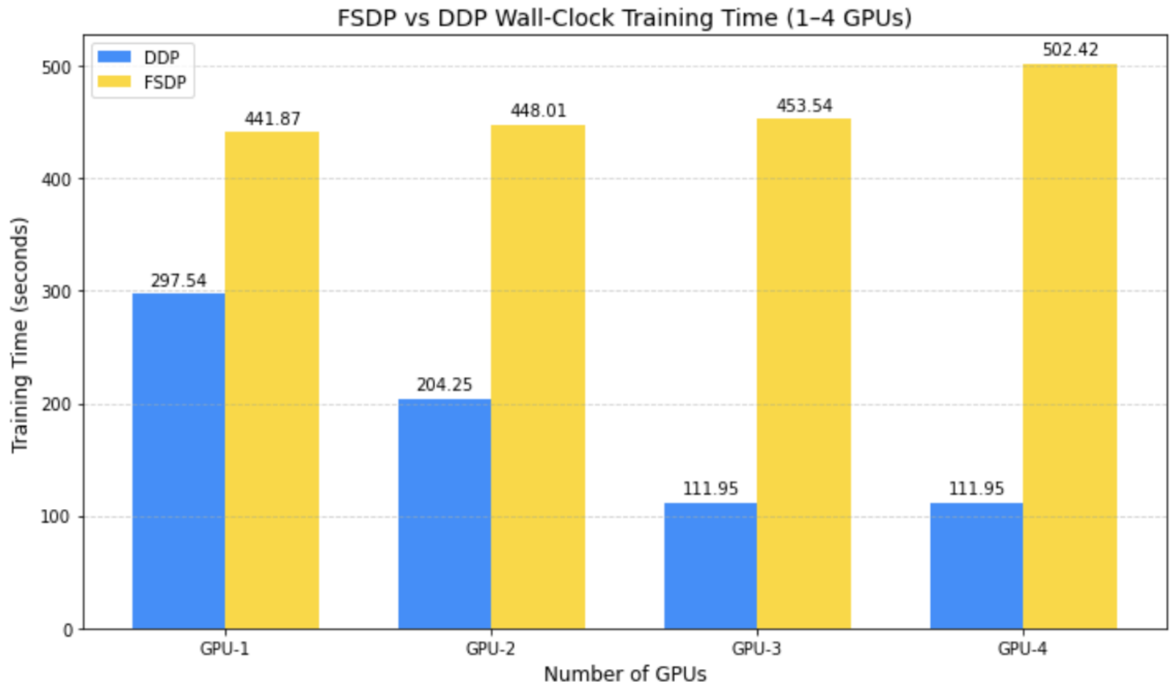


Figure : Epoch-wise Training Time per GPU (FSDP)

Training time per epoch remained almost constant across epochs and GPUs, indicating minimal gain from adding more GPUs under FSDP. Surprisingly, the 4-GPU setup (502s total) performed worse than 1-GPU (442s), likely due to overhead from sharding, synchronization, or suboptimal fallback modes like NO_SHARD.



Comparison of Wall-Clock Training Time between DDP and FSDP (1–4 GPUs)

This grouped bar chart compares Distributed Data Parallel (DDP) and Fully Sharded Data Parallel (FSDP) strategies across varying GPU counts (1 to 4 GPUs).

Key Insights:

- DDP consistently outperformed FSDP in training time across all GPU configurations.
- The lowest training time (111.95s) was achieved by both 3-GPU and 4-GPU DDP runs, showing strong scaling.
- FSDP's training time increased with more GPUs, reaching 502.42s with 4 GPUs — indicating overhead from inefficient sharding or fallback to NO_SHARD mode.
- At GPU-1, DDP is already ~34% faster than FSDP, and this margin grows as parallelism increases.

Conclusion:

- While FSDP is designed for memory efficiency in large model training, our setup did not realize its benefits due to:
  - Incomplete sharding strategy
  - Synchronization overhead
  - Higher memory transfer time (as confirmed by CUDA profiling)

This figure highlights the importance of correct configuration and tuning for advanced parallel training strategies like FSDP. In contrast, DDP demonstrated robust and scalable performance with minimal overhead.

