TUGAS BESAR MATA KULIAH SISTEM PARALEL DAN TERDISTRIBUSI

Program Download Parallel

Oleh :
DIVA ANNISA FEBECCA - 1301204302
HERMAN GEMILANG - 1301204014
JOHANES RAPHAEL NADAPUTRA - 1301204243
MUHAMMAD NAUFAL ABDILLAH - 1301201586
RICARDO HAMONANGAN- 1301204201

Team SeTu
IF-44-01

PROGRAM STUDI INFORMATIKA

FAKULTAS INFORMATIKA

2022

Daftar Isi

BAB 1

PENDAHULUAN

a. Deskripsi Tugas Besar

Topik tugas besar SISTER yang kami pilih adalah mendownload secara paralel, dimana komputasi paralel merupakan penggunaan beberapa sumber daya komputasi secara bersamaan untuk memecahkan masalah komputasi. Masalah tersebut akan dipecah menjadi beberapa bagian diskrit yang dapat diselesaikan secara bersamaan pada prosesor yang berbeda, sehingga waktu yang dibutuhkan untuk mengeksekusi program akan lebih cepat terselesaikan.

b. Tanggung Jawab

Nama	Tanggung Jawab
Diva Annisa Febecca	Laporan
Herman Gemilang	Program, Laporan
Johanes Raphael Nandaputra	Program, Laporan
Muhammad Naufal Abdillah	Program, Laporan
Ricardo Hamonangan	Laporan

BAB 2

ANALISIS

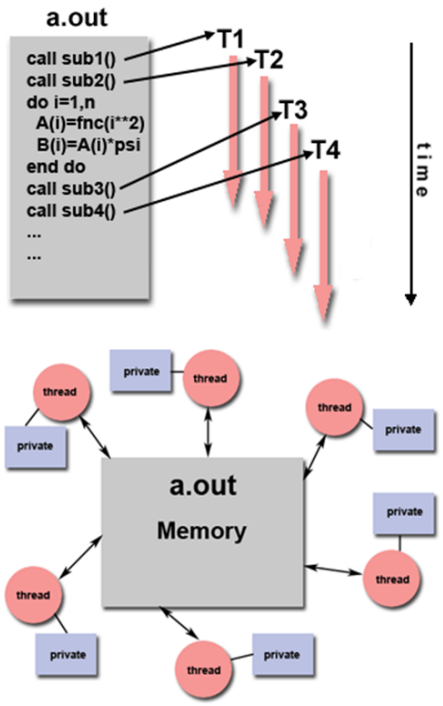
a. Pemilihan Solusi

Untuk proses pengimplementasian proses Download secara paralel yang menjadi topik pada tugas besar kali ini, menggunakan sistem paralel dengan model yang mengimplementasikan *Shared Memory Model* dengan *threads*. Model ini memanfaatkan thread untuk melakukan suatu '*lightweight*' proses yang melaksanakan download yang dibutuhkan. Thread - thread ini akan berjalan secara *concurrently,* sehingga task download akan berjalan hampir bersamaan tiap program dijalankan.

b. Model Sistem

Model sistem paralel yang digunakan adalah *Shared Memory Model* dengan thread, dimana model ini menggunakan konsep *threading* yang membuat suatu task bisa dilakukan secara *concurrently*. Sistem ini akan meminta input yang berbeda - beda, dalam kasus dari tugas besar ini, inputnya adalah URL dari file yang ingin di download, dan kemudian sistem akan mendownload URL itu secara *concurrently*, yang secara teori akan menyebabkan lamanya download lebih cepat ketimbang mendownload secara serial. Dalam sistem ini, setiap thread tidak tersinkronisasi sehingga setiap thread bisa menyelesaikan task mereka tanpa menunggu thread yang lainnya selesai, begitu juga ketika jika ada thread yang error, maka thread yang lainnya tidak terpengaruh.

