Tugas 4 – IF3901

1301154211 (R Rizky Falih)

1301154409 (Naufal Ihsan)

1301150057 (Ridhwan Alifudin)

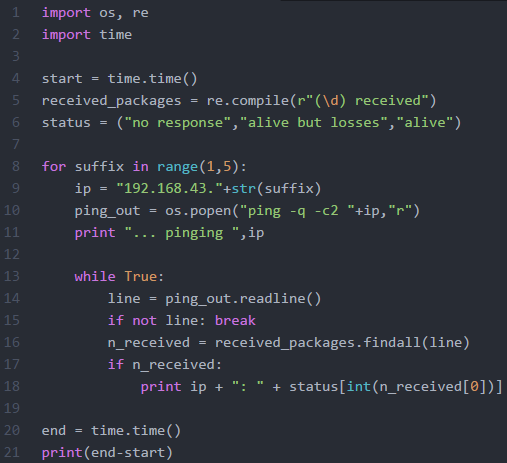
1301154667 (M Turmudzi)

**Thread**

Pada tutorial kali ini, akan diimplementasikan salah satu model pemrograman parallel yaitu *shared memory model*. Thread adalah salah satu implementasi shared memory model. Thread adalah lightweight process. Thread mempunyai overhead yang lebih kecil dibandingkan dengan proses dan lebih ringan.

**Serial Ping**

Untuk memahami thread kita akan melakukan sesuatu hal secara serial kemudian mengubahnya menjadi parallel. Berikut adalah contoh pekerjaan secara serial yaitu melakukan ping ke suatu jaringan untuk mengetahui apakah node hidup atau mati. Pertama kita akan melakukan ping ke sebuah node (192.168.43.1), menunggu hasil ping kemudian menampilkannya. Kita akan mengulangi hal yang sama (ping, tunggu dan menampilkan) untuk node selanjutnya (192.168.43.2). Kita melakukan ping secara serial (satu per satu).



Line 1: import modul os (untuk melakukan perintah ping melalui terminal) dan modul re (untuk melakukan parsing text).

Line 2: import modul time. Modul time akan kita gunakan untuk menghitung lamanya waktu eksekusi

Line 4: variable start menampung waktu dimulainya eksekusi

Line 5: variable received\_packages akan mencari teks yang telah kita tentukan yaitu “(\d) received” .

Jika melakukan ping, hasilnya adalah sebagai berikut. Perhatikan 2 line terbawah. Terdapat kata “ , 2 received, ”. Kata kunci tersebut yang akan kita gunakan untuk menentukan status node yang sedang kita ping.

ping yang menunjukkan node hidup (2 received, maka status akan mengambil indek ke 2 yaitu alive)

$ ping -q -c2 192.168.178.26

PING 192.168.178.26 (192.168.178.26) 56(84) bytes of data.

--- 192.168.178.26 ping statistics ---

2 packets transmitted**, 2 received**, 0% packet loss, time 999ms

rtt min/avg/max/mdev = 0.022/0.032/0.042/0.010 ms

ping yang menunjukkan node mati (0 received, maka status akan mengambil indek ke 0 yaitu no response)

$ ping -q -c2 192.168.178.23

PING 192.168.178.23 (192.168.178.23) 56(84) bytes of data.

--- 192.168.178.23 ping statistics ---

2 packets transmitted, **0 received**, +2 errors, 100% packet loss, time 1006ms

Line 6: terdapat 3 status node (alive, alive but loses, no response). Alive berarti node hidup, aliev but loses berarti ada ping yang hilang, no response berarti node mati/ada firewall

Line 8-11: kita akan melakukan ping dari 192.168.43.1-192.168.43.5. Ping hanya dilakukan 2 kali untuk setiap node (ping -c 2 192.168.43.1 berarti melakukan ping sebanyak 2 kali).

Line 13-18: kita akan mengambil hasil ping (line 14), memfilter informasi yang sesuai (line 16) kemudian menampilkan hasilnya.

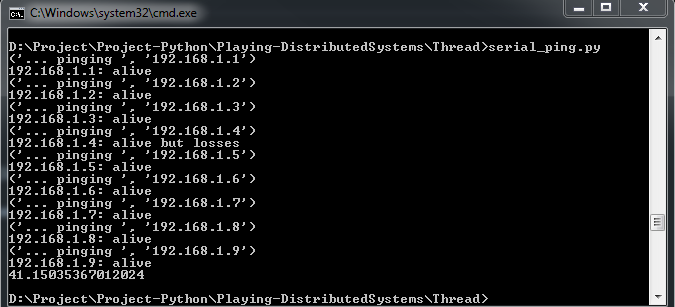
Line 20-21: waktu eksekusi dicatat kemudian ditampilkan waktu akhir-waktu awal untuk mengetahu berapa lama eksekusi proses.

