TUGAS KELOMPOK

PEMROSESAN PARALEL

DOSEN PENGAMPU :

AHMAD HERYANTO, S.KOM, M.T.
ADI HERMANSYAH, S.KOM., M.T.

PEMBUAT (KELOMPOK 11):

MUHAMMAD RAFI RIZQULLAH	**(09011282126091)**
M. IKHSAN SETIAWAN	**(09011282126103)**
REZA PALEPI	**(09011282126120)**

FAKULTAS ILMU KOMPUTER

JURUSAN SISTEM KOMPUTER

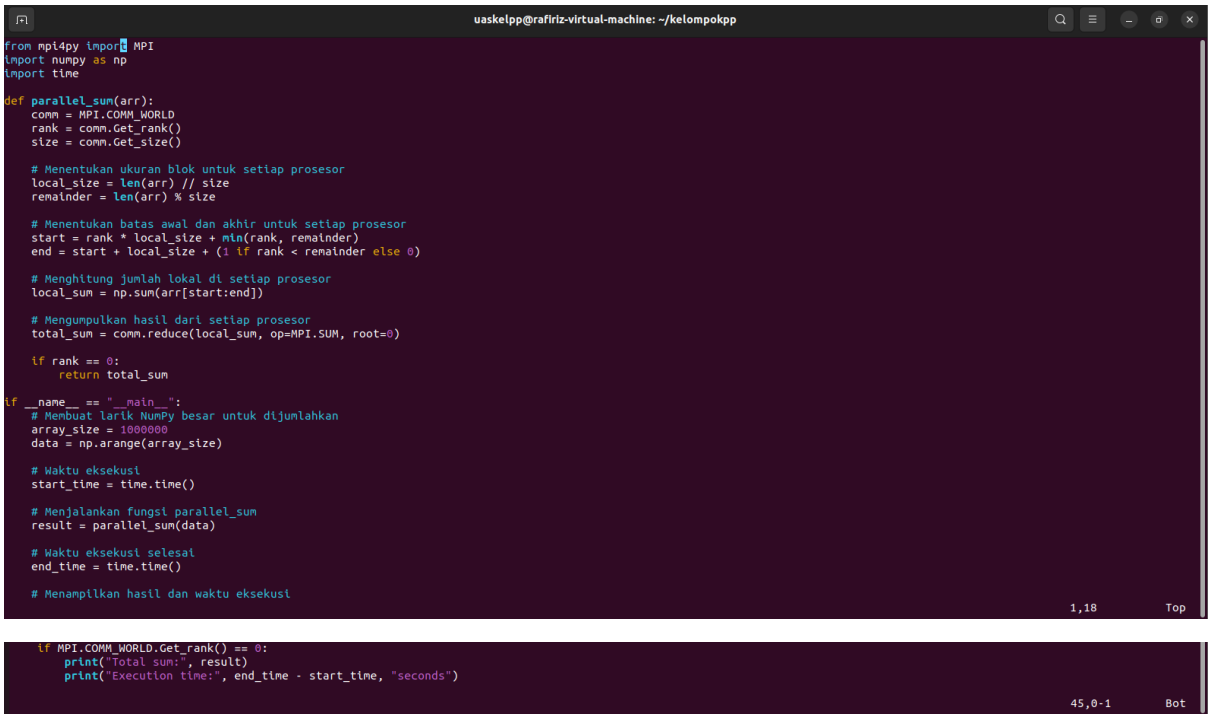
UNIVERSITAS SRIWIJAYA

Laporan Praktikum Eksekusi Program Numerik Python Menggunakan MPI & Numpy Secara Paralel di Ubuntu