Gambaran bagaimana model bekerja



BAB 3

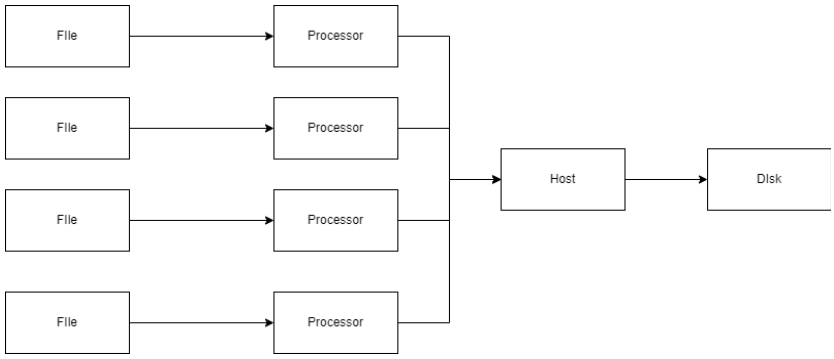
PERANCANGAN

a. Arsitektur Sistem dan Jaringan

- Skema Kerja

  Dalam tugas besar ini, program akan menggunakan konsep arsitektur peer-to-peer, yaitu dimana tidak adanya suatu entitas yang menjadi pusat pada jaringan. Sistem dari program ini akan mengambil suatu file dari satu atau lebih jaringan, kemudian akan mendownloadnya ke local storage.

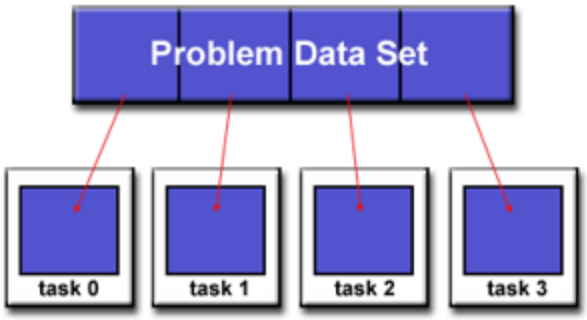
Gambaran Skema Kerja Program



  Permasalahan yang dialami ialah mendownload file, mendownload file dapat di paralelkan karena file tersebut dapat dibagi-bagi menjadi *chunk* yang lebih kecil dan di download secara bersamaan oleh processor yang berbeda. Hal ini dapat mempercepat proses mendownload dibandingkan mendownload secara serial.

- *Problem Partitioning*

  Pada *Problem Partitioning*, akan menggunakan *Decomposition.* Dalam *Decomposition*, program akan memecah masalah ke *chunk* dan mendistribusikannya ke beberapa pekerjaan. Cara *decomposition* yang digunakan adalah *Domain Decomposition*, dimana Problem Data Set pada kasus ini adalah list file yang akan didownload dan akan dikerjakan oleh task-task yang didistribusikan ke setiap data tersebut.



- *Communication*

  Untuk *Communication*, Program yang kami buat tidak membutuhkan *communication* karena tidak perlu membagikan data antar task, dimana hal ini sering disebut dengan *embarrassingly parallel* yang berarti mereka *straight-forward* dan sangat sedikit komunikasi antar *task* yang diperlukan.

- *Synchronization*

  Synchronization merupakan salah satu bagian yang penting dalam program yang berbentuk paralel. Hal ini digunakan untuk memastikan thread atau proses yang bekerja terkontrol dan terkoordinasi. Pada program ini kami menggunakan method '.join()', dimana thread akan memblok main thread(main program) sampai thread yang mengeksekusi .join() selesai. Sebagai tambahan, 'with open' saat menyimpan file ke disk juga memastikan bahwa file sudah tersimpan saat code selesai.

- *Data Dependency*

  Dalam hal ini, program kami tidak membutuhkan *data dependency* karena hasil thread tidak dibutuhkan oleh thread lainnya. Sehingga hasil dari tiap thread yang berupa file akan langsung tersimpan di disk User.

