

LAPORAN RESMI
PRAKTIKUM ORGANISASI DAN ARSITEKTUR KOMPUTER



JUDUL:

PARALLEL COMPUTING MENGGUNAKAN MPI PYTHON

Disusun Oleh :

TANGGAL PRAKTIKUM	: 15 November 2023
NAMA	: Dinda Rintie Rose
NIM	: 09030582226002
KELAS	: TK3B
DOSEN PENGAMPU	: Adi Hermansyah, M.T

LABORATORIUM JARINGAN KOMPUTER
FAKULTAS ILMU KOMPUTER
UNIVERSITAS SRWIJAYA
PALEMBANG 2023

PRAKTIKUM I

I. JUDUL PRAKTIKUM

Parallel Computing menggunakan MPI Python

II. TUJUAN PRAKTIKUM

1. Remote server Master dengan Putty
2. Mampu menggunakan Parallel Komputing dengan 4 Komputer sebagai ‘Slave’(penyedia resources) dan 1 Komputer sebagai Master
3. Menjalankan 1 Program pada Komputer yang berperan sebagai ‘Master’

III. ALAT

1. Virtual Box
2. Desktop Ubuntu
3. Akses Internet

IV. TEORI DASAR

Parallel Computing menggunakan MPI (Message Passing Interface) adalah teknik yang memungkinkan banyak prosesor bekerja bersama untuk menyelesaikan satu masalah, dengan membagi tugas menjadi sub-tugas yang lebih kecil dan mendistribusikannya ke berbagai prosesor¹²³⁴. MPI adalah standar untuk perpustakaan pengiriman pesan untuk program paralel³. MPI-1 standar dirilis pada tahun 1994 dan standar terbaru adalah MPI-3.13. Python, melalui modul mpi4py, menyediakan antarmuka ke MPI¹²³⁴. Modul ini didasarkan pada binding C++ MPI-2 dan hampir semua panggilan MPI didukung³. Operasi utamanya adalah metode pada objek komunikator¹²³. Modul ini mendukung komunikasi objek Python yang dapat di-pickle dan komunikasi yang dioptimalkan dari array NumPy³.

Untuk melakukan komputasi paralel, Anda tidak menggunakan beberapa thread: gunakan beberapa proses³. Modul multiprocessing memberikan API yang sangat mirip dengan modul threading yang mendukung komputasi paralel³. Ada banyak modul Python lainnya yang tersedia yang mendukung komputasi paralel³. Program Python yang menggunakan perintah MPI harus dijalankan menggunakan interpreter MPI, yang disediakan dengan perintah mpirun³⁴. Pada beberapa sistem, perintah ini disebut mpiexec dan mpi4py tampaknya mencakup keduanya³⁴. Anda dapat menjalankan skrip Python MPI menggunakan perintah mpirun sebagai berikut: mpirun -n 4 python script.py³⁴. Di sini -n 4 memberi tahu MPI untuk menggunakan empat proses³⁴. Dalam contoh mpi4py ini, setiap pekerja menampilkan peringkatnya dan ukuran dunia³⁴:

Kode :

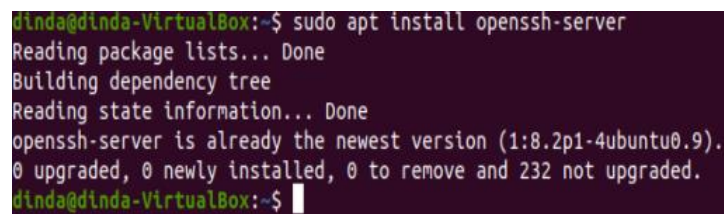
```
from mpi4py import MPI
comm = MPI.COMM_WORLD
print("%d of %d" % (comm.Get_rank(), comm.Get_size()))
```

Anda dapat menggunakan mpirun dan python untuk menjalankan skrip ini: mpirun -n 4 python script.py

