**Assignment Task - Parallel/Distributed ML Implementation**

**Objective:**

You are expected to design a parallel/distributed machine learning solution by integrating foundational optimization techniques and GPU-accelerated tools, as described in the assignment problem statement. Your design should demonstrate your ability to evaluate trade-offs and system-level decisions for performance optimization in ML systems.

**Problem Statement 3: Parallel K-Means Clustering using cuML and Multi-threaded NumPy**

Generate a synthetic dataset of 500,000 samples and 5 features and perform K-Means clustering in two ways:

Using a multi-threaded CPU implementation (NumPy/Scikit-learn)

Using cuML’s GPU-based KMeans

Tasks:

* Measure convergence time and silhouette scores
* Visualize cluster separability in 2D using PCA
* Analyze scalability and parallelism patterns

Demonstrate the system-level differences in unsupervised learning performance when scaled across CPU and GPU execution environments.

You are required to prepare a design document as part of your assignment submission. Please **address the following sections in detail**:

**1. Design Overview [3 Marks]**

Provide a high-level explanation of your proposed approach:

* Mention whether you are using Data Parallelism, Task Parallelism, or a Hybrid strategy.
* Justify your choice of strategy in the context of the selected ML algorithm(s).

**2. System Architecture Diagram [3 Marks]**

Include a diagram (block diagram or flowchart) showing:

* Interaction between components (e.g., data preprocessing, training loops, evaluation modules)
* Execution pathways on CPU vs. GPU
* Integration of cuML vs. native Python-based code

**3. Parallelization Strategy [3 Marks]**

Describe the parallelization method in detail:

* Are you distributing data or functional tasks?
* Handling of mini-batches, gradient updates, parameter synchronization.
* If you're using GPU-based Random Forest: Compare the internal parallelization strategies of cuML vs. scikit-learn.

**4. Development Environment [3 Marks]**

Provide a table or list with details of your implementation environment.

* Programming Language
* ML Libraries
* Data Handling
* Visualization
* Dataset
* Preprocessing

**5. Execution Platform & Implementation [10 Marks]**

Provide clear information on your execution setup:

* Hardware used (e.g., CPU cores, GPU model)
* Local system or cloud platform (e.g., AWS EC2, Colab, Google Cloud)
* CUDA/CuML compatibility, GPU memory utilization, etc.
* Execution strategy (e.g., batch size, number of workers, etc.)
* Implementation – code & outputs.

**6. Initial Challenges Identified [3 Marks]**

List potential or observed implementation challenges:

* Computation bottlenecks (e.g., single-threaded vs. parallel libraries, CPU-bound preprocessing)
* Compatibility issues (e.g., cuML only works on GPUs with CUDA)
* Communication costs in distributed mode (e.g., during syncing gradients or batch transfers)
* Data preprocessing overhead (especially in large datasets or cuDF handling)
* Fault tolerance (especially when scaling out or training on cloud nodes)

**Submission Format**

Submit your design document in **Word or PDF** format and Python notebooks.

Include:

* Labeled diagrams
* Tables, screenshots of outputs
* Python scripts or Jupyter notebooks
* Code outputs must be included as screenshots; submissions with code-only will not be evaluated.

Clearly mention:

* Assumptions
* Library versions
* Hardware specifications

Please follow the structure carefully. Submissions that miss any of the sections or required artifacts (e.g., outputs, diagrams) will be penalized.