**Tugas A:**

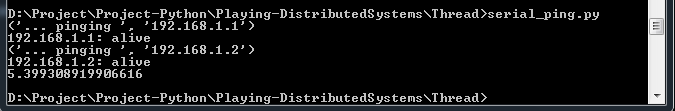
Jalankan program serial\_ping.py (ubah ip yang sesuai) kemudian amati waktu eksekusi jika jumlah node yang diping dibuat menjadi banyak.

**Jawab:**

Ketika jumlah node yang di ping semakin banyak maka waktu eksekusi yang dihasilkan semakin besar/lama. Hal itu dikarenakan proses perpindahan ping dari node sat uke node lainnya memerlukan waktu yang cukup signifikan karena harus mencari jalur koneksi dan mengecek jalur koneksi apakah jalur untuk melakukan ping pada node lain tersedia, tidak tersedia, atau terkoneksi tetapi ditengah jalan hilang ketika node mengirimkan ack sehingga losses.

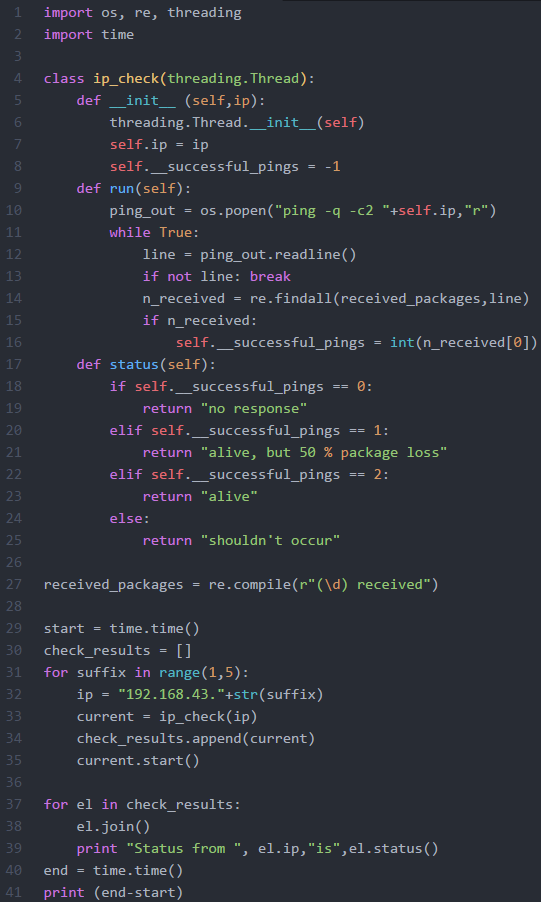


Gambar hasil running ketika melakukan ping pada 9 node.

Gambar hasil running ketika melakukan ping pada 2 node.

Dapat dilihat dari 2 gambar tersebut bahwa waktu running time pada saat melakukan ping. Lebih cepat ketika melakukan ping pada 2 node dibandingkan ketika melakukan ping pada 9 node

**Paralel Ping**



Kita akan mengubah serial ping menjadi parallel ping. Ide dasarnya adalah setiap ping, tunggu hasil dan menampilkan hasil untuk setiap node merupakan sebuah thread. Ping ke 192.168.43.1 merupakan 1 thread, ping ke 192.168.43.2 merupakan 1 thread, dst.

Line 1-2: import modul-modul yang diperlukan. Modul untuk thread adalah threading

Line 4-8: akan dibuat kelas baru bernama ip\_check yang mensupport threading. Threading dilakukan dengan menambahkan fungsi \_\_init\_\_ threading (line 6). Kode sebenarnya yang akan dijalankan ketika thread berjalan berada pada fungsi run().

Line 9-16: otak dari program kita. Pada saat thread berjalan, thread akan melakukan ping sebanyak 2 kali (line 10), menunggu hasilnya (line 12) kemudian menampilkan kondisi node (line 14-16)

Line 17-25: merupakan daftar status untuk setiap node. Perbedaan dengan serial ping adalah status pada serial ping hanya menggunakan 3 status.

Line 27: kata kunci yang ingin kita ketahui

Line 30: kita akan menggunakan array untuk menampung setiap node yang akan kita ping

Line 31-35: kita akan melakukan ping node 192.168.43.1-5, memasukkan hasilnya ke array check\_result, kemudian menjalankan thread masing-masing.

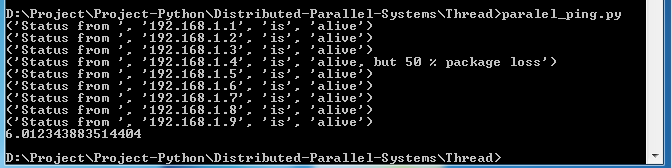
Line 37-39: join digunakan untuk menandakan bahwa thread sudah selesai. Jika thread sudah selesai tampilkan hasil ping.

