

A.A. Gede Oka Aryanata (1301140220)

M. Iqbal Maulana (1301144060)

IF3810

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).

```

1  import os, re
2  import time
3
4  start = time.time()
5  received_packages = re.compile(r"(\d) received")
6  status = ("no response", "alive but losses", "alive")
7
8  for suffix in range(1,5):
9      ip = "192.168.43."+str(suffix)
10     ping_out = os.popen("ping -q -c2 "+ip,"r")
11     print "... pinging ",ip
12
13     while True:
14         line = ping_out.readline()
15         if not line: break
16         n_received = received_packages.findall(line)
17         if n_received:
18             print ip + ": " + status[int(n_received[0])]
19
20 end = time.time()
21 print(end-start)

```

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 indeks 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
```

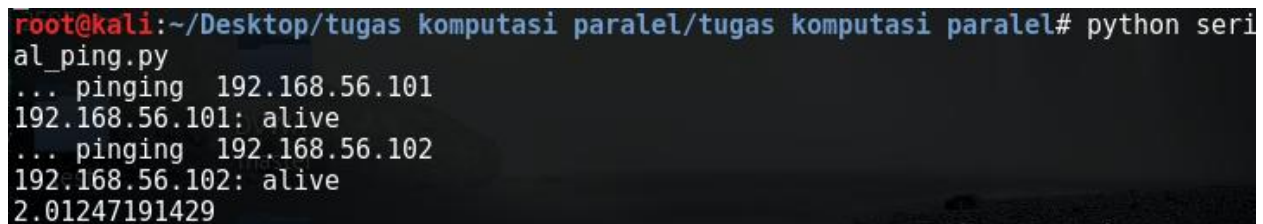
Line 6: terdapat 3 status node (alive, alive but loses, no response). Alive berarti node hidup, alive but loses berarti ada ping yang hilang, no response berarti node mati/ada firewall

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 mengetahui berapa lama eksekusi proses.

Tugas (wajib): jalankan program (ubah ip yang sesuai dengan jaringan anda) kemudian amati waktu eksekusi jika jumlah node yang diping dibuat menjadi banyak. Berikan analisa dari output program tersebut.



```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python serial_ping.py
... pinging 192.168.56.101
192.168.56.101: alive
... pinging 192.168.56.102
192.168.56.102: alive
2.01247191429
```

Figure 1 ping 2 node

Pada gambar diatas dilakukan ping pada 2 buah node yang aktif. Untuk melakukan ping ke dua buah node yang aktif diperlukan waktu kurang lebih 2 detik yang berarti diperlukan waktu satu detik untuk melakukan ping ke masing – masing node yang aktif tersebut karena pada konsep serial ping ping ke node selanjutnya baru bisa dilakukan setelah ping ke node sebelumnya selesai.

```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python serial_ping.py
... pinging 192.168.56.101
192.168.56.101: alive
... pinging 192.168.56.102
192.168.56.102: alive
... pinging 192.168.56.103
192.168.56.103: no response
13.0400619507
```

Figure 2 ping 3 node

Pada gambar diatas dilakukan ping pada 3 buah node yang mana satu node tidak aktif (192.168.56.103). untuk melakukan ping ke tiga buah node tersebut dibutuhkan waktu kurang lebih 13 detik. Dari gambar figure 1 diketahui untuk melakukan ping ke 2 node yang aktif memerlukan waktu sekitar 2 detik, pada figure 2 diketahui setelah ditambahkan satu node yang aktif waktu eksekusi menjadi 13 detik, berarti diperlukan waktu sekitar 11 detik untuk mengetahui apakah node ketiga tersebut aktif atau tidak.

Paralel Ping

