

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

def parallel_sum(data, comm):
    local_sum = np.sum(data)
    total_sum = comm.reduce(local_sum, op=MPI.SUM, root=0)
    return total_sum

if __name__ == "__main__":
    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()

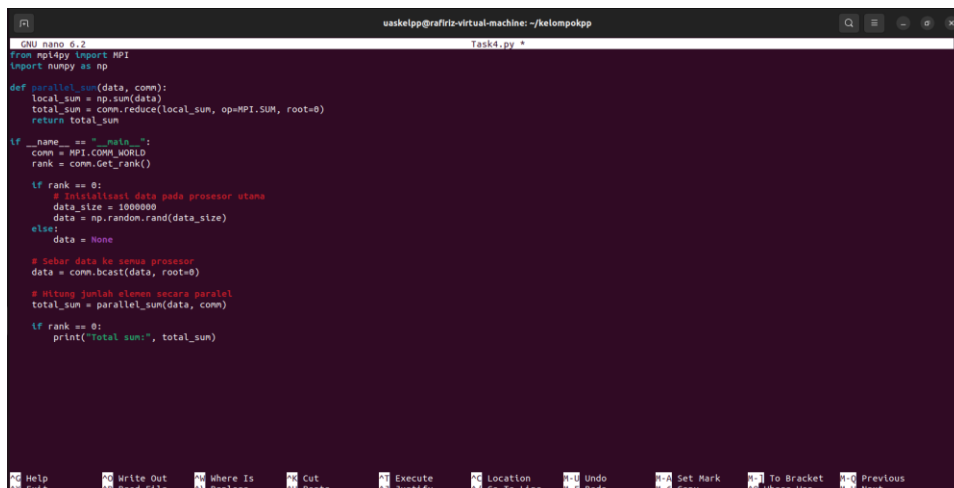
    if rank == 0:
        # Inisialisasi data pada prosesor utama
        data_size = 1000000
        data = np.random.rand(data_size)
    else:
        data = None

    # Sebar data ke semua prosesor
    data = comm.bcast(data, root=0)

    # Hitung jumlah elemen secara paralel
    total_sum = parallel_sum(data, comm)

    if rank == 0:
        print("Total sum:", total_sum)
```

Setelah itu eksekusi kode tersebut kedalam mpi4python seperti dibawah ini.



```
GNU nano 0.2 Task4.py
from mpi4py import MPI
import numpy as np

def parallel_sum(data, comm):
    local_sum = np.sum(data)
    total_sum = comm.reduce(local_sum, op=MPI.SUM, root=0)
    return total_sum

if __name__ == "__main__":
    comm = MPI.COMM_WORLD
    rank = comm.Get_rank()

    if rank == 0:
        # Inisialisasi data pada prosesor utama
        data_size = 1000000
        data = np.random.rand(data_size)
    else:
        data = None

    # Sebar data ke semua prosesor
    data = comm.bcast(data, root=0)

    # Hitung jumlah elemen secara paralel
    total_sum = parallel_sum(data, comm)

    if rank == 0:
        print("Total sum:", total_sum)
```

Dan kemudian dapat dijalankan dengan hasil output seperti dibawah ini.

```
uaskelpp@arafiriz-virtual-machine:~/kelompokpp$ mpirun -np 2 -host master,worker1 python3 Task4.py  
Total sum: 1000158.3241263614
```

Penjelasan :

Program ini adalah contoh sederhana dari program Python yang menggunakan MPI (Message Passing Interface) dan NumPy untuk menghitung jumlah elemen dalam suatu array secara paralel. Berikut adalah penjelasan ringkas dari setiap bagian kode:

- Import Library: Program mengimpor modul MPI dari pustaka mpi4py untuk mendukung operasi komunikasi dan paralelisme menggunakan MPI. Selain itu, NumPy digunakan untuk operasi numerik.
- Fungsi parallel_sum: Fungsi ini menghitung jumlah elemen dalam array secara lokal pada setiap prosesor, kemudian mengumpulkan hasilnya ke prosesor utama dan menghitung jumlah total. Fungsi ini menggunakan fungsi np.sum dari NumPy untuk menghitung jumlah lokal dan fungsi comm.reduce dari MPI untuk mengumpulkan jumlah lokal menjadi jumlah total.
- Blokkode Utama (__main__): Program ini memulai eksekusi dari sini. Inisialisasi komunikator MPI (comm) dan mendapatkan nomor identitas prosesor saat ini (rank).
- Inisialisasi Data: Prosesor utama (dengan rank 0) menginisialisasi array data dengan satu juta elemen acak menggunakan NumPy.
- Sebar Data: Prosesor utama menggunakan fungsi comm.bcast dari MPI untuk mendistribusikan array data ke semua prosesor.
- Hitung Jumlah Elemen Secara Paralel: Semua prosesor, termasuk prosesor utama, memanggil fungsi parallel_sum untuk menghitung jumlah elemen array secara paralel.
- 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.