Leveraging Parallel Processing for Advanced Graph Computations

Kaustubh Shivdikar, *PHD Student, Northeastern University*
Ayman Alabbasi, *REU Participant Student, North Shore Community College*
David Kaeli, *College Of Engineering, Northeastern University*

Northeastern University College of Engineering
Northeastern University Center for STEM Education

Abstract

Goal - The goal is to accelerate various machine learning workloads while also lowering execution time for a more functional review

Mission - To democratize Graph Neural Network accelerator development

Hypothesis - In essence it's all about efficiency and production. We reengineer on chip communication to speed up GNN's. With heavy emphasis on communication.

Who Will This Benefit?

GNNs assist in protein molecule analysis in Bioinformatics

Also social network analysis, traffic prediction, and computer vision.

GNNs enhance machine learning by modeling complex relationships in data structured as graphs or networks.

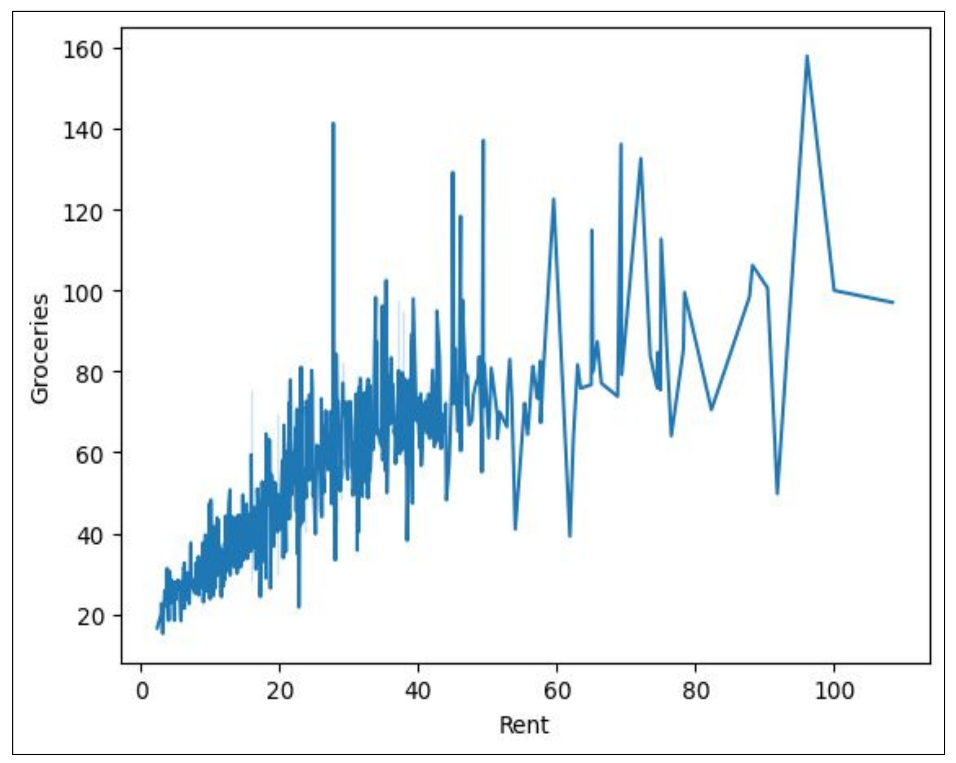
Background

Graph Neural Networks: a deep learning model for data, capturing dependencies between nodes.
Python: versatile, programming language known for its readability, and libraries for diverse applications.
Parallel Computing: Simultaneous execution of tasks using multiple processors to achieve faster computation and increased efficiency.