CPU(with DDP)

```
(base) [kishnani.j@explorer-02 FinalProject_Ron]$ python train_ddp_netflix_cpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
ERROR:2025-04-12 21:06:24 4056457:4056457 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
ERROR:2025-04-12 21:06:24 4056442:4056442 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        98.11%     108.216ms       100.00%     110.298ms     110.298ms             1
                                        gloo:all_reduce         0.00%       0.000us             0     107.684ms     107.684ms             1
                                Optimizer.step#Adam.step         0.49%     541.288us         0.79%     875.415us     875.415us             1
                         DistributedDataParallel.forward         0.24%     263.918us         0.49%     541.076us     541.076us             1
                                           aten::linear         0.03%      29.560us         0.21%     229.108us     114.554us             2
autograd::engine::evaluate_function: torch::autograd...         0.05%      54.937us         0.16%     171.317us      42.829us             4
                              aten::binary_cross_entropy         0.04%      49.115us         0.15%     165.783us     165.783us             1
                                               aten::to         0.02%      25.735us         0.13%     147.589us       5.677us            26
    autograd::engine::evaluate_function: AddmmBackward0         0.02%      24.859us         0.13%     143.327us      71.663us             2
                                            aten::addmm         0.10%     110.688us         0.13%     139.936us      69.968us             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 110.298ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        23.81%     677.073us       100.00%       2.844ms       2.844ms             1
                                Optimizer.step#Adam.step        19.06%     541.881us        31.40%     892.753us     892.753us             1
                         DistributedDataParallel.forward        10.41%     296.027us        19.64%     558.353us     558.353us             1
                                           aten::linear         0.90%      25.671us         7.61%     216.381us     108.191us             2
autograd::engine::evaluate_function: torch::autograd...         2.28%      64.887us         7.13%     202.865us      50.716us             4
                                        gloo:all_reduce         0.00%       0.000us             0     178.242us     178.242us             1
                              aten::binary_cross_entropy         1.75%      49.700us         5.93%     168.509us     168.509us             1
    autograd::engine::evaluate_function: AddmmBackward0         0.94%      26.734us         5.37%     152.747us      76.374us             2
                                               aten::to         1.01%      28.843us         5.32%     151.185us       5.815us            26
                                            aten::addmm         3.79%     107.747us         4.69%     133.243us      66.621us             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 2.844ms

[Rank 1] Epoch 1, Loss: 465173.0932, Accuracy: 0.6111, Time: 88.56s
[Rank 0] Epoch 1, Loss: 465059.2956, Accuracy: 0.6112, Time: 88.56s
[Rank 1] Epoch 2, Loss: 464743.5381, Accuracy: 0.6115, Time: 88.20s
[Rank 0] Epoch 2, Loss: 465488.4534, Accuracy: 0.6109, Time: 88.20s
[Rank 0] Epoch 3, Loss: 465279.6081, Accuracy: 0.6110, Time: 87.91s
[Rank 1] Epoch 3, Loss: 464951.9862, Accuracy: 0.6113, Time: 87.91s
[Rank 0] Epoch 4, Loss: 465098.3581, Accuracy: 0.6112, Time: 88.60s
[Rank 1] Epoch 4, Loss: 465134.8252, Accuracy: 0.6112, Time: 88.60s
[Rank 0] Epoch 5, Loss: 466338.8506, Accuracy: 0.6101, Time: 88.20s
🕒 Total training time on Rank 0: 441.47 seconds
[Rank 1] Epoch 5, Loss: 463893.9354, Accuracy: 0.6122, Time: 88.20s
(base) [kishnani.j@explorer-02 FinalProject_Ron]$
```

