Universitatea Tehnica “Gheorghe Asachi” din Iasi

Facultatea de Automatica si Calculatoare

Specializarea: Tehnologia informatiei

Disciplina: Algoritmi paraleli si distribuiti

MapReduce

Coordonator,

Alexandrescu Adrian

Student,

Mircea