Research

Cost of Living Index

Goal: to analyze and compare living expenses across different locations
Process: We utilized the 2022 Cost of Living Index as our dataset



Single Threaded Matrix Multiplication

Goal: Our goal was to solve for Matrix C using single threading

Reasoning: To understand efficiency and performance trade-offs in single-threaded implementations

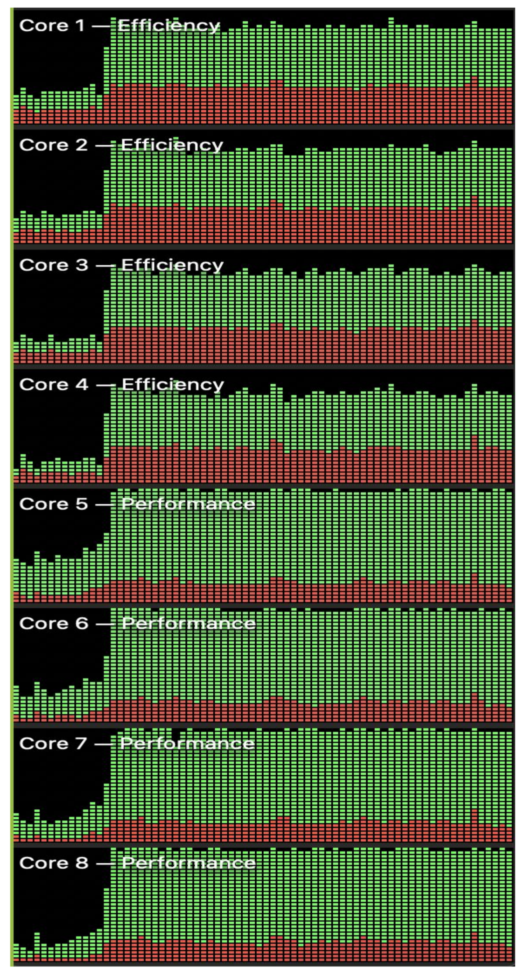
```
import numpy as np

matrix_A = np.array([[1,2,3], [4,5,6], [7,8,9]])
matrix_B = np.array([[1,2,3], [4,5,6], [7,8,9]])

#The np.dot function calculates the dot product between the two matrices
matrix_C = np.dot(matrix_A, matrix_B)

print("Matrix A:")
print(matrix_A)

print("Matrix B:")
print(matrix_B)

print("Matrix C:")
print(matrix_C)
```

Solution

```
Matrix A:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Matrix B:
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Matrix C:
[[ 30  36  42]
 [ 66  81  96]
 [102 126 150]]
```

Multi Threaded Parallel Computing

Multi Threading: the ability of a processor to execute multiple threads concurrently. in a CPU



Assignment: change the initial code from addition to multiplication while implementing multi threading techniques

```
@ray.remote
def my_function(data):
    matrix_A, matrix_B, i = data
    x = i % 3
    y = i // 3
    matrix_C_element = 0
    for k in range(3):
        matrix_C_element += matrix_A[x][k] * matrix_B[k][y]
    return matrix_C_element

# Define matrices A and B
matrix_A = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
matrix_B = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])

# Define the total number of elements in the matrices
total_numbers = 9
numbers = list(range(total_numbers))

# Execute the matrix multiplication
result_ids = []
for i in numbers:
    data = (matrix_A, matrix_B, i)
    result_ids.append(my_function.remote(data))

# Gather results
results = ray.get(result_ids)

# Create matrix C for storing the result
matrix_C = np.zeros((3, 3))

# Fill matrix C with values
for i in range(3):
    for j in range(3):
        matrix_C[i][j] = results[j * 3 + i]

print(matrix_C)
```

```
[[ 30.  36.  42.]
 [ 66.  81.  96.]
 [102. 126. 150.]]
```

Solution: multiplied matrix C using MT & PC

Why Multi Threading: MT streamlines the utilization of resources as the threads share the same memory and data space. It also allows the concurrent appearance of multiple tasks and reduces the response time. This improves the performance.