```

1  import os, re, threading
2  import time
3
4  class ip_check(threading.Thread):
5      def __init__(self,ip):
6          threading.Thread.__init__(self)
7          self.ip = ip
8          self.__successful_pings = -1
9      def run(self):
10         ping_out = os.popen("ping -q -c2 "+self.ip,"r")
11         while True:
12             line = ping_out.readline()
13             if not line: break
14             n_received = re.findall(received_packages,line)
15             if n_received:
16                 self.__successful_pings = int(n_received[0])
17         def status(self):
18             if self.__successful_pings == 0:
19                 return "no response"
20             elif self.__successful_pings == 1:
21                 return "alive, but 50 % package loss"
22             elif self.__successful_pings == 2:
23                 return "alive"
24             else:
25                 return "shouldn't occur"
26
27     received_packages = re.compile(r"(\d) received")
28
29     start = time.time()
30     check_results = []
31     for suffix in range(1,5):
32         ip = "192.168.43."+str(suffix)
33         current = ip_check(ip)
34         check_results.append(current)
35         current.start()
36
37     for el in check_results:
38         el.join()
39         print "Status from ", el.ip,"is",el.status()
40     end = time.time()
41     print (end-start)

```

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.

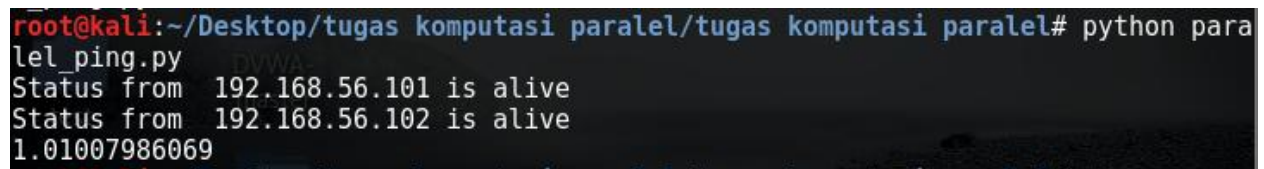
Line 27: kata kunci yang ingin kita ketahui

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.

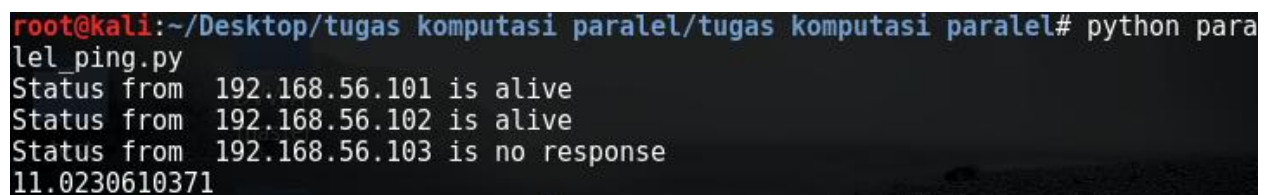
Tugas (wajib): Jalankan script, ubah IP yang sesuai. Berikan observasi waktu yang digunakan untuk menjalankan serial dan parallel ping. Berikan analisa dari output program tersebut.



```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python parallel_ping.py
Status from 192.168.56.101 is alive
Status from 192.168.56.102 is alive
1.01007986069
```

Dari gambar Figure 3 ping paralel 2 node

Dari figure 3 diketahui untuk melakukan ping kedua node yang aktif hanya memerlukan waktu kurang lebih satu detik. Itu karena parallel ping melakukan ping ke dua node tersebut dalam waktu bersamaan (tidak perlu menunggu proses ping pertama selesai).



```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python parallel_ping.py
Status from 192.168.56.101 is alive
Status from 192.168.56.102 is alive
Status from 192.168.56.103 is no response
11.0230610371
```

Figure 4 ping paralel 3 node

Dilakukan ping ke 3 node yang salah satunya mati (192.168.56.103). dari figure 4 diketahui waktu eksekusi program adalah kurang lebih 11 detik. Berdasarkan data dari figure 3, dapat kita ketahui diperlukan waktu kurang lebih 10 detik untuk mengetahui node 3 aktif atau tidak.

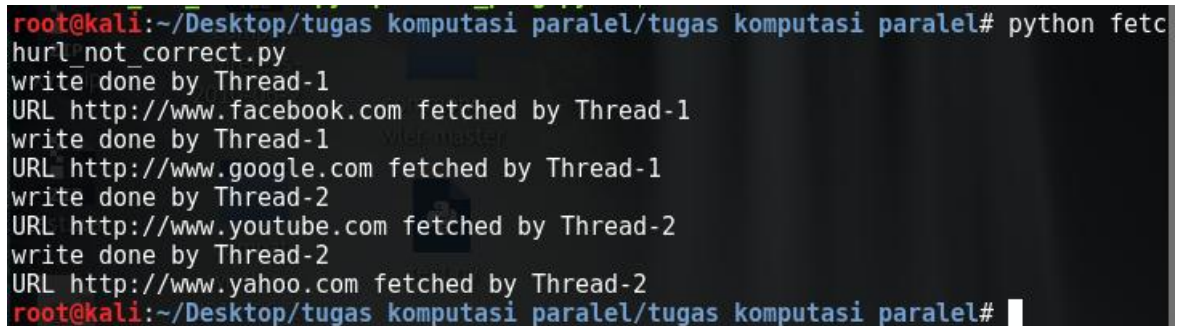
Lock

Permasalahan 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 (wajib):

1. Jalankan script `fetchurl_notcorrect.py`. Amati apa yang terjadi. Jelaskan mengapa script tersebut tidak benar.




