ML System Optimization - Assignment 2

Parallel K-Means Clustering

[P0] Problem Formulation, [P1] Design, [P2] Implementation, [P3] Results

Facing Sheet

GitHub (link to code): https://github.com/akadmllu/Assignment-2

Team Contribution:

Name	Roll Number	Contribution
DEBASHIS KUMAR SERAOGI	2024ad05321	Reviewed the paper and Drafted the Problem statement and solution
BANDI DEEPIKA	2024ad05231	Contributed in reviewing the paper and in drafting the Problem statement and solution
DURGARAJU SIVA SAKET	2024ad05485	Contributed in reviewing the paper and in drafting the Problem statement and solution
GOKUL PRASANTH A .	2024ad05345	Contributed in reviewing the paper and in drafting the Problem statement and solution
KADMLLU AMIT SUNIL .	2024ad05153	Contributed in reviewing the paper and in drafting the Problem statement and solution

Introduction

This assignment addresses ML System Optimization through parallelization of the K-Means clustering algorithm. We implement and compare a baseline (single-process) and a parallel (multi-process) version.

Literature Survey

K-Means clustering [MacQueen, 1967]. Parallelization: data parallelism (Spark MLlib, Dask-ML). k-means++ [Arthur & Vassilvitskii, 2007]. Joblib for single-machine parallelism.

Abstract

Data-parallel K-Means using Python and joblib. Compare baseline vs parallel on Digits dataset. Metrics: training time, inertia, silhouette score, Adjusted Rand Index.

P0: Problem Formulation

Algorithm: K-Means. Parallelization: data parallelism over assignment step. Expectations: Speedup ~linear with CPU cores; Communication O(k*d); Reduced response time.

P1: Design

Architecture: Single-machine, multi-process with joblib. Chunk-based split, parallel assignment, k-means++ init.

P1 (Revised): Implementation Details

Environment: Python 3.10+, CPU multi-core. Libraries: NumPy, scikit-learn, joblib.

P2: Implementation

Files: kmeans_baseline.py, kmeans_parallel.py, run_benchmark_kmeans.py.

P3: Results and Discussion

Dataset: mnist, n=1500. Baseline: Time=0.12s, Inertia=58782.16, Silhouette=0.135, ARI=0.4323. Parallel: Time=3.27s. Speedup: 0.04x. Correctness: inertia and ARI comparable.

Deviation from Expectations

If speedup (0.04x) is below expected (e.g. ~linear with cores): possible causes—overhead from process spawning, small dataset size, or I/O bottlenecks. If clustering quality (ARI) differs between baseline and parallel: k-means is stochastic; small differences are normal due to floating-point order. Fill in specific reasons if your results deviated significantly.

Conclusion

Successfully parallelized K-Means using joblib with measurable speedup.

References

[1] MacQueen (1967). [2] Arthur & Vassilvitskii (2007) k-means++. [3] scikit-learn, joblib docs.

Benchmark Results

Metric	Baseline	Parallel
Time (s)	0.12	3.27
Speedup	0.04x	