Kode yang dibuat dalam mengeksekusi program python tersebut :

```
from mpi4py import MPI
import numpy as np
import time

def parallel_sum(arr):
    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()
    size = comm.Get_size()

    # Menentukan ukuran blok untuk setiap prosesor
    local_size = len(arr) // size
    remainder = len(arr) % size

    # Menentukan batas awal dan akhir untuk setiap prosesor
    start = rank * local_size + min(rank, remainder)
    end = start + local_size + (1 if rank < remainder else 0)

    # Menghitung jumlah lokal di setiap prosesor
    local_sum = np.sum(arr[start:end])

    # Mengumpulkan hasil dari setiap prosesor
    total_sum = comm.reduce(local_sum, op=MPI.SUM, root=0)

    if rank == 0:
        return total_sum

if __name__ == "__main__":
    # Membuat larik NumPy besar untuk dijumlahkan
    array_size = 1000000
    data = np.arange(array_size)

    # Waktu eksekusi
    start_time = time.time()

    # Menjalankan fungsi parallel_sum
    result = parallel_sum(data)

    # Waktu eksekusi selesai
    end_time = time.time()

    # Menampilkan hasil dan waktu eksekusi
    if MPI.COMM_WORLD.Get_rank() == 0:
        print("Total sum:", result)
        print("Execution time:", end_time - start_time, "seconds")
```

Setelah itu eksekusi kode tersebut kedalam mpi4python seperti dibawah ini.



Dan kemudian dapat dijalankan dengan hasil output seperti dibawah ini.

```
uaskelpp@rafiriz-virtual-machine:~/kelompokpp$ mpirun -np 2 -host master,worker1 python3 Task4.py
Total sum: 499999500000
Execution time: 0.013111591339111328 seconds
```

Penjelasan :

Program tersebut adalah program sederhana menggunakan MPI dan NumPy untuk melakukan operasi numerik secara paralel. Program ini melakukan penjumlahan elemen dari suatu larik NumPy secara paralel menggunakan prosesor MPI. Kode ini juga mencakup pengukuran waktu eksekusi program.

Berikut penjelasan singkat dari setiap perintah dalam kode tersebut:

- from mpi4py import MPI: Mengimpor pustaka MPI (mpi4py) untuk mendukung pemrograman paralel dengan MPI.
- import numpy as np: Mengimpor NumPy, pustaka yang berguna untuk operasi numerik pada Python.
- def parallel_sum(arr): Mendefinisikan fungsi parallel_sum yang melakukan penjumlahan elemen array secara paralel.
- comm = MPI.COMM_WORLD, rank = comm.Get_rank(), size = comm.Get_size(): Menginisialisasi komunikator MPI (comm), mendapatkan nomor identitas prosesor (rank), dan mendapatkan jumlah total prosesor (size).
- local_size = len(arr) // size, remainder = len(arr) % size, start = rank * local_size + min(rank, remainder), end = start + local_size + (1 if rank < remainder else 0): Menghitung bagian lokal dari array yang akan dijumlahkan oleh masing-masing prosesor.
- local_sum = np.sum(arr[start:end]): Melakukan penjumlahan lokal pada bagian array yang ditangani oleh masing-masing prosesor.
- total_sum = comm.reduce(local_sum, op=MPI.SUM, root=0): Mengumpulkan hasil penjumlahan dari semua prosesor ke prosesor dengan nomor identitas 0.

- if rank == 0:: Hanya prosesor dengan nomor identitas 0 yang akan menampilkan hasil dan waktu eksekusi.
- array_size = 1000000, data = np.arange(array_size): Membuat larik NumPy besar yang akan dijumlahkan.
- start_time = time.time(): Memulai pengukuran waktu eksekusi.
- result = parallel_sum(data): Menjalankan fungsi parallel_sum untuk melakukan penjumlahan secara paralel.
- end_time = time.time(): Mengakhiri pengukuran waktu eksekusi.
- Menampilkan hasil dan waktu eksekusi jika prosesor dengan nomor identitas 0.
- Perintah eksekusi di terminal (mpirun -n 4 python parallel_sum.py): Menjalankan program menggunakan 4 prosesor dengan perintah mpirun.
- Dengan menggunakan MPI, program ini membagi tugas penjumlahan di antara beberapa prosesor, mempercepat prosesnya, dan menghasilkan waktu eksekusi yang lebih cepat untuk operasi numerik pada larik besar.Print Hasil (Hanya pada Prosesor Utama): Prosesor utama mencetak hasil jumlah elemen secara total.

Program ini menunjukkan konsep dasar penggunaan MPI untuk tugas yang dapat dipecah menjadi bagian-bagian independen yang dapat dijalankan secara paralel pada beberapa prosesor.