CPU PySpark
Vs
GPU CUDA
Gravity Simulator

Our Team: Group 1

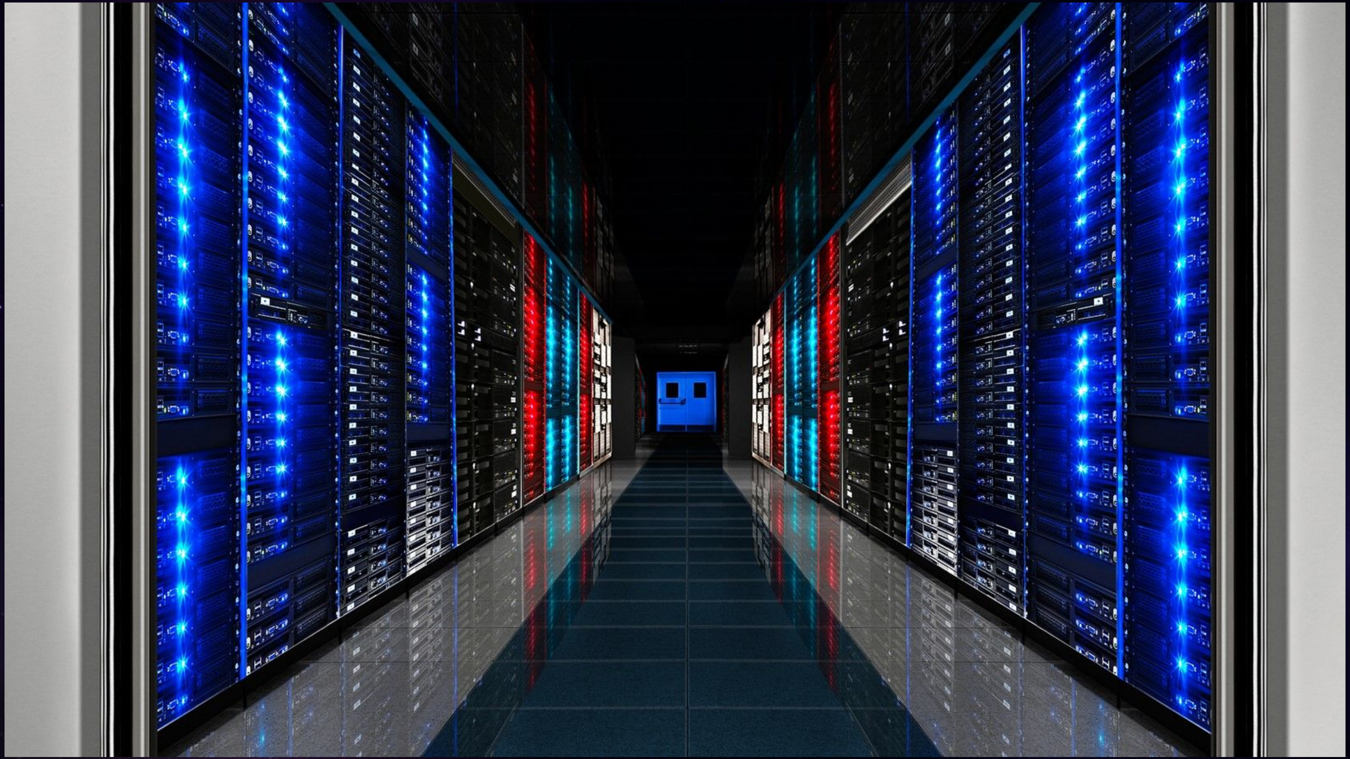
Pranav Angiya Janarthanan

Aditya Sharma

Table of Contents

01 Objectives

Why Compare?

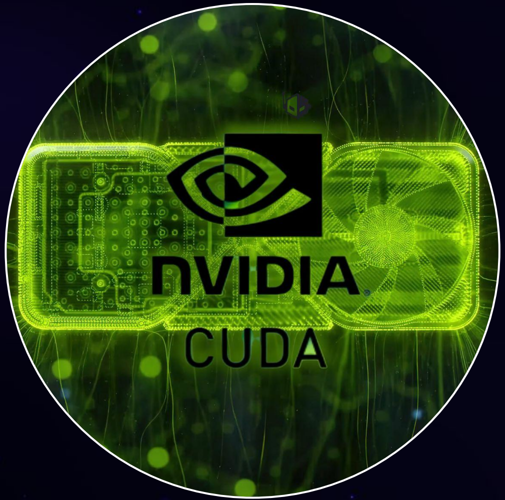


Supercomputer stations are becoming extremely popular with multi-cluster systems working together

GPU Computational Advantage

- Can perform thousands of calculations at once
- Easy to collect and integrate
- High demand for AI and other ML based calculations

Which is Better?



