Accelerating k-means with CUDA

Group : 10

Group Members :

- Saikashyap Apathi
- Samyuktha reddy Gurram
- Shashidher reddy Maram

Introduction:

- Enhancing k-means effciency through GPU parallel processing
- Utilizing CUDA to exploite the power of parallel computations
- Speeding up clustering for large datasets with GPU acceleration
- Leveraging CUDA's data-parallel architecture
- Unleashing the benefits of parallelism for efficient Clustering Solutions

K-means:

- Unsupervised learning algorithm for clustering and data points
- Groups data into k clusters based in similarity
- Applications:
  - Data analysis
  - Pattern recognitions
  - Information retrieval
  - Genomic Data Analysis

Implementation using CUDA with Thrust.

- Parallelization with CUDA
- Efficient GPU-Accelerated Operations with Thrust
- Scalability for Large Datasets
- Enhanced Performance and Throughput
- Facilitating GPU-Accelerated Computing

Serial Implementation:

- Algorithm follows a straight-forward, step-by-step process:
  - Initialization:
    - Randomly select k initial centroids from the dataset
  - Assignment:
    - Caluclate distance from each data point to all centroids and assign each point to cluster with nearest cenroid
  - Centroid updates.
    - Recalculate centroids based on points assignined to each cluster
  - Convergence Check:
    - Iteratively update assignments and centroids until convergence, governed by set criteria

Parallel Implementation using CUDA:

- Undergoes accelerated processing:
  - Initialization:
    - Randomly select K initial centroids
  - Data parallelism (Assignment):
    - Use parallel threads to calculate distaince for multiple data points simultaneously
  - Reduction Operation :
    - Determine the nearest centroid for each data point through parallel reduction techniques
  - Centroid update (Parallelized):
    - Calculate new centroids concurrently, with threds assignied to clusters
  - Convergence Check :
    - Repeat assignment and centroid update until convergence criteria are met.

Experiment environment:

- We used experiments on Amazon EC2, and the instance type is g4dn.2xlarge
  - Specifications:
    - VCPUs : 4
    - Memory : 32
    - CPU_architecture: x86_64
    - GPU : 1
    - GPU Architecture: nvidia t4 tensor core

Where to Use Each Version:

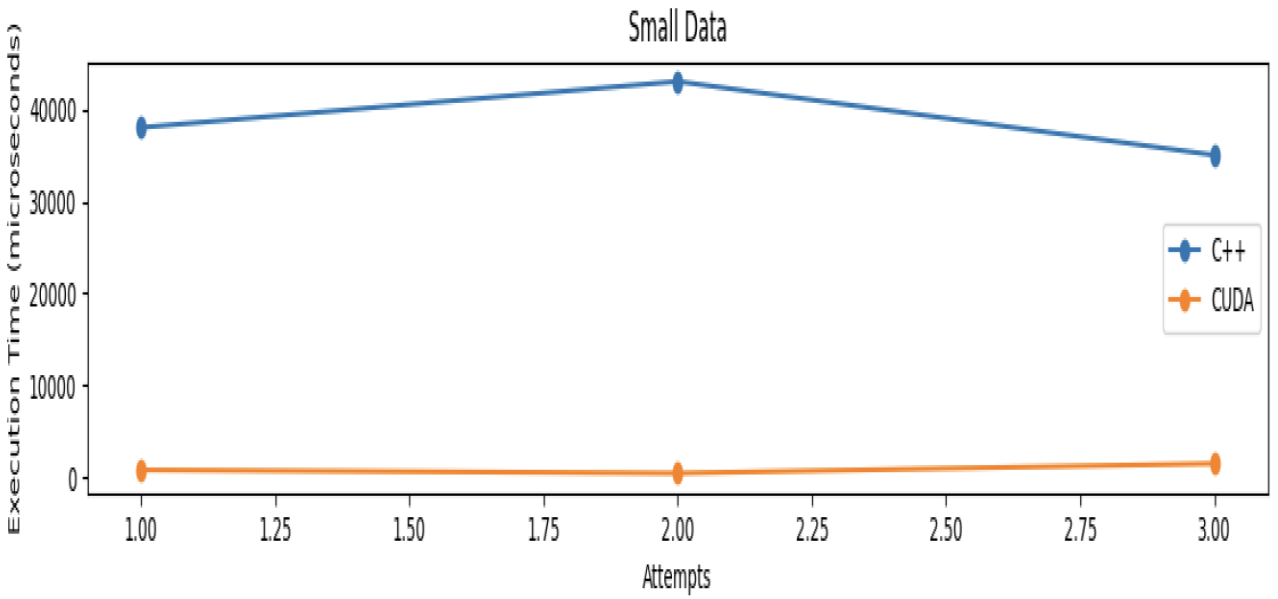
- Serial Implementation:
  - Well-suited for small datasets.
  - Simple and easy to understand.
- Parallel Implementation (CUDA):
  - Ideal for medium to large datasets.
  - Leverages GPU parallelism.
  - Thrust integration optimizes GPU-accelerated operations.