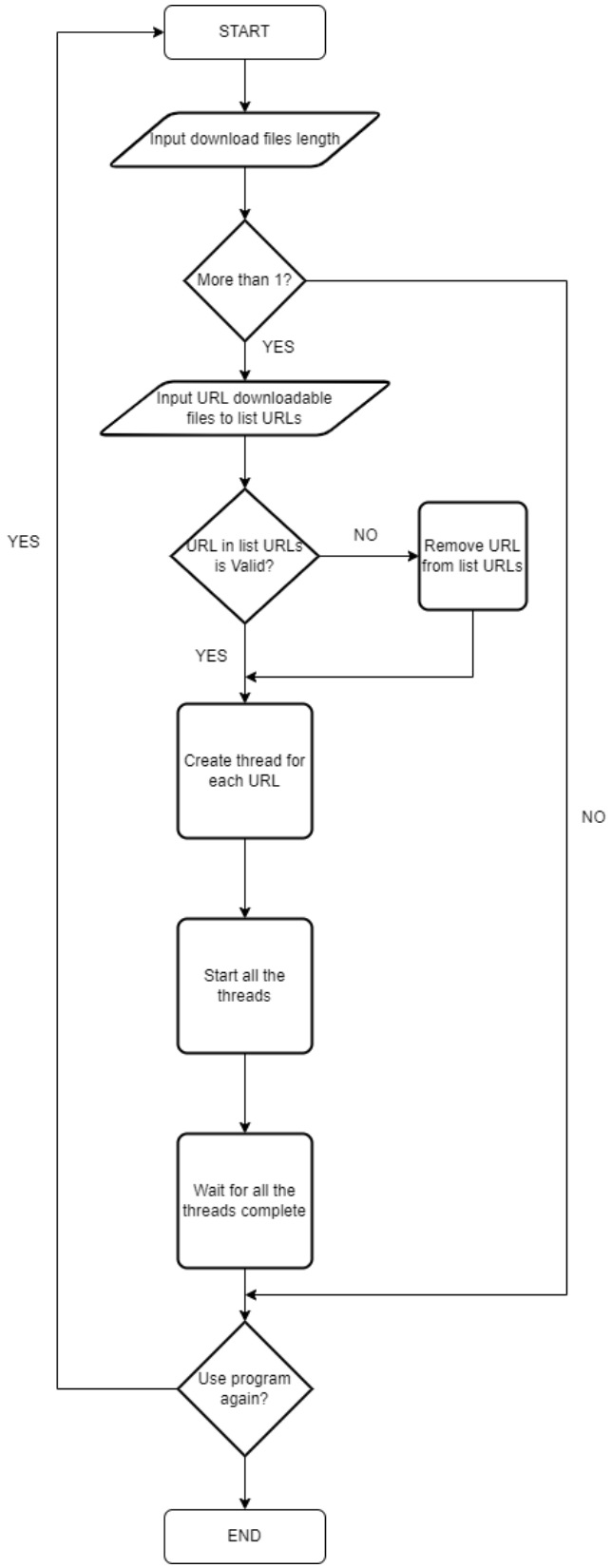
- *Load Balancing*

  Load balancing dalam program bersifat statis, dimana setiap task akan memegang satu thread/proses dari awal sampai akhir task terselesaikan.

- *Granularity*

  Granularity adalah rasio dari komputasi / komunikasi yang dilakukan thread. Pada program ini merupakan Coarse-Grain Parallelism dimana komputasinya lebih tinggi daripada komunikasi yang dilakukan.

b. Alur Proses Aplikasi



Alur dari proses download yang dibuat adalah :

1. User menginput berapa banyak URL dari file yang akan didownload.
2. Sistem melakukan pengecekan apabila banyak file yang akan didownload lebih dari 1 maka akan lanjut ke flow berikutnya (no. 4).
3. Jika tidak, maka akan diberi pemberitahuan bahwa minimal mendownload 2 file dan diminta untuk menjalankan program kembali.
4. User lalu menginput URL dari banyak file yang ingin didownload kemudian akan tersimpan di list URLs.
5. Sistem akan melakukan iterasi pengecekan di list URLs, jika semua URL yang dimasukkan oleh User valid dan dapat didownload maka akan lanjut ke flow berikutnya (no. 7).
6. Jika terdapat beberapa URL yang tidak valid untuk dapat didownload, maka URL tersebut akan dihapus dari list URLs dan menyisakan URL-URL yang dapat didownload saja.
7. Dari banyaknya URL valid yang terdapat dalam list URLs, akan dibuat thread yang ditugaskan untuk tiap URL tersebut (masuk ke born state).
8. Kemudian, thread-thread tersebut akan di start untuk menjalankan tugasnya masing-masing (masuk ke ready state).
9. Semua thread tersebut akan berjalan secara konkuren/paralel dan ketika tugas mendownload tiap thread selesai maka akan langsung tersimpan di disk User.
10. Terakhir, akan bertanya jika ingin menggunakan program kembali, jika tidak maka program berakhir.

