**PEMROSESAN PARALEL**

**“Ubuntu Web Server”**



**Disusun oleh :**

Nama : Ahmad Gimnastiar

NIM : 09011282126050

Kelas : SK5B Indralaya

Dosen Pengampuh : Adi Hermansyah, S.Kom., M.T

**FAKULTAS ILMU KOMPUTER**

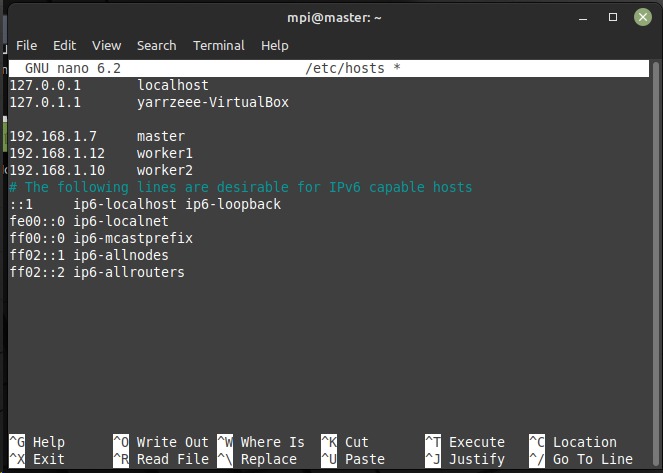
**JURUSAN SISTEM KOMPUTER**

**UNIVERSITAS SRIWIJAYA**

**TAHUN 2023/2024**

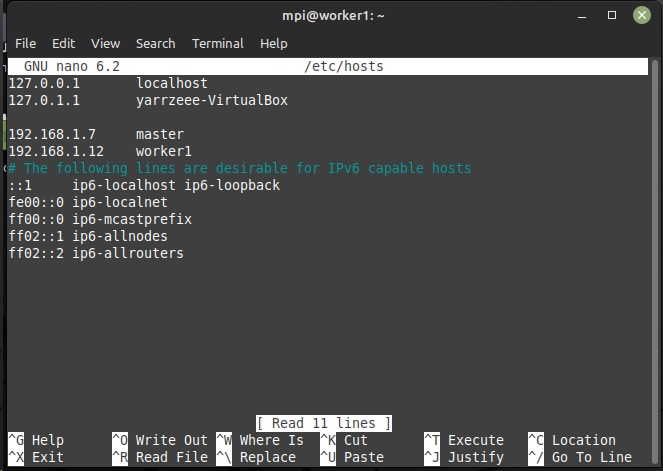
**1. Konfigurasi Hosts**

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

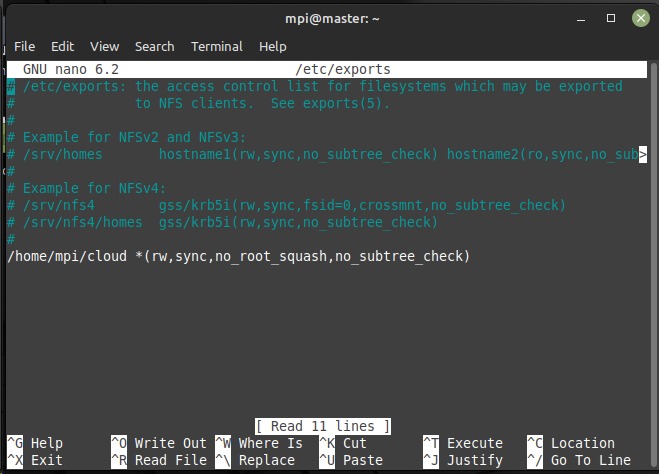
**9. Konfigurasi NFS**

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**

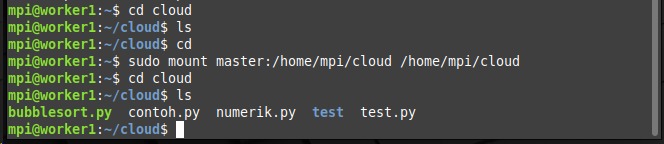
WORKER

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

from mpi4py import MPI

def bubble\_sort\_parallel(data):

    comm = MPI.COMM\_WORLD

    rank = comm.Get\_rank()

    size = comm.Get\_size()

    local\_data = data[rank::size]

    local\_data.sort()

    for step in range(1, size):

        if rank % 2 == 0:

            if rank < size - 1:

                comm.send(local\_data, dest=rank+1)

                received\_data = comm.recv(source=rank+1)

                local\_data = merge(local\_data, received\_data)

        else:

            comm.send(local\_data, dest=rank-1)

            received\_data = comm.recv(source=rank-1)

            local\_data = merge(local\_data, received\_data)

    sorted\_data = comm.gather(local\_data, root=0)

    if rank == 0:

        sorted\_data = merge\_sorted\_arrays(sorted\_data)

        return sorted\_data

    else:

        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()

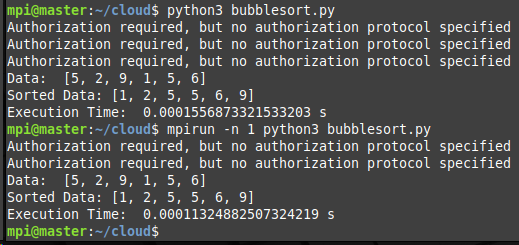
    if rank == 0:

        sorted\_data = bubble\_sort\_parallel(data)

        print("Sorted Data:", sorted\_data)

    else:

        bubble\_sort\_parallel(data)



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

**14. Numerik**

MASTER

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:

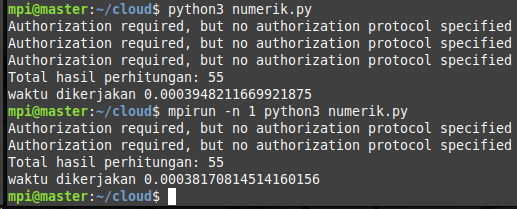
        print("Total hasil perhitungan:", total\_sum)

if \_name\_ == '\_main\_':

    main()

end = time.time()

print("waktu dikerjakan", end-start)



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