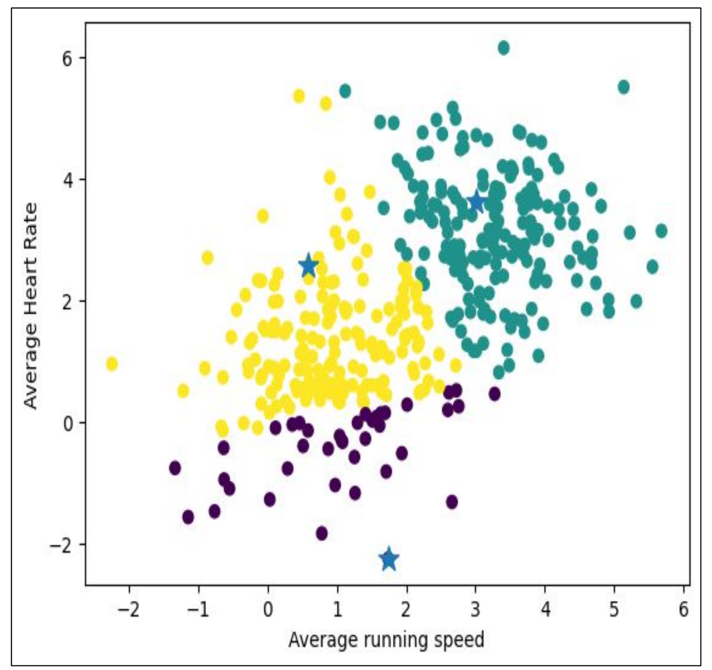
Why Parallel Computing: accelerates tasks by distributing them across multiple processors, reducing execution time, increasing throughput, solving larger problems, and improving efficiency.

K Means Clustering

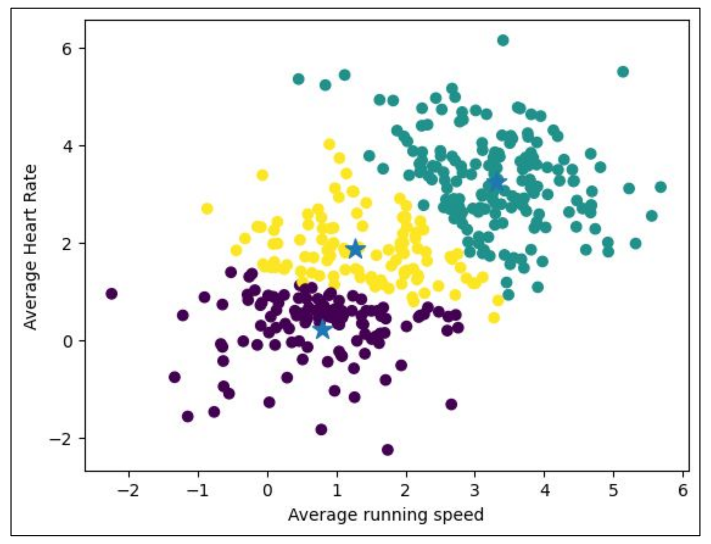
K Means Clustering: is a machine learning technique that partitions data into K clusters based on similarity, aiming to minimize variance.

Assignment: Given a collection of data points categorize them into three clusters based on levels of fitness.

First Iteration: We divided the data points into 3 distinct clusters, but the centroids are not centered and the groups are uncoordinated.



Final Iteration: We obtained the most accurate response after running it five times.The data points are divided evenly across three established clusters. Each clusters centroids are located exactly in the center.



Conclusion

Our research demonstrates the potential of using parallel processing and multithreading for complex graph computations. Our approach enhances performance and scalability by leveraging the capabilities of several processors. Advantages such as faster processing, better resource usage, and the capacity to manage complex graph structures, make it a tempting solution for a variety of real-world applications. In the age of big data, embracing parallel computing is critical for realizing the full potential of graph analytics.

NSF REU-Site: Research Experience for Undergraduates site to create "Pathways Opening World Energy Resources" (REU - POWER)

This work was supported by the National Science Foundation
Grant ##2150417