BAB 4

IMPLEMENTASI PROGRAM

a. Kode Program

Program dibagi menjadi beberapa bagian dengan masing - masing fungsi yang sudah ditetapkan. Untuk program ini, ada dua macam cara downloading untuk digunakan sebagai pembanding, pertama download dengan cara serial(sequential), dan kedua download secara paralel, topik dari tugas besar ini.

Bahasa pemrograman yang digunakan adalah Python, dengan menggunakan beberapa package yang tersedia. Package - package yang digunakan adalah:

- threading
- requests
- Time
- validators

Berikut merupakan kode dan penjelasannya.

```python
def get_url(n):
  # declare lists
  urls = []

  # get url from user and save it to lists
  for i in range(n):
    url = input(f"Enter Downloadable URL-{i+1}: ")
    urls.append(url)
  print()
  # return lists
  return urls
```

Fungsi dari function get_url() adalah untuk mengambil URL - URL berdasarkan inputan user. User pertama akan menginput seberapa banyak url yang akan ia masuki, kemudian program akan meminta url - url tersebut dengan looping, yang kemudian URL nya akan di append kedalam suatu list. Function akan me-return list yang sudah terbentuk.

```
def check_url(url):
    cek = validators.url(url)
    return cek
```

Pada function check_url(), akan dilakukan pengecekan apakah string yang dimasukkan user benar - benar URL apa tidak. Function akan mengecek apakah URL yang di input user mempunyai sebuah content/data yang bisa didownload.

```
def download_url(url):
    # start time
    t0 = time.time()

    # Send request and download the file
    response = requests.get(url)
    data = response.content

    # Save the file to disk
    filename = url.split('/')[-1]
    with open(filename, 'wb') as f:
      f.write(data)

    # return file and time for measuring
    print(f"File: {filename}, Time: {time.time() - t0} ")
    return filename, time.time() - t0
```

Pada function download_url(), program akan memintas request dari url yang sudah diinputkan oleh user sebelumnya, kemudian mendownload datanya sesuai dengan nama yang ada di link URL tersebut. Setelah di download, function akan memprint nama file dan waktu yang telah berlalu ketika mendownload datanya, kemudian akan me-return hal yang sama.

```python
def download_parallel(urls):
  # start time for parallel download
  t0 = time.time()

  # create thread for each URL
  threads = [threading.Thread(target=download_url, args=(url,)) for url in urls]

  for url in urls:
    print(f"Downloading {url.split('/')[-1]}")
  print()
  # start all the threads
  for thread in threads:
    thread.start()

  # wait for all threads to complete
  for thread in threads:
    thread.join()

  # output total time for parallel download
  print(f"Total Time Parallel Download: {time.time() - t0} seconds")
```

Function download_parallel() adalah inti dari program yang telah dikerjakan. Function akan meminta list yang berisi url-url yang ingin di download, kemudian, program akan membuat thread untuk tiap url, agar dapat melakukan task mendownload secara paralel. Setelah itu, thread - thread yang sudah dibentuk di dalam list tersebut akan di start dengan loop, dan agar memastikan thread selesai semua sebelum main program selesai, akan digunakan .join() yang akan memblok jalannya main program sampai thread selesai.

```python
# MAIN PROGRAM COMBINED
if __name__ == "__main__":
    print("========WELCOME TO PARALLEL DOWNLOADING PROGRAM========")
    while True:
        print("This is a simple program to download files using threads. \nDownload will run concurently thus
        quickening the process of downloading multiple files.")
        print()
        # get user input for files length
        n = int(input("How many files you want to download? "))

        # check files length
        if n < 2:
            print("Sorry, please download atleast 2 files")
        else:
            # get url from user
            urls = get_url(n)

            # check if url is downloadable, remove it if not valid/downloadble
            for url in urls[:]:
                if check_url(url):
                    continue
                else:
                    print(f"{url} is not a downloadable URL")
                    urls.remove(url)

            # check if lists URLs is empty
            if len(urls) > 0:
                download_parallel(urls)
            else:
                print("Sorry, please re-run and enter a valid URL to download")

        print()
        ask = input("Do you want to use the program again?(y/n): ")

        if ask == "y":
            print()
            continue
        else:
            break
```