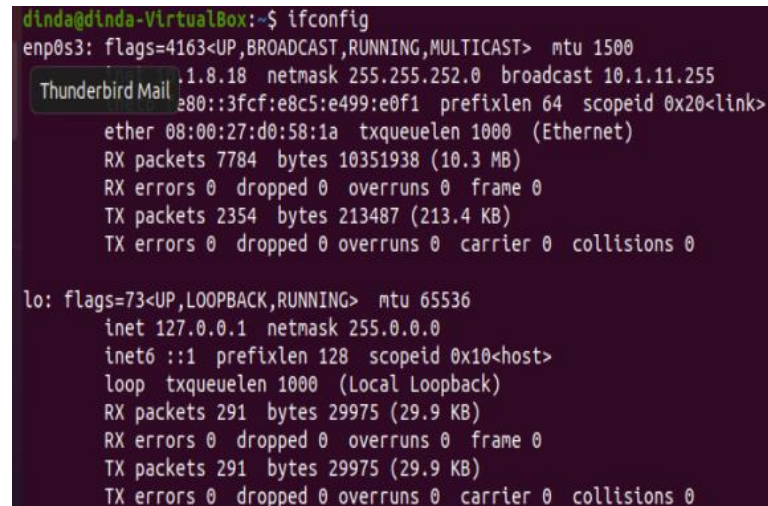
V. PROSEDUR PERCOBAAN

A. Remote Server Master dengan Putty

- **Aktifkan SSH dan cek Alamat IP di Ubuntu**



```
dinda@dinda-VirtualBox:~$ sudo apt install openssh-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
openssh-server is already the newest version (1:8.2p1-4ubuntu0.9).
0 upgraded, 0 newly installed, 0 to remove and 232 not upgraded.
dinda@dinda-VirtualBox:~$
```



```
dinda@dinda-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    ether 08:00:27:d0:58:1a txqueuelen 1000 (Ethernet)
    RX packets 7784 bytes 10351938 (10.3 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2354 bytes 213487 (213.4 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 291 bytes 29975 (29.9 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 291 bytes 29975 (29.9 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- **Setelah selesai silahkan lanjut untuk mengkonfigurasi antara slave dengan master**

```
dinda@dinda-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dinda/.ssh/id_rsa):
/home/dinda/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dinda/.ssh/id_rsa
Your public key has been saved in /home/dinda/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:WMMi4GcYVLQPmTIUybvWvykDQ4eXv/d//9h612+2a7M dinda@dinda-VirtualBox
The key's randomart image is:
+---[RSA 3072]-----+
|
| 0*=0
| oo+ +
| =00.. +
| Help =++ = .
| . = .S
| = . .
| . 0 . . .
| 0 0.. =B|
| 000 ...=E#|
+-----[SHA256]-----+
dinda@dinda-VirtualBox:~$
```

B. Menghubungkan 1 Komputer(Master) ke 4 komputer lain(Slave)

- Disetiap Menjalankan 'ssh-keygen -t rsa' di terminal ubuntu
- Setelah itu tambahkan hasil generate kegen kamu dari 'master' dengan command 'ssh-copy-id <username>@<alamat_ip_master>' di terminal ubuntu

```
dinda@dinda-VirtualBox:~$ ssh-copy-id aidil@192.168.225.243
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter
out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are promp
ted now it is to install the new keys
aidil@192.168.225.243's password:

Number of key(s) added: 1

Now try logging into the machine, with: "ssh 'aidil@192.168.225.243'"
and check to make sure that only the key(s) you wanted were added.
Show Applications
dinda@dinda-VirtualBox:~$
```

VI. KESIMPULAN

Dalam mengatasi tantangan menyambungkan dua mesin virtual Ubuntu sebagai "Master" dan "Slave" untuk menjalankan perintah MPI, kami mulai dengan langkah-langkah persiapan, termasuk instalasi VirtualBox, Ubuntu, dan MPI di semua mesin. Setelah itu, kami memastikan kunci SSH tanpa passphrase dihasilkan di mesin "Master" dan disalin ke "Slave", memastikan perizinan yang benar, dan memformat hostfile. Kendala muncul saat menjalankan perintah MPI, dan langkah-langkah pemecahan masalah melibatkan penyesuaian hostfile, periksa kembali kunci SSH, dan pastikan konfigurasi SSH di mesin "Slave" diatur dengan benar. Meskipun terjadi kesalahan otentikasi SSH, solusi disusun dengan cermat, termasuk perbaikan hostfile, perizinan file, dan periksa log SSH di mesin "Slave". Keseluruhan proses ini memberikan pemahaman mendalam tentang konfigurasi MPI dan manajemen otentikasi untuk menjalankan aplikasi terdistribusi secara efektif di lingkungan virtual.

Dalam perjalanan ini, penting untuk mencatat bahwa menangani kesalahan otentikasi SSH memerlukan pemahaman mendalam tentang konsep-konsep dasar MPI, VirtualBox, dan manajemen kunci SSH. Proses ini juga menekankan pentingnya format yang benar pada hostfile, dan kesiapan setiap mesin virtual untuk berkomunikasi melalui jaringan. Solusi yang telah diberikan memerlukan penggalian detail dalam konfigurasi sistem dan ketersediaan sumber daya pada setiap mesin. Meskipun dapat menantang, perjalanan ini memberikan wawasan yang berharga dalam menangani konfigurasi sistem yang kompleks dan membangun lingkungan terdistribusi yang dapat diandalkan untuk aplikasi MPI. Selanjutnya, pemahaman ini dapat diaplikasikan untuk skenario yang lebih luas dalam pengembangan aplikasi terdistribusi dan pengolahan paralel.