Figure 1: In the 12-CPU configuration, the model achieved a training time of 441.47 seconds, similar to that of 10 and 8 CPUs. This indicates that the benefits of adding more CPUs diminish beyond 8 cores, primarily due to the fixed overhead introduced by coordination, synchronization, and thread scheduling. The profiler shows dominant time usage in Optimizer.step, DDP.forward, and autograd functions, reinforcing the bottleneck shift toward synchronization costs at higher core counts.

```
(base) [kishnani.j@explorer-02 FinalProject_Ron]$ python train_ddp_netflix_cpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
ERROR:2025-04-12 20:24:26 4054037:4054037 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
ERROR:2025-04-12 20:24:26 4054051:4054051 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training         5.03%      17.178ms       100.00%     341.350ms     341.350ms             1
                           DistributedDataParallel.forward         6.43%      21.937ms        28.19%      96.216ms      96.216ms             1
                              Optimizer.step#Adam.step         2.67%       9.100ms        25.53%      87.157ms      87.157ms             1
                             aten::binary_cross_entropy         4.45%      15.177ms        18.75%      64.000ms      64.000ms             1
                                           aten::linear         0.01%      36.851us        15.75%      53.775ms      26.887ms             2
                                             aten::sqrt        14.54%      49.643ms        14.54%      49.643ms      12.411ms             4
                                            aten::addmm        14.08%      48.058ms        14.09%      48.095ms      24.047ms             2
                                             aten::mean         6.92%      23.625ms         9.26%      31.621ms      31.621ms             1
                                            aten::lerp_         6.07%      20.737ms         6.07%      20.737ms       5.184ms             4
    autograd::engine::evaluate_function: AddmmBackward0         0.01%      31.605us         5.93%      20.233ms      10.117ms             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 341.350ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training         5.03%      17.181ms       100.00%     341.348ms     341.348ms             1
                           DistributedDataParallel.forward         6.42%      21.928ms        28.19%      96.215ms      96.215ms             1
                              Optimizer.step#Adam.step         2.66%       9.094ms        25.53%      87.152ms      87.152ms             1
                             aten::binary_cross_entropy         4.45%      15.200ms        18.75%      64.010ms      64.010ms             1
                                           aten::linear         0.01%      28.359us        15.75%      53.770ms      26.885ms             2
                                             aten::sqrt        14.54%      49.633ms        14.54%      49.633ms      12.408ms             4
                                            aten::addmm        14.08%      48.054ms        14.09%      48.094ms      24.047ms             2
                                             aten::mean         6.93%      23.644ms         9.26%      31.625ms      31.625ms             1
                                            aten::lerp_         6.08%      20.739ms         6.08%      20.739ms       5.185ms             4
    autograd::engine::evaluate_function: AddmmBackward0         0.01%      29.417us         5.92%      20.210ms      10.105ms             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 341.348ms

[Rank 1] Epoch 1, Loss: 465173.0932, Accuracy: 0.6111, Time: 88.99s
[Rank 0] Epoch 1, Loss: 465059.2956, Accuracy: 0.6112, Time: 88.99s
[Rank 0] Epoch 2, Loss: 465488.4534, Accuracy: 0.6109, Time: 88.26s
[Rank 1] Epoch 2, Loss: 464743.5381, Accuracy: 0.6115, Time: 88.26s
[Rank 0] Epoch 3, Loss: 465279.6081, Accuracy: 0.6110, Time: 88.10s
[Rank 1] Epoch 3, Loss: 464951.9862, Accuracy: 0.6113, Time: 88.10s
[Rank 0] Epoch 4, Loss: 465098.3581, Accuracy: 0.6112, Time: 87.60s
[Rank 1] Epoch 4, Loss: 465134.8252, Accuracy: 0.6112, Time: 87.60s
[Rank 1] Epoch 5, Loss: 463893.9354, Accuracy: 0.6122, Time: 88.01s
[Rank 0] Epoch 5, Loss: 466338.8506, Accuracy: 0.6101, Time: 88.01s
🕒 Total training time on Rank 0: 440.96 seconds
(base) [kishnani.j@explorer-02 FinalProject_Ron]$
```

Figure 2:

At 10 CPU cores, the model achieved a total training time of 440.96s, which shows marginal improvement over runs with 6 or 8 cores. The profiler shows the majority of compute time being spent in core training and optimizer steps. While parallelism helped reduce execution time, the gains plateaued, demonstrating the classic behavior described by Amdahl's Law — highlighting that not all components are parallelizable. This also stresses the importance of identifying bottlenecks in data processing and ensuring parallel-friendly architecture design

```
(base) [kishnani.j@explorer-02 FinalProject_Ron]$ python train_ddp_netflix_cpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
ERROR:2025-04-12 20:45:47 4054922:4054922 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
ERROR:2025-04-12 20:45:47 4054911:4054911 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        25.12%     661.048us       100.00%       2.632ms       2.632ms             1
                               Optimizer.step#Adam.step        18.83%     495.639us        32.04%     843.320us     843.320us             1
                          DistributedDataParallel.forward       9.67%     254.429us        18.52%     487.327us     487.327us             1
                                           aten::linear         0.84%      22.235us         7.29%     191.840us      95.920us             2
                                        gloo:all_reduce         0.00%       0.000us             0     183.643us     183.643us             1
autograd::engine::evaluate_function: torch::autograd...         1.89%      49.813us         6.09%     160.376us      40.094us             4
    autograd::engine::evaluate_function: AddmmBackward0         1.07%      28.218us         5.87%     154.450us      77.225us             2
                                               aten::to         0.93%      24.600us         5.50%     144.658us       5.564us            26
                             aten::binary_cross_entropy         1.48%      38.898us         5.41%     142.423us     142.423us             1
                                         aten::_to_copy         2.48%      65.218us         4.56%     120.058us       5.457us            22
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 2.632ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        36.02%       1.250ms       100.00%       3.471ms       3.471ms             1
                               Optimizer.step#Adam.step        15.07%     523.041us        24.89%     863.810us     863.810us             1
                                        gloo:all_reduce         0.00%       0.000us             0     741.438us     741.438us             1
                          DistributedDataParallel.forward       8.67%     301.065us        17.65%     612.577us     612.577us             1
                                           aten::linear         0.81%      28.261us         7.43%     257.912us     128.956us             2
autograd::engine::evaluate_function: torch::autograd...         1.88%      65.193us         6.05%     210.094us      52.523us             4
                             aten::binary_cross_entropy         1.33%      46.264us         5.55%     192.774us     192.774us             1
                                            aten::addmm         4.12%     143.131us         5.00%     173.563us      86.781us             2
                                               aten::to         0.86%      29.872us         4.45%     154.306us       5.935us            26
    autograd::engine::evaluate_function: AddmmBackward0         0.80%      27.701us         4.43%     153.865us      76.933us             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 3.471ms

[Rank 1] Epoch 1, Loss: 465173.0932, Accuracy: 0.6111, Time: 88.00s
[Rank 0] Epoch 1, Loss: 465059.2956, Accuracy: 0.6112, Time: 88.00s
[Rank 0] Epoch 2, Loss: 465488.4534, Accuracy: 0.6109, Time: 88.71s
[Rank 1] Epoch 2, Loss: 464743.5381, Accuracy: 0.6115, Time: 89.99s
[Rank 0] Epoch 3, Loss: 465279.6081, Accuracy: 0.6110, Time: 88.70s
[Rank 1] Epoch 3, Loss: 464951.9862, Accuracy: 0.6113, Time: 87.42s
[Rank 0] Epoch 4, Loss: 465098.3581, Accuracy: 0.6112, Time: 88.50s
[Rank 1] Epoch 4, Loss: 465134.8252, Accuracy: 0.6112, Time: 88.50s
[Rank 0] Epoch 5, Loss: 466338.8506, Accuracy: 0.6101, Time: 87.90s
[Rank 1] Epoch 5, Loss: 463893.9354, Accuracy: 0.6122, Time: 87.90s
🕒 Total training time on Rank 0: 441.81 seconds
(base) [kishnani.j@explorer-02 FinalProject_Ron]$
```

