**Performance Comparison Report: CuPy vs. NumPy in Matrix Computations**

**Introduction**

This report evaluates the performance of matrix operations using GPU-accelerated computing with CuPy versus traditional CPU-based computing with NumPy. The task involves setting up a specific dense matrix K and solving a linear system Ku=f, with with 𝑢 as the unknown vector. The matrix K is tridiagonal with specific conditions on its diagonal and sub/super-diagonal elements.

**Methodology**

* **Matrix Setup**: Matrix 𝐾*K* is tridiagonal, with four main diagonal elements, except the last element is 2. The elements directly next to the diagonal are -2. Vector 𝑓 is initialized with all zeros except for the previous element set to 1/*N*​.
* **Solution of Linear Systems**: The equation 𝐾𝑢=𝑓 is solved to find vector 𝑢.
* **Programming Environment**:
  + **GPU**: Using CuPy library, exploiting CUDA toolkit for operations executed on a GPU.
  + **CPU**: NumPy library is used for matrix operations performed solely on the CPU.
* **Performance Measurement**: Execution time for the entire process, including matrix setup and system solving, is recorded for GPU and CPU implementations.

**Result Summary:**

speedup =

Speedup (GPU over CPU): 1.913883419223433 times

**Conclusion**

GPU computing with CuPy significantly enhances performance for solving large systems of equations compared to traditional CPU methods using NumPy. This speedup can benefit high-performance computing applications with standard large matrices and high computational loads. Organizations and researchers should consider integrating GPU-accelerated computations to optimize performance in applicable scenarios.

**Program Output:**

1. **Using CuPy (main\_program.py**)

Matrix K: [[ 4. -2. 0. ... 0. 0. 0.]

[-2. 4. -2. ... 0. 0. 0.]

[ 0. -2. 4. ... 0. 0. 0.]

...

[ 0. 0. 0. ... 4. -2. 0.]

[ 0. 0. 0. ... -2. 4. -2.]

[ 0. 0. 0. ... 0. -2. 2.]]

Matrix f: [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00

0.00000000e+00 3.33333333e-05]

Solution vector u:

[1.66666667e-05 3.33333333e-05 5.00000000e-05 ... 4.99966667e-01

4.99983333e-01 5.00000000e-01]

Value of u[N-1]: 0.4999999998518403

CPU time: 6.958054065704346 seconds

1. **NumPy functions (CPU only:** **main\_program\_Numpy\_CPU\_only.py)**

Matrix K: [[ 4. -2. 0. ... 0. 0. 0.]

[-2. 4. -2. ... 0. 0. 0.]

[ 0. -2. 4. ... 0. 0. 0.]

...

[ 0. 0. 0. ... 4. -2. 0.]

[ 0. 0. 0. ... -2. 4. -2.]

[ 0. 0. 0. ... 0. -2. 2.]]

Matrix f: [0.00000000e+00 0.00000000e+00 0.00000000e+00 ... 0.00000000e+00

0.00000000e+00 3.33333333e-05]

Solution vector u (CPU):

[1.66666667e-05 3.33333333e-05 5.00000000e-05 ... 4.99966667e-01

4.99983333e-01 5.00000000e-01]

Value of u[N-1] on CPU: 0.4999999998518403

CPU execution time (NumPy): 13.316904306411743 seconds