GPU-Based (CUDA)

Cluster-Based (PySpark)

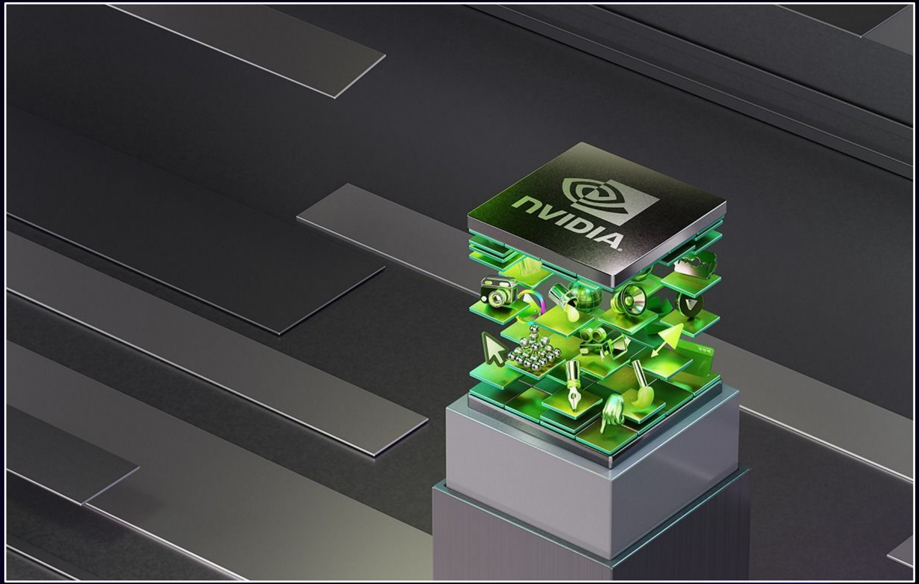
02

Background

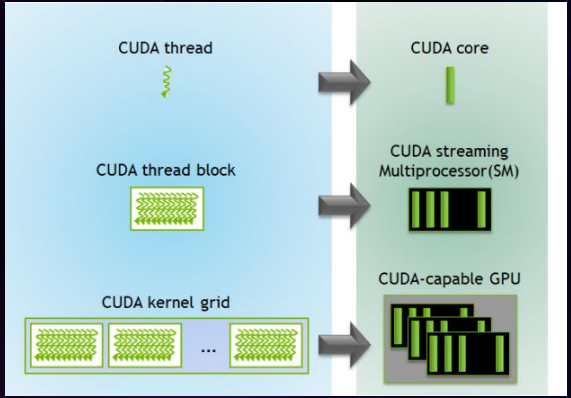
What is CUDA and Pyspark?

CUDA

- Parallel Computing Platform developed by NVIDIA
- Traditionally used for graphics rendering
- Utilizes thousands of GPU Cores to run parallel computational tasks for AI, data processing, etc



Computing Architecture



Threads

Handles a set of computations (Traditionally single computations per thread)