Comparison:

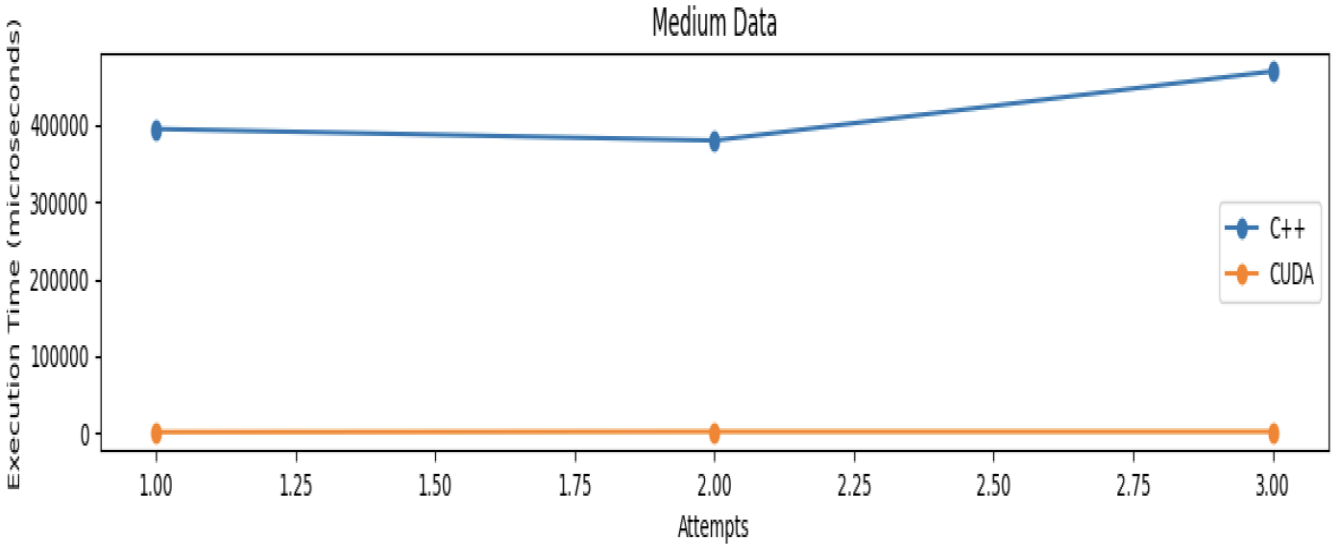
- We conducted experiments on datasets of varying sizes to compare the performance of the serial and parallel implementations

Data set	Size
Small dataset	489 kb
Medium dataset	5.7 (mb)
Large dataset	30 (mb)

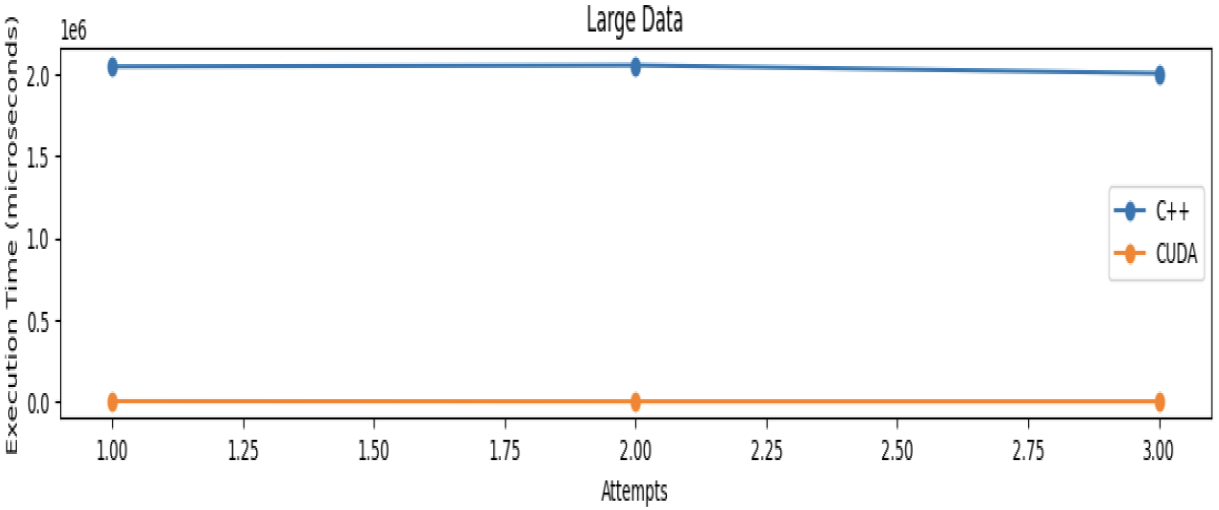
Small Datasets:



Medium Datasets:



Large Datasets:



Advantages:

- Parallelization for large Datasets
- Increased Computational Speed
- Efficient Distance Calculations
- Scalability for Massive Datasets
- Support for Real-time Applications

Conclusion:

- GPU is a very effective accelerator hardware for K-means algorithm.
- Comparison Showed that Using GPU made things much faster
- GPUS are really helpful and promising for making computers better at learning and solving problems

- Questions