**Tugas B:**

1. Jalankan script parallel\_ping.py, ubah IP yang sesuai.
2. Berikan observasi waktu yang digunakan untuk menjalankan serial dan parallel ping.

Catatan: program di atas dijalankan pada OS Linux. Untuk windows, parameter ping dan kata kunci yang dicari harus diubah.

**Jawab:**

1. Berikut hasil running ketika script pararel\_ping.py dijalankan  
   
2. Dari hasil running serial\_ping.py dan parallel\_ping.py dapat kita amati bahwa waktu yang dibutuhkan pada parallel\_ping lebih kecil (lebih cepat) dibandingkan waktu ketika melakukan serial\_ping. Hal itu dikarenakan pada pararel\_ping fungsi ping dikerjakan dengan menggunakan threading. Sehingga process yang berjalan tidak dikerjakan secara serial akan tetapi dikerjakan secara pararel yang membuat 1 processor/cpu mengerjakan banyak proses atau banyak cpu mengerjakan banyak proses yang membuat waktu yang dibutuhkan semakin kecil/cepat.

**Lock**

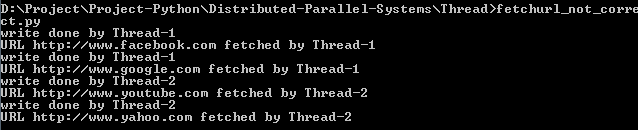
Permasalah utama yang dihadapi thread adalah sinkronisasi. Thread yang mengakses (menulis) ke resource yang sama (file misalnya) dapat menghasilkan output yang tidak konsisten. Thread 1 sedang menulis sesuatu ke dalam file. Thread 2 yang berjalan bersamaan juga menulis hal lain ke file yang sama. Akibatnya hasil akhirnya tidak beraturan dan menghasilkan output yang salah.

Lock adalah salah satu mekanisme sinkronisasi thread (mekanisme lain adalah semaphore, event, rlock). Ide dari lock adalah hanya ada 1 thread yang mengakses resource. Ketika sebuah thread akan mengakses resource (file) maka thread akan meminta (*acquire*) lock. Jika thread tersebut mendapatkan lock maka ia berhak mengakses resource. Thread yang tidak mendapatkan lock tidak bisa mengakses resource. Jika thread tersebut selesai mengakses resource maka ia harus melepaskan lock yang ia dapatkan (*release*). Thread lain kemudian berlomba untuk mendapatkan lock jika ingin mengakses resource.

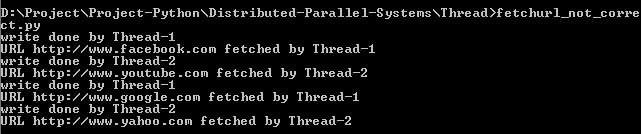
**Tugas C:**

1. Jalankan script fetchurl\_not\_correct.py. Amati apa yang terjadi. Jelaskan mengapa script tersebut tidak benar.

**Jawab:**



Hasil running 1



Hasil running 2

**Analisis:**  Dapat kita lihat bahwa ketika program dijalankan 2x urutan thread berubah-ubah. Hal ini mendakan bahwa script tersebut menjalankan fungsi parallel dan tidak menjalankan fungsi thread yang seharusnya. Dikarenakan tidak adanya proses lock and release. Sehingga ada kemungkinan thread berjalan (pararel) bersamaan atau 1 thread mengakses 2 fertch url secara langsung (tidak bergantiaan)

1. Jalankan script fetchurl\_correct.py. Berikan analisis Anda.

**Jawab:**



Hasil running script fetchurl\_correct.py

**Analisis:** Berbeda dengan script fetchurl\_not\_correct.py ketika script fetchurl\_correct.py dijalankan hasil running akan selalu sama seperti gambar diatas. Hal itu dikarenakan hasil pada script fetchurl\_not\_correct.py menggunakan fitur lock and released pada thredingnya yang dimana tiap thread akan melakukan proses step by step (serial). Misal thread 1 memiliki 2 proses yang dijalankan. Maka untuk menjalankan proses ke 2, thread 1 harus menyelesaikan proses 1 terlebih dahulu (thread di lock/dikunci) dan baru bisa mengakses proses ke 2 ketika proses 1 sudah terselesaikan dan thread sudah di released.

**Tugas PR:**

Buatlah program yang melakukan indexing semua file yang ada di komputer. Terlampir contoh program untuk indexing setiap file dalam computer (dir\_tra.py). Ubahlah script tersebut sehingga berjalan secara parallel.