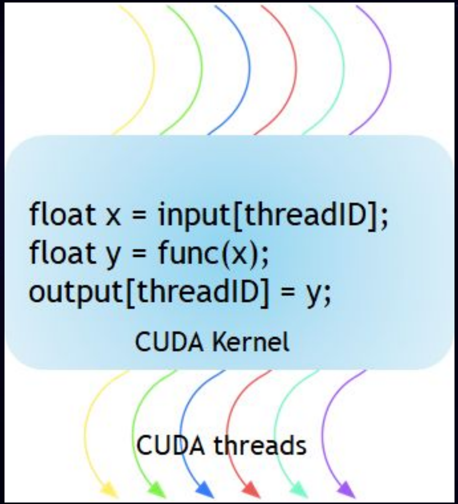
Thread Blocks

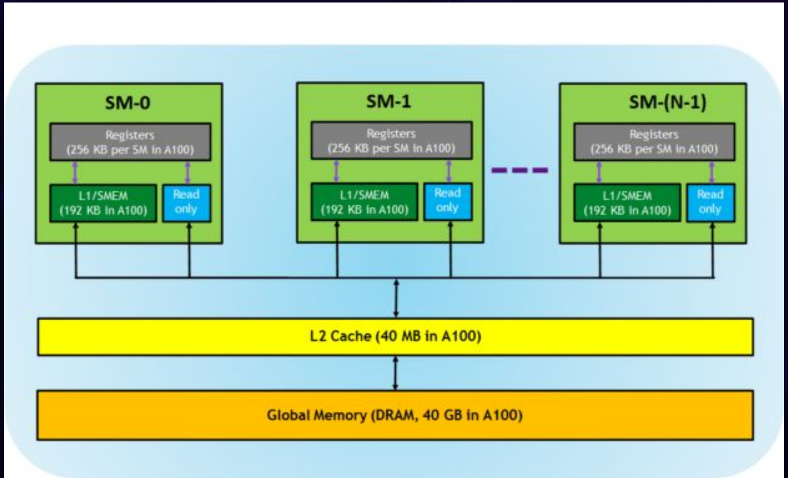
Collects a block of threads that often share some small and fast memory (Chosen in multiples of 32)

Kernel Grid

A stack or grid of Blocks used to organize dimensions (1D for vectors, 2D or 3D for multidimensional problems)

Kernels and Memory Management





Functions on GPUs

Functions launched from CPU (host) to run onto GPU (Device) (Requires memory allocated variables)

Memory Levels

Threads have registers. Streaming Multiprocessors have shared memory for threads in blocks. Grids have L2 and DRAM

PySpark

Fault Tolerance

Parallel Data Processing

Distributed Computing

Local and Cluster Based

03

Methodology

What are we comparing?

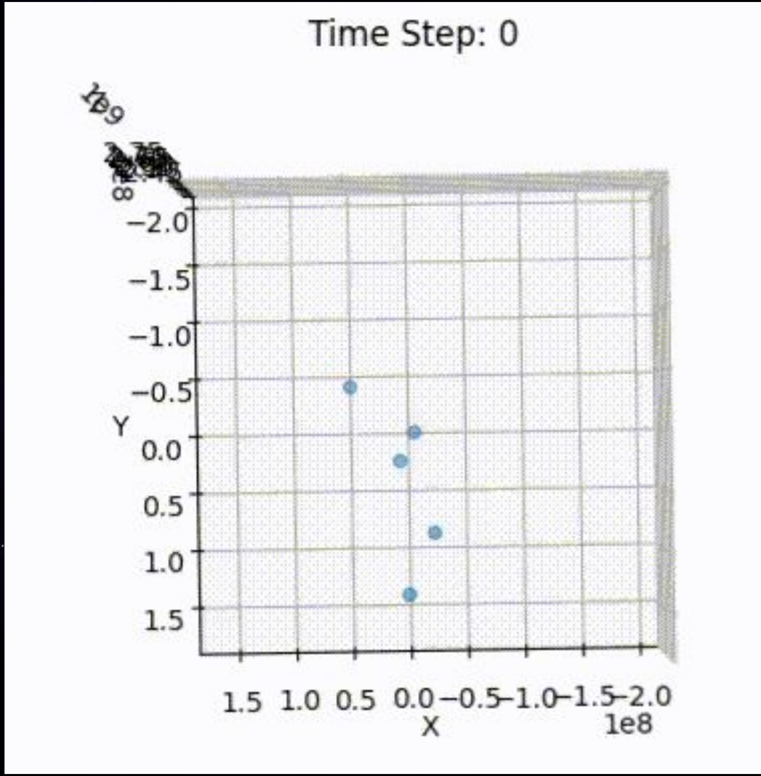
Gravity Simulator

- Large amount of vector calculations for each body

$$F_g = G\frac{m_1 m_2}{r^2}$$

- Scalable Problem for Size N

- Places stress on resources like CPU, GPU, and memory



System Setup

PySpark (Local)

- 4 Intel i7 Cores per Worker Node
- 6 GB RAM per Session
- 1 Worker Node
- 8 Threads

CUDA (Google Colab)

