LAPORAN PEMROSESAN PARALEL

"MPI"

Disusun oleh :

NAMA   :   CATRIN MONICA

NIM   :   09011182126012

KELAS   :   SK5B – INDRALAYA

DOSEN PENGAMPUH   :   ADI HERMANSYAH, S.KOM., M.T.

AHMAD HERYANTO. S.KOM., MT

FAKULTAS ILMU KOMPUTER

JURUSAN SISTEM KOMPUTER

UNIVERSITAS SRIWIJAYA

2023/2024

1. Konfigurasi Hosts

MASTER

WORKER

Pada konfigurasi worker cukup isi dengan ip master dengan ip worker itu sendiri

2. Create User MPI

MASTER & WORKER

```
sudo adduser mpi
```

3. Kasih Akses Root ke User

MASTER & WORKER

```
sudo usermod -aG sudo mpi
```

4. Masuk ke User

MASTER & WORKER

```
su - mpi
```

5. Konfigurasi SSH

MASTER & WORKER

Sebelum melakukan konfigurasi SSH, install openssh-server terlebih dahulu

```
sudo apt install openssh-server
```

Untuk melakukan pengecekan SSH, lakukan command berikut.

```
MASTER : ssh mpi@worker
WORKER  : ssh mpi@master
```

Jika telah berganti user maka ssh telah tersambung. Untuk kembali ke user awal cukup lakukan perintah "exit".

6. Generate Keygen

MASTER

```
ssh-keygen -t rsa
```

7. Copy Keygen ke Worker

MASTER

```
cd .ssh
cat id_rsa.pub | ssh mpi@worker "mkdir .ssh; cat >>
                .ssh/authorized_keys"
```

8. Create Shared Folder

MASTER & WORKER

```
cd mkdir
    cloud
```
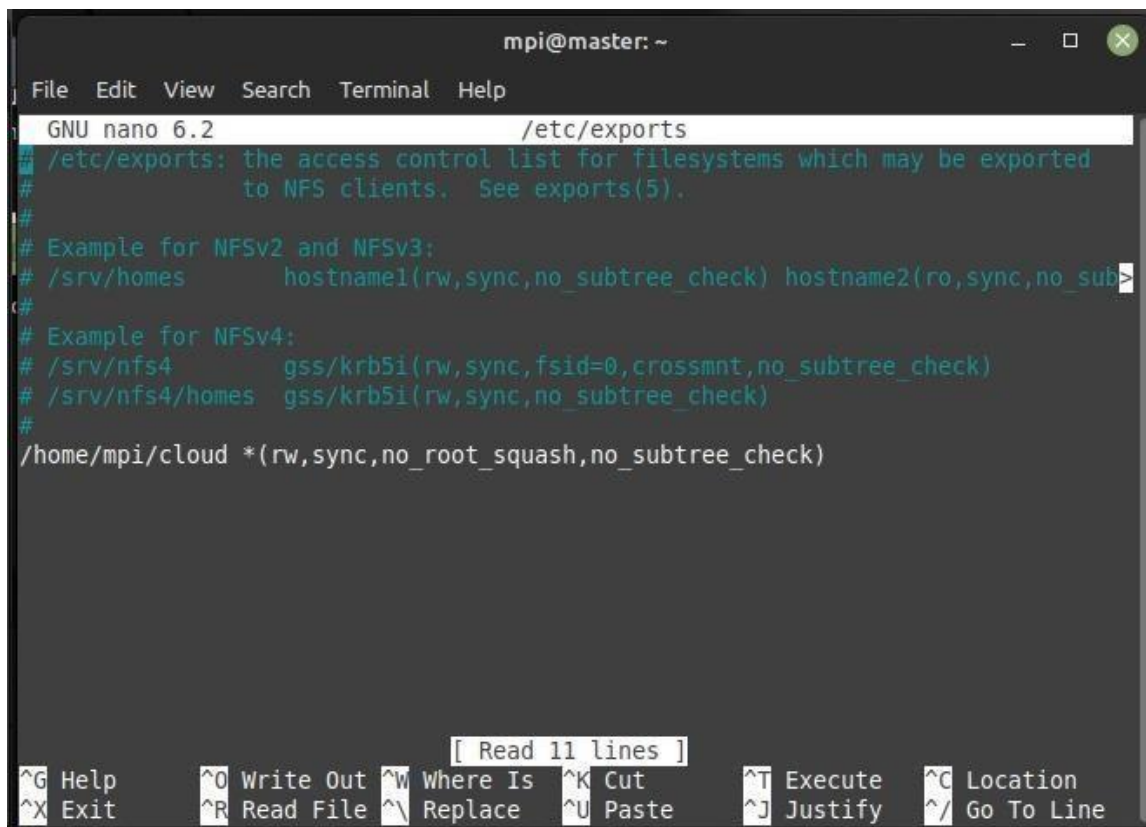
9. Konfigurasi NFS