Untuk main program, program ini akan ditujukan langsung untuk melakukan proses download paralel menggunakan threads. User akan diminta untuk menginputkan berapa url yang ingin dimasukkan, dan program akan meminta url sesuai dengan inputan user.

```
========WELCOME TO THE PARALLEL DOWNLOADING PROGRAM========
This is a simple program to download files using threads.
Download will run concurently thus quickening the process of downloading multiple files.

How many files you want to download?
```

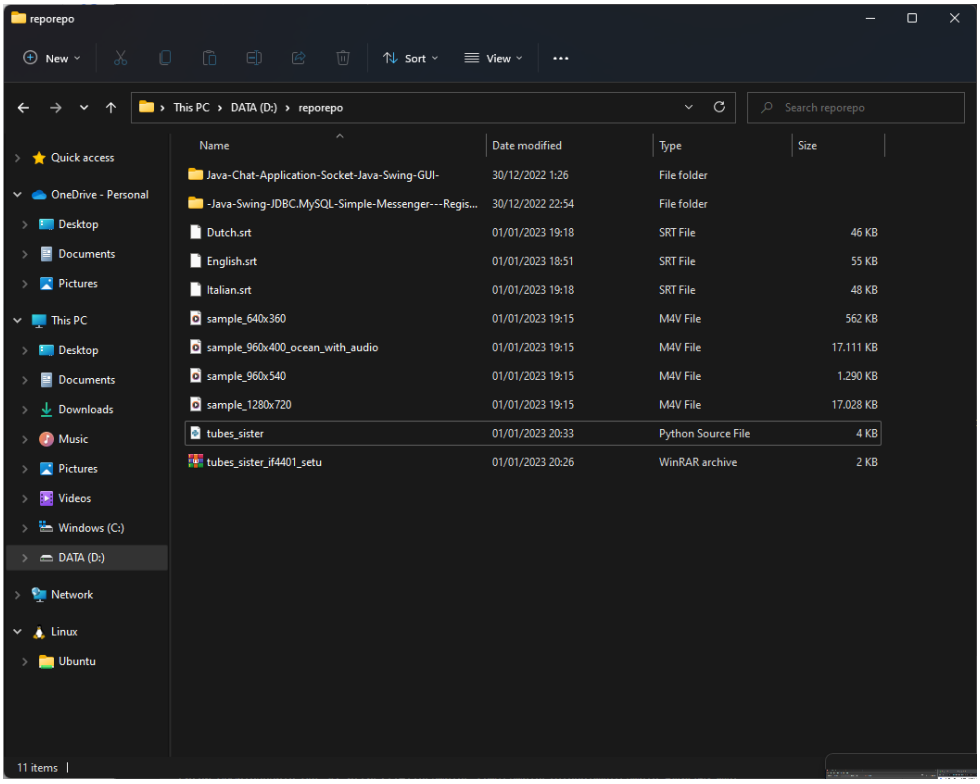
```
How many files you want to download? 4
Enter Downloadable URL-1: https://filesamples.com/samples/video/m4v/sample_960x400_ocean_with_audio.m4v
Enter Downloadable URL-2: https://filesamples.com/samples/video/m4v/sample_960x540.m4v
Enter Downloadable URL-3: https://filesamples.com/samples/video/m4v/sample_1280x720.m4v
Enter Downloadable URL-4: https://filesamples.com/samples/video/m4v/sample_640x360.m4v
```

