LAPORAN PRAKTIKUM 5

**“Parallel Computing menggunakan MPI Python”**



**Nama : Ahmad Aidil Fajri**

**NIM : 0903582226006**

**Prodi : Teknik Komputer**

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

# PROGRAM STUDI TEKNIK KOMPUTER

**UNIVERSITAS SRIWIJAYA**

# 2023

**PRAKTIKUM I**

### JUDUL PRAKTIKUM

Parallel Computing menggunakan MPI Python

### 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’

### ALAT

* 1. Virtual Box
  2. Putty
  3. Akses Internet

### 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:

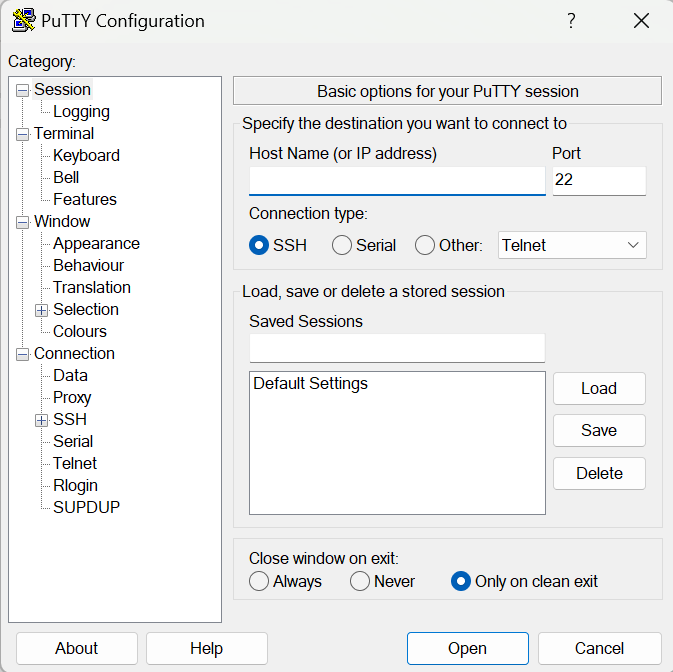
Kode :

from mpi4py import MPIcomm = MPI.COMM\_WORLDprint("%d of %d" % (comm.Get\_rank(), comm.Get\_size()))

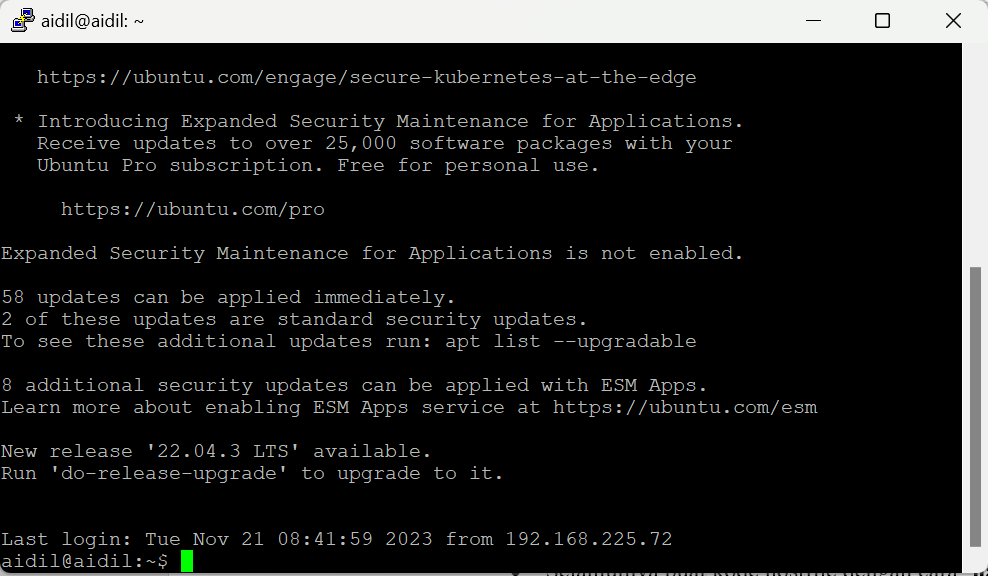
Anda dapat menggunakan mpirun dan python untuk menjalankan skrip ini: mpirun -n 4 python script.py

1. **PROSEDUR PERCOBAAN**
2. **Remote Server Master dengan Putty**

* **Masuk ke Putty lalu masukkan alamat IP kalian ke Putty, pastikan anda memilih ‘ssh’, dan port 22**

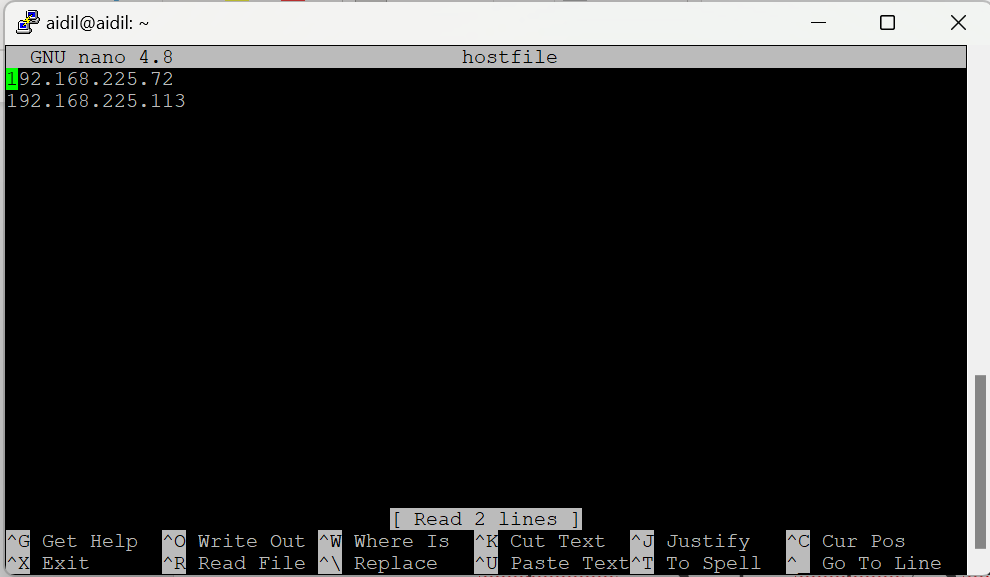
****

* **Setelah selesai silahkan lanjut untuk mengkonfigurasi antara master dengan Slave**

****

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

* Disetiap ‘Slave’ jalankan ‘**ssh-keygen -t rsa’**
* Setelah itu tambahkan hasil generate kegen kamu ke ‘master ’ dengan command ‘**ssh-copy-id <username>@<alamat\_ip\_master>’**
* Selanjutnya buat kode hostfile dengan cara ‘**nano hostfile’**
* Masukkan IP Masing2 ‘Slave’ kedalam nya lalu save



1. **Menjalankan 1 Program pada Komputer yang berperan sebagai ‘Master’**

* Jalankan satu file kode dengan ‘**mpirun -np <jumlah\_slave> -hostfile <file\_host> python cekmpi.py’**

### 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.