MASTER

Lakukan installasi NFS Server terlebih dahulu

```
sudo apt install nfs-kernel-server
```

Kemudian tambahkan "/home/mpi/cloud *(rw,sync,no_root_squash,no_subtree_check)" pada file "/etc/exports"

Kemudian lakukan export dan restart nfs

```
sudo exportfs -a
sudo systemctl restart nfs-kernel-server
```

10. Konfigurasi NFS Client

WORKER

```
sudo apt install nfs-common
```

11. Mounting

WORKER

```
sudo mount master:/home/mpi/cloud /home/mpi/cloud
```


12. Install MPI

MASTER & WORKER

```
sudo apt install openmpi-bin libopenmpi-dev
```

13. Bubble Sort

MASTER

```python
from mpi4py import MPI
 def
bubble_sort_parallel(dat
a):
    comm = MPI.COMM_WORLD
rank =
comm.Get_rank()size
= comm.Get_size()
        local_dat
a =
data[rank::size]
local_data.sort()
        for step in range(1,
size):                                          else
        if rank % 2 == 0:
            if rank < size - 1:
                comm.send(local_data,
root=0)      if rank == 0:
        sorted_data =
merge_sorted_arrays(sorted_data)
                returnsorted data
```

```python
        return None
 def merge(arr1, arr2):        merged_array
= []      i = j = 0      while i <
len(arr1) and j < len(arr2):          if
arr1[i] < arr2[j]:

merged_array.append(arr1[i])
i += 1          else:

merged_array.append(arr2[j])
j += 1
merged_array.extend(arr1[i:])
merged_array.extend(arr2[j:])
return merged_array
 def
merge_sorted_arrays(arrays):
    merged_array = []      for array in arrays:
merged_array = merge(merged_array, array)
return merged_array
 if _name_ ==
"_main_":
    data = [5, 2, 9, 1, 5, 6]
comm = MPI.COMM_WORLD
rank = comm.Get_rank()
        if rank
== 0:
        sorted_data =
bubble_sort_parallel(data)
print("Sorted Data:", sorted_data)      else:
        bubble_sort_parallel(data)
```

```
mpi@master:~/cloud$ python3 bubblesort.py
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Data:  [5, 2, 9, 1, 5, 6]
Sorted Data: [1, 2, 5, 5, 6, 9]
Execution Time:  0.0001556873321533203 s
mpi@master:~/cloud$ mpirun -n 1 python3 bubblesort.py
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Data:  [5, 2, 9, 1, 5, 6]
Sorted Data: [1, 2, 5, 5, 6, 9]
Execution Time:  0.00011324882507324219 s
mpi@master:~/cloud$
```

Untuk waktu eksekusi MPI lebih cepat 0.00004243850708007811 dari eksekusi python direct.

14. Numerik

MASTER

```python
from mpi4py import MPI
import time
 start =
time.time()
def
main():
    comm = MPI.COMM_WORLD
rank = comm.Get_rank()
size = comm.Get_size()

    # Data yang akan dihitung     data =
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

    # Bagi data di antara proses
chunk_size = len(data) // size
start = rank * chunk_size     end
= (rank + 1) * chunk_size
    if rank == size -
1:
        # Pastikan semua data terhitung jika panjang data tidak habis
dibagi oleh jumlah proses          end = len(data)
    local_sum =
sum(data[start:end])

    # Kumpulkan hasil dari semua proses     total_sum =
comm.reduce(local_sum, op=MPI.SUM, root=0)
    if rank ==
0:
        print("Total hasil perhitungan:", total_sum)
 if _name_ ==
'_main_':
    main() end = time.time()
print("waktu dikerjakan", end-
start)
```

```
mpi@master:~/cloud$ python3 numerik.py
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Total hasil perhitungan: 55
waktu dikerjakan 0.0003948211669921875
mpi@master:~/cloud$ mpirun -n 1 python3 numerik.py
Authorization required, but no authorization protocol specified
Authorization required, but no authorization protocol specified
Total hasil perhitungan: 55
waktu dikerjakan 0.00038170814514160156
mpi@master:~/cloud$
```

Untuk waktu eksekusi MPI lebih cepat 0.00001311302185058594 dari eksekusi python direct.