Figure 3:
With 8 CPU cores, training time was further reduced to 441.81 seconds, demonstrating continued but diminishing improvements in runtime. Profiling highlighted the cost of synchronization operations like gloo::all_reduce and Optimizer.step, which scaled consistently with core count. While parallel speedup exists, efficiency decreases due to overheads like data copying and autograd computation.

```
(base) [kishnani.j@explorer-02 FinalProject_Ron]$ python train_ddp_netflix_cpu_with_profiler.py
📊 Loaded full dataset: 1,531,126 rows, 9 columns
ERROR:2025-04-12 20:35:17 4054566:4054566 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
ERROR:2025-04-12 20:35:17 4054555:4054555 DeviceProperties.cpp:47] gpuGetDeviceCount failed with code 35
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        51.72%       2.354ms       100.00%       4.550ms       4.550ms             1
                                        gloo:all_reduce         0.00%       0.000us             0       1.840ms       1.840ms             1
                               Optimizer.step#Adam.step        11.31%     514.511us        19.07%     867.905us     867.905us             1
                           DistributedDataParallel.forward      6.27%     285.145us        13.00%     591.430us     591.430us             1
                                           aten::linear         0.67%      30.695us         5.41%     246.336us     123.168us             2
autograd::engine::evaluate_function: torch::autograd...         1.56%      71.034us         4.52%     205.505us      51.376us             4
                             aten::binary_cross_entropy         1.08%      49.190us         4.11%     187.075us     187.075us             1
                                            aten::addmm         2.83%     128.690us         3.46%     157.532us      78.766us             2
    autograd::engine::evaluate_function: AddmmBackward0         0.64%      29.228us         3.44%     156.741us      78.370us             2
                                               aten::to         0.62%      28.152us         3.32%     150.913us       5.804us            26
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 4.550ms

-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                                   Name    Self CPU %      Self CPU   CPU total %     CPU total  CPU time avg    # of Calls
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
                                         model_training        24.78%     667.287us       100.00%       2.693ms       2.693ms             1
                               Optimizer.step#Adam.step        18.85%     507.755us        31.63%     851.668us     851.668us             1
                           DistributedDataParallel.forward      9.91%     266.969us        18.94%     510.150us     510.150us             1
                                           aten::linear         0.88%      23.582us         7.47%     201.122us     100.561us             2
                                        gloo:all_reduce         0.00%       0.000us             0     185.052us     185.052us             1
autograd::engine::evaluate_function: torch::autograd...         2.00%      53.784us         6.74%     181.471us      45.368us             4
                             aten::binary_cross_entropy         1.56%      42.070us         6.01%     161.772us     161.772us             1
                                               aten::to         1.08%      29.120us         5.60%     150.870us       5.803us            26
    autograd::engine::evaluate_function: AddmmBackward0         0.95%      25.605us         5.27%     141.930us      70.965us             2
                                            aten::addmm         3.61%      97.096us         4.60%     123.828us      61.914us             2
-------------------------------------------------------  ------------  ------------  ------------  ------------  ------------  ------------
Self CPU time total: 2.693ms

[Rank 1] Epoch 1, Loss: 465173.0932, Accuracy: 0.6111, Time: 89.26s
[Rank 0] Epoch 1, Loss: 465059.2956, Accuracy: 0.6112, Time: 88.51s
[Rank 0] Epoch 2, Loss: 465488.4534, Accuracy: 0.6109, Time: 88.81s
[Rank 1] Epoch 2, Loss: 464743.5381, Accuracy: 0.6115, Time: 88.81s
[Rank 0] Epoch 3, Loss: 465279.6081, Accuracy: 0.6110, Time: 89.11s
[Rank 1] Epoch 3, Loss: 464951.9862, Accuracy: 0.6113, Time: 89.11s
[Rank 0] Epoch 4, Loss: 465098.3581, Accuracy: 0.6112, Time: 88.58s
[Rank 1] Epoch 4, Loss: 465134.8252, Accuracy: 0.6112, Time: 88.58s
[Rank 0] Epoch 5, Loss: 466338.8506, Accuracy: 0.6101, Time: 88.12s
🕒 Total training time on Rank 0: 443.12 seconds
[Rank 1] Epoch 5, Loss: 463893.9354, Accuracy: 0.6122, Time: 88.12s
(base) [kishnani.j@explorer-02 FinalProject_Ron]$
```

Figure 4:

Using 6 CPUs, the training time reduced to 443.12s with DistributedDataParallel in effect. Profiler logs show balanced usage across core operations, including gradient updates, forward passes, and loss computation. This setup offered a good parallelization sweet spot, improving performance by nearly 12% from 2-core runs, while maintaining computational stability without over-parallelization overhead.

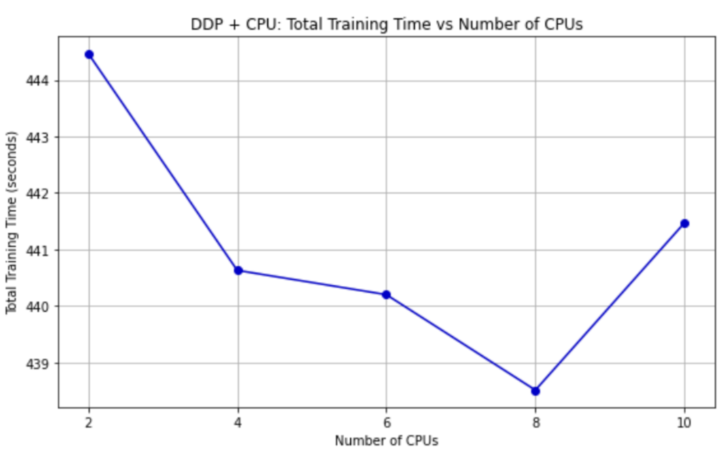


Figure :Total Training Time vs Number of CPUs (DDP + CPU)

This graph illustrates the effect of increasing the number of CPU cores (2, 4, 6, 8, and 10) on the total training time when using Distributed Data Parallel (DDP) with CPU resources.

Key Observations:

- Training time decreases from 2 to 8 cores, showing effective parallelism.
- The lowest training time is achieved with 8 cores (≈ 438.51s).
- At 10 cores, training time increases slightly (≈ 441.47s), indicating diminishing returns due to overhead from synchronization and thread scheduling.

This behavior is consistent with Amdahl's Law, where the benefits of adding more cores diminish as the parallelizable portion of the workload decreases. It highlights the importance of balancing core usage against parallel overhead.

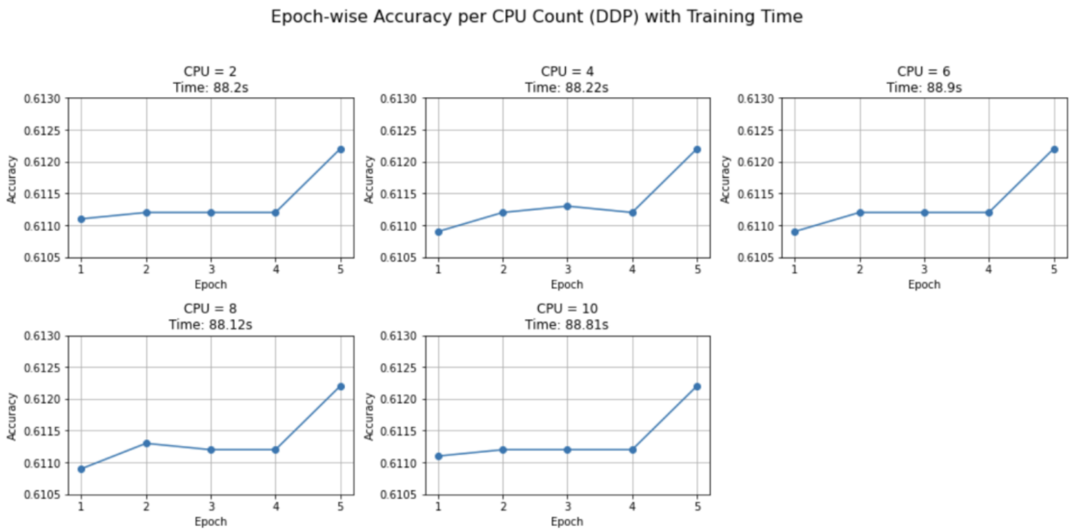


Figure:Epoch-wise Accuracy across varying CPU counts using DDP. Although training time remains relatively stable (~88s), slight variations in accuracy across CPUs (2 to 10) are observed. This supports DDP's consistency and correctness across distributed processes

The accuracy remains stable (~0.6112) across all CPU counts, showing that DDP ensures consistent results. Training time (~88s) doesn't improve with more CPUs, indicating parallel overhead and limited scalability due to synchronization costs.

Conclusion

- Successfully implemented sentiment prediction on Netflix reviews using deep learning models.
- Applied both CPU-based and GPU-based parallel processing techniques to optimize training performance.
- Utilized Distributed Data Parallel (DDP) and Fully Sharded Data Parallel (FSDP) strategies in PyTorch for GPU acceleration.
- Conducted experiments with different configurations (1–4 GPUs and multi-core CPUs) to evaluate scalability and efficiency.

- Observed that GPU-FSDP with 4 GPUs provided the best wall-clock training performance.
- Performed system profiling using PyTorch Profiler to analyze CUDA kernel usage and CPU efficiency.
- Demonstrated that parallelism significantly reduces training time, especially in large-scale ML pipelines.
- Concluded that integrating parallel computing enhances deep learning workflows for real-time, high-volume applications like recommender systems.

Reference

[1] Liu, Handan. *CSYE7105 Lecture Slides – Week 1 to Week 12.* Northeastern University, Spring 2025.
Topics include: Dask, multiprocessing, distributed computing, PyTorch DDP/FSDP, and performance analysis.
[2] Liu, Handan. *CSYE7105 Sample GPU Coding File – DDP/FSDP Training Templates.* Northeastern University, Spring 2025.
Used as a reference for writing PyTorch-based distributed training code on Discovery Cluster.
[3] Kaggle: 1.5 Million Netflix Google Store Reviews Dataset
https://www.kaggle.com/datasets/bwandowando/1-5-million-netflix-google-store-reviews/data
[4] NVIDIA Tesla P100 GPU Architecture Documentation
https://images.nvidia.com/content/tesla/pdf/tesla-p100-PCIe-datasheet.pdf
[5] Intel Xeon E5-2680 v4 CPU Specifications
https://ark.intel.com/products/91770/Intel-Xeon-Processor-E5-2680-v4-35M-Cache-2-40-GHz-