Setelah itu, program akan menyimpan URL yang telah diinput User ke dalam list URLs dan dilakukan pengecekan tiap URL apakah URL tersebut valid dan dapat di download.

```
Downloading sample_960x400_ocean_with_audio.m4v
Downloading sample_960x540.m4v
Downloading sample_1280x720.m4v
Downloading sample_640x360.m4v

File: sample_640x360.m4v, Time: 0.7062463760375977
File: sample_960x540.m4v, Time: 1.1008875370025635
File: sample_1280x720.m4v, Time: 6.926232814788818
File: sample_960x400_ocean_with_audio.m4v, Time: 8.019315004348755
Total Time Parallel Download: 8.023322820663452 seconds
```

Dapat dilihat hasil akhir berupa waktu yang diperoleh untuk melakukan download secara paralel dengan file hasil download tersimpan di direktori yang sama saat program dijalankan.



Untuk perbandingan, Penulis membuat sebuah menu CLI sederhana untuk melakukan proses perbandingan kecepatan download apabila program download dilakukan secara serial.

```
=========WELCOME TO SERIAL DOWNLOADING PROGRAM=========

How many files you want to download? 4
Enter Downloadable URL-1: https://filesamples.com/samples/video/m4v/sample_960x400_ocean_with_audio.m4v
Enter Downloadable URL-2: https://filesamples.com/samples/video/m4v/sample_960x540.m4v
Enter Downloadable URL-3: https://filesamples.com/samples/video/m4v/sample_1280x720.m4v
Enter Downloadable URL-4: https://filesamples.com/samples/video/m4v/sample_640x360.m4v

Downloading sample_960x400_ocean_with_audio.m4v
File: sample_960x400_ocean_with_audio.m4v, Time: 12.508406639099121
Downloading sample_960x540.m4v
File: sample_960x540.m4v, Time: 0.6198070049285889
Downloading sample_1280x720.m4v
File: sample_1280x720.m4v, Time: 7.597609519958496
Downloading sample_640x360.m4v
File: sample_640x360.m4v, Time: 0.576014518737793
Total Time Serial Download: 21.305870532989502 seconds
```

Bisa dilihat dalam total waktu dan waktu tiap download, bahwa parallel download hampir dua kali lebih cepat dibanding serial download. Parallel download juga memperlihatkan bahwa file - file didownload secara bersamaan, dikarenakan waktu setiap download filenya tidak jauh berbeda dibanding total time, sedangkan total time dari serial download merupakan pertambahan dari waktu tiap download file.

b. Keterbatasan

Ada beberapa keterbatasan dari program, diantaranya adalah:

- Pada saat download parallel, program tidak bisa mendownload satu file yang sama dalam dua atau lebih slot yang berbeda. Ketika mendownloadnya, karena mempunyai nama file yang sama, program akan bingung dalam *writing* filenya, sehingga membuat program berjalan lebih lama ketimbang jika mendownloadnya dengan serial.
- Program tidak bisa mendisplay progress download yang sedang berjalan.
- Program akan menyimpan file dimana script program diletakan.
- Ketika mendownload file yang besar, ketidakadaannya progress bar membuat program terlihat seperti sedang stop/freeze, padahal masih berjalan.
- check_url() hanya bisa mengecek apakah string yang dimasukkan user betul - betul url apa tidak. Function akan mem throw error jika link download tidak ada file.
- Dan keterbatasan lainnya yang belum terlihat ketika testing.

c. Lampiran Implementasi

Link Source Code : https://colab.research.google.com/drive/1MAbvaf_Cf8x72xdOG5EVuRWnjz-7rvc3?usp=sharing

Link Presentasi : https://youtu.be/1CI2WcFuw5Y

DAFTAR PUSTAKA

https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=2292039

https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=2292035

https://lms.telkomuniversity.ac.id/mod/resource/view.php?id=2292030

https://codepolitan.com/blog/menggunakan-modul-threading-di-python-589fcc066c479

https://opensourceoptions.com/blog/use-python-to-download-multiple-files-or-urls-in-parallel/

https://www.tutorialspoint.com/parallel_computer_architecture/parallel_computer_architecture_models.htm

https://www.codecademy.com/resources/docs/python/threading