```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python fetchurl_not_correct.py
write done by Thread-1
URL http://www.facebook.com fetched by Thread-1
write done by Thread-1
URL http://www.google.com fetched by Thread-1
write done by Thread-2
URL http://www.youtube.com fetched by Thread-2
write done by Thread-2
URL http://www.yahoo.com fetched by Thread-2
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel#
```

Figure 5 hasil running `fetchurl_not_correct.py`

Berdasarkan figure 5 dapat diketahui bahwa tidak ada system lock pada baris program tersebut. Tidak ada penanda bahwa thread mana yang sedang di lock. Dampak dari permasalahan tersebut adalah terdapat lebih dari satu thread yang mengakses resource secara bersamaan dan dapat menghasilkan output yang salah.

Jalankan script `fetchurl_correct.py`. Berikan analisis Anda.



```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python fetchurl_correct.py
lock acquired by Thread-2
write done by Thread-2
lock released by Thread-2
URL http://www.youtube.com fetched by Thread-2
lock acquired by Thread-1
write done by Thread-1
lock released by Thread-1
URL http://www.facebook.com fetched by Thread-1
lock acquired by Thread-1
write done by Thread-1
lock released by Thread-1
URL http://www.google.com fetched by Thread-1
lock acquired by Thread-2
write done by Thread-2
lock released by Thread-2
URL http://www.yahoo.com fetched by Thread-2
```

Figure 6 hasil running `fetchurl_correct.py`

Dari figure 6 dapat diketahui bahwa baris program tersebut telah menerapkan system lock, dimana terlihat bahwa thread mana yang sedang mengakses resource. Dan selama thread tersebut mengakses resource tidak ada thread lain yang dapat mengakses resource tersebut. Serta jika thread tersebut telah selesai menggunakan resource maka lock akan segera dilepaskan.

Tugas:

- **Buat resume tentang threading. (wajib)**

Thread merupakan sebutan dari proses ringan (lightweight) atau Thread adalah unit terkecil dalam suatu proses yang bisa dijadwalkan oleh sistem operasi. Thread merupakan unit dasar dari utilitas CPU. Di dalamnya terdapat ID thread, program counter, register, dan stack. Dan saling berbagi dengan thread lain dalam proses yang sama.

Keuntungan memakai Thread:

- Tanggap: Multi-threading mengizinkan program untuk terus berjalan walaupun pada bagian program tersebut diblock atau sedang dalam keadaan menjalankan operasi yang lama/panjang. Contohnya multithread web browser dapat mengizinkan pengguna berinteraksi dengan suatu thread ketika suatu gambar sedang di-load oleh thread yang lain.
- Pembagian sumber daya: Secara default, thread membagi memori dan sumber daya dari proses. Keuntungan dari pembagian kode adalah aplikasi mempunyai perbedaan aktifitas thread dengan alokasi.
- Ekonomis: Mengalokasikan memori dan sumber daya untuk membuat proses itu sangat mahal. Alternatifnya thread membagi sumber daya dari proses, Jadi lebih ekonomis.
- Pemberdayaan arsitektur multiprosesor: Keuntungan dari multithreading dapat ditingkatkan dengan arsitektur multiprosesor, dimana setiap thread dapat berjalan secara parallel pada prosesor yang berbeda. Pada arsitektur prosesor tunggal, CPU biasanya berpindah-pindah antara setiap thread dengan cepat, sehingga terdapat ilusi paralelisme, tetapi pada kenyataannya hanya satu thread yang berjalan di setiap waktu.

Macam – macam thread :

- Single-Threading adalah sebuah lightweight process (proses sederhana) yang mempunyai thread tunggal yang berfungsi sebagai pengendali/ controller.

- Multi-Threading adalah proses dengan thread yang banyak dan mengerjakan lebih dari satu tugas dalam satu waktu.

Model Multi-Threading :

- Many-to-One : satu kernel thread digunakan oleh banyak user thread
- One-to-One : satu kernel thread digunakan oleh banyak satu thread
- Many-to-Many : banyak kernel thread digunakan oleh banyak user thread

Thread cancellation

Thread cancellation/ pembatalan thread ialah pemberhentian thread sebelum tugasnya selesai. Thread yang akan diberhentikan disebut sebagai target thread. Pembatalan Thread terdiri dari 2 jenis:

1. Asynchronous cancellation: suatu thread seketika itu juga memberhentikan target thread.
2. Deferred cancellation: target thread secara periodik memeriksa apakah dia harus berhenti, cara ini memperbolehkan target thread untuk memberhentikan dirinya sendiri secara teratur.

Sumber : <https://tintakopi.wordpress.com/2011/10/24/thread-multithread/>

- **Buatlah web crawler (spider) sederhana! (bonus)**
- **Tool untuk mendeteksi node dalam satu jaringan (bonus)**