- Tesla T4
- 2560 CUDA Cores
- 15 GB GPU Ram

CUDA Code

```
Body *d_bodies;
float3 *d_acc;
cudaMalloc(&d_bodies, n*sizeof(Body));      // Allocate Memory in GPU for Bodies
cudaMalloc(&d_acc, n*sizeof(float3));       // Allocate Memory in GPU for Accelerations

cudaMemcpy(d_bodies, h_bodies, n*sizeof(Body), cudaMemcpyHostToDevice);  // Copy data from CPU to GPU

// Setup CUDA launch parameters
int blockSize = 64;          // Number of threads per block
int gridSize = 2;            // Number of blocks

// Benchmark simulation
auto start_sim = std::chrono::high_resolution_clock::now();
for (int s = 0; s < steps; s++) {
    compute_accelerations<<<gridSize, blockSize>>>(d_bodies, d_acc, n, G, eps2);    // Run kernal to compute accelerations
    update_bodies<<<gridSize, blockSize>>>(d_bodies, d_acc, n, dt);                 // Run kernal to update bodies
    cudaDeviceSynchronize();
}
auto end_sim = std::chrono::high_resolution_clock::now();
double sim_time = std::chrono::duration<double>(end_sim - start_sim).count();

cudaMemcpy(h_bodies, d_bodies, n*sizeof(Body), cudaMemcpyDeviceToHost);          // Copy data from GPU back to CPU
```

04

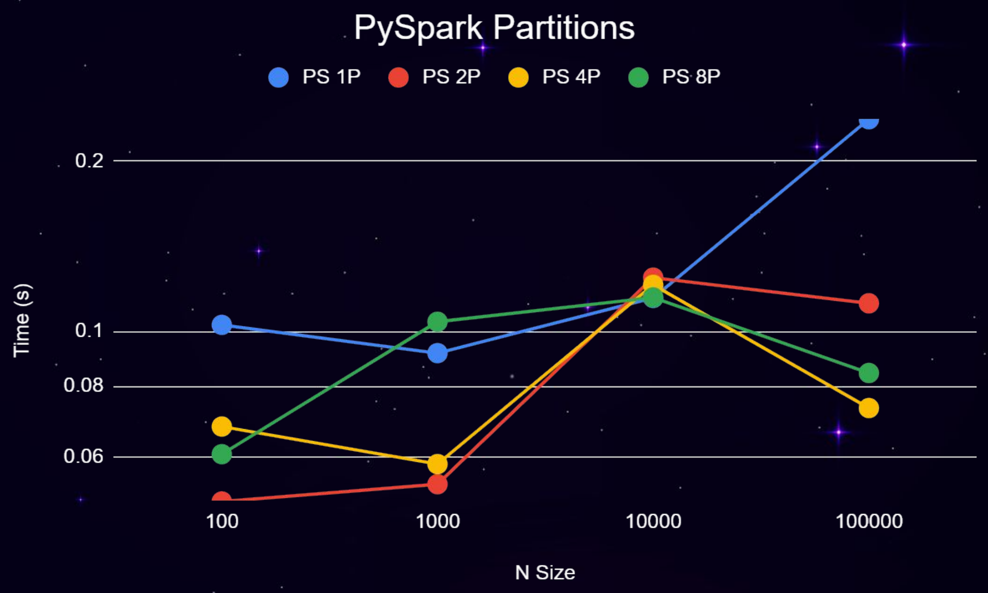
Analysis

How did each perform?

Results

N	PySpark 1	PySpark 2	PySpark 4	PySpark 8	CUDA
100	0.1026408672	0.049995422	0.0678431988	0.06061911583	0.010807
1000	0.09149217606	0.0536363125	0.05823016167	0.1039690971	0.067764
10,000	0.114677023	0.1244206429	0.1207845211	0.1145432327	0.392621
100,000	0.2371878624	0.1120731831	0.07315087318	0.0843768120	11.347148

PySpark Partitions



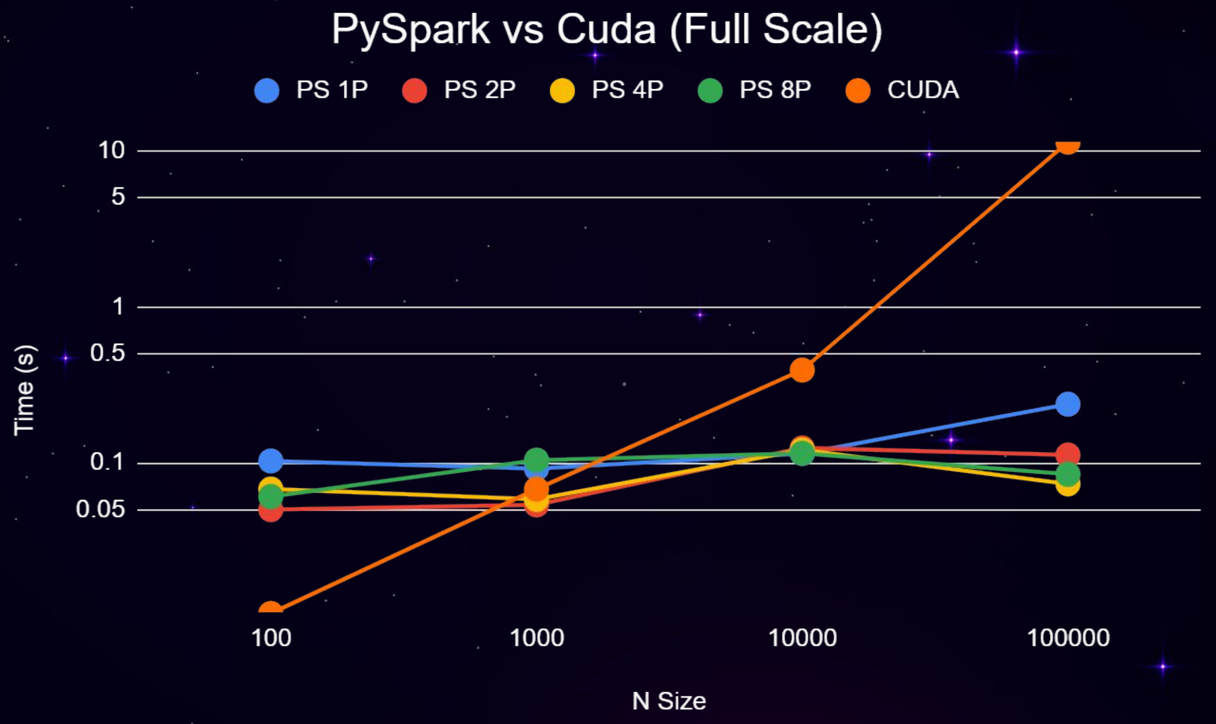
1 Partition

2 Partition

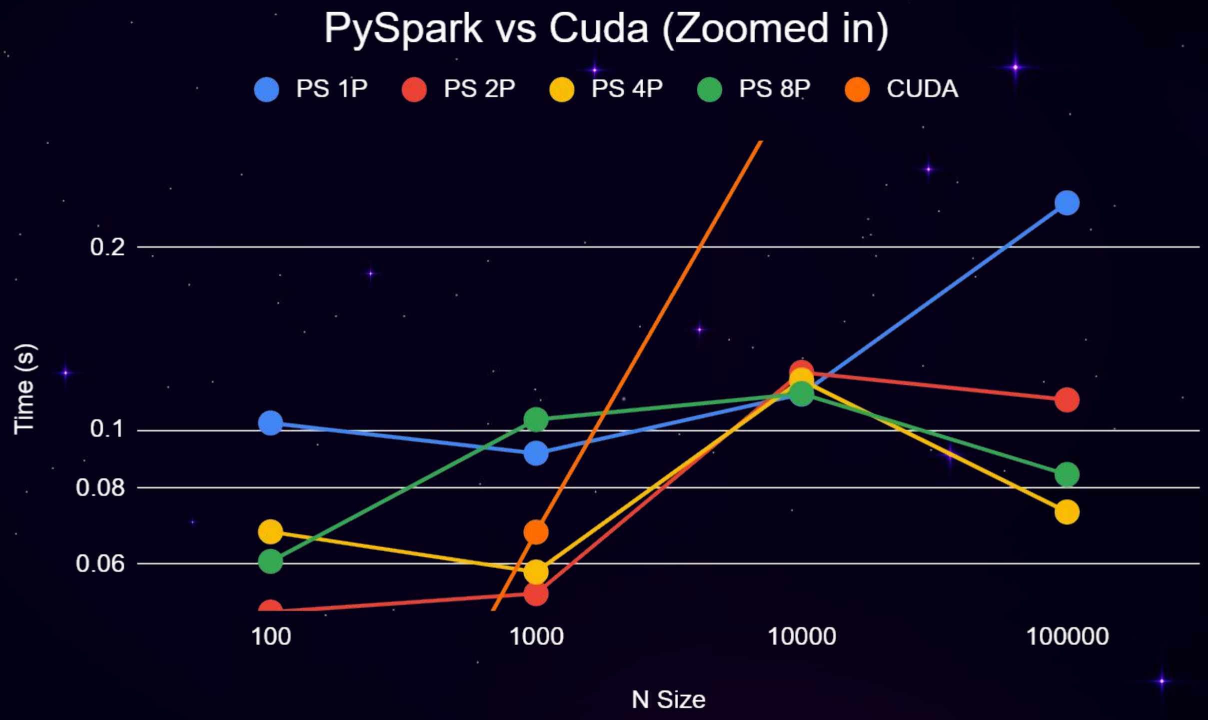
4 Partition

8 Partition

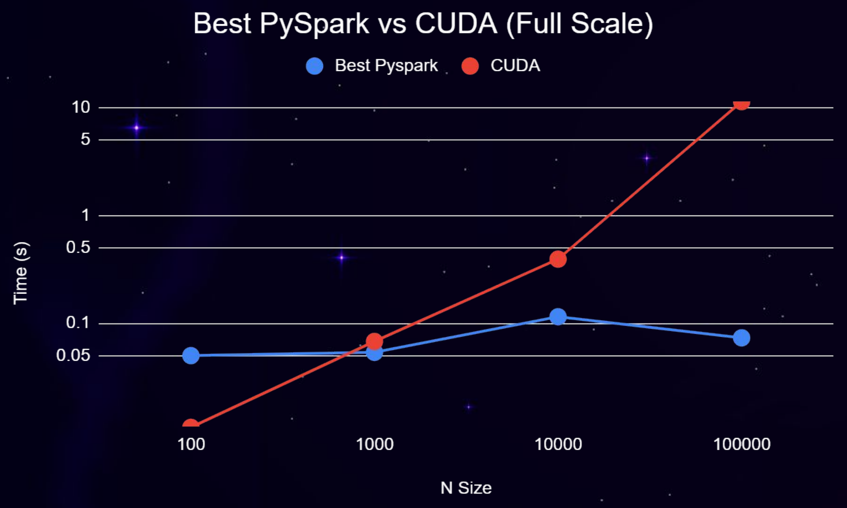
PySpark vs CUDA (Full Scale)

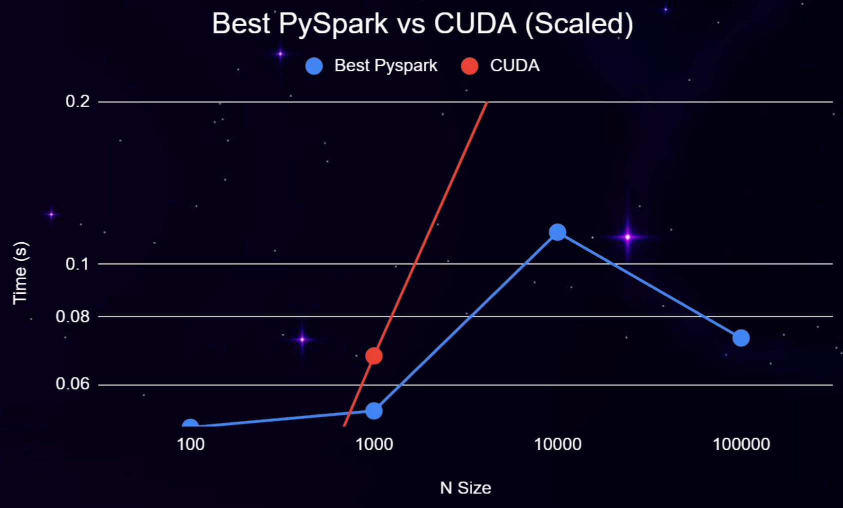


PySpark vs CUDA (Zoomed in)



Best PySpark vs CUDA





05
Conclusions

What did we find?

PySpark vs CUDA

Low N performance for CUDA

- Given low values for N, CUDA performs better
- Limited by Block Size to 1024
- Scaling issue with data past 1000

PySpark Scalability with partitions

- Ability to add Partitions and choose optimal value for performance
- Requires either large dataset or increased partitions and resources

Specialized Use Cases

- CUDA specialized in Neural Networks and Large Parallelization
- PySpark excels with large data manipulation

Thanks

Do you have any questions?

as4108@rutgers.edu
pa446@rutgers.edu