

//----myV0

Problem Statement:

You are tasked with simulating a distributed ML training job using multiple parallel trainer processes. The training is simplified to summing numeric data.

Requirements:

1. Input Data:

 Use a CSV file or an in-memory array that contains one numeric value per row (e.g., [1, 2, 3, ..., 1000] □).

2. Parallel Trainers:

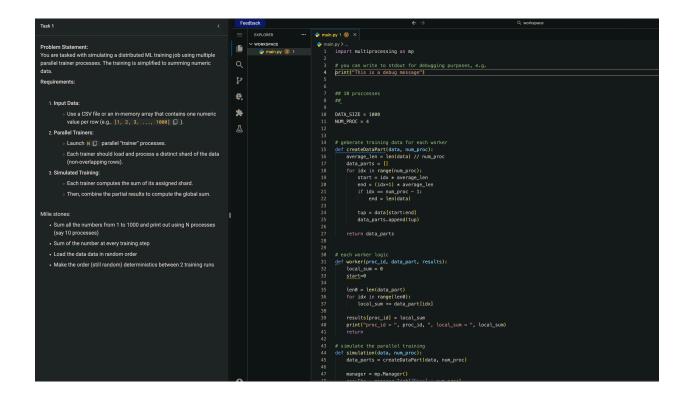
- ∘ Launch N □ parallel "trainer" processes.
- Each trainer should load and process a distinct shard of the data (non-overlapping rows).

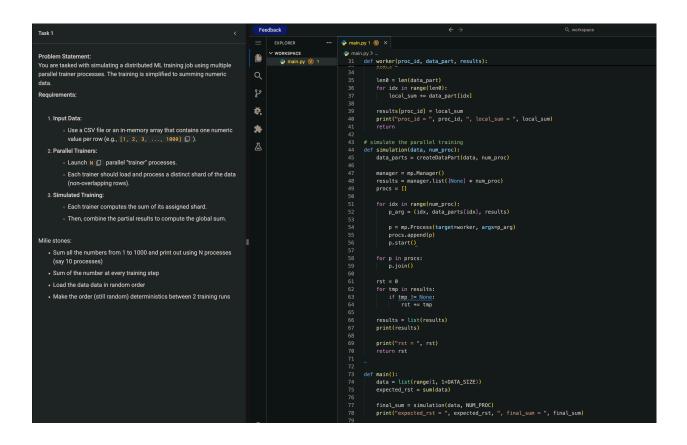
3. Simulated Training:

- Each trainer computes the sum of its assigned shard.
- Then, combine the partial results to compute the global sum.

Milie stones:

- Sum all the numbers from 1 to 1000 and print out using N processes (say 10 processes)
- · Sum of the number at every training step
- · Load the data data in random order
- Make the order (still random) deterministics between 2 training runs





#	
simulating distributed ml training with checkpointing	

objective:

implement a simplified version of a distributed machine leanning (ml) training job to demonstrate you understanding of parallelism, data partitioning, and fault tolorance

problem statement:

you are tasked with simulating a distributed ml training job using parallel trainer processes. the training is simplified to summing numeric data

requirements:

1. input data

using a csv file or an im-memory array that contains one numeric value per row(re.g., [1,2,3,..., 1000])

2. parallel trainers:

launch N parallel "trainer" processes each trainer should be load and process a distinct shard of the data (non-overlapping rows)

3. simulated training

each trainer computes the sum of its assigned shard then combine the partial results to compute the global sum

- 4. sum of the number at every straining step
- 5. load the data in random order
- 6. make the order (still random) deterministic between 2 training runs
- 7. please use this code as an example