```
root@kali:~/Desktop/tugas komputasi paralel/tugas komputasi paralel# python parallel_ping.py
Status from 192.168.56.1 is alive
Status from 192.168.56.2 is no response
Status from 192.168.56.3 is no response
Status from 192.168.56.4 is no response
Status from 192.168.56.5 is no response
Status from 192.168.56.6 is no response
Status from 192.168.56.7 is no response
Status from 192.168.56.8 is no response
Status from 192.168.56.9 is no response
Status from 192.168.56.10 is no response
Status from 192.168.56.11 is no response
Status from 192.168.56.12 is no response
Status from 192.168.56.13 is no response
Status from 192.168.56.14 is no response
Status from 192.168.56.15 is no response
Status from 192.168.56.16 is no response
```

```
Status from 192.168.56.90 is no response
Status from 192.168.56.91 is no response
Status from 192.168.56.92 is no response
Status from 192.168.56.93 is no response
Status from 192.168.56.94 is no response
Status from 192.168.56.95 is no response
Status from 192.168.56.96 is no response
Status from 192.168.56.97 is no response
Status from 192.168.56.98 is no response
Status from 192.168.56.99 is no response
Status from 192.168.56.100 is no response
Status from 192.168.56.101 is alive
Status from 192.168.56.102 is alive
Status from 192.168.56.103 is no response
Status from 192.168.56.104 is no response
Status from 192.168.56.105 is no response
Status from 192.168.56.106 is no response
Status from 192.168.56.107 is no response
Status from 192.168.56.108 is no response
Status from 192.168.56.109 is no response
Status from 192.168.56.240 is no response
Status from 192.168.56.241 is no response
Status from 192.168.56.242 is no response
Status from 192.168.56.243 is no response
Status from 192.168.56.244 is no response
Status from 192.168.56.245 is no response
Status from 192.168.56.246 is no response
Status from 192.168.56.247 is no response
Status from 192.168.56.248 is no response
Status from 192.168.56.249 is no response
Status from 192.168.56.250 is no response
Status from 192.168.56.251 is no response
Status from 192.168.56.252 is no response
Status from 192.168.56.253 is no response
Status from 192.168.56.254 is no response
12.0414187908
```

Jawaban ditulis dalam format word. Dikerjakan kelompok max 2 orang. Kirim ke email arif.nugroho@telkomuniversity.ac.id subject [PARALEL_THREAD_NIM_NIM] sebelum 12 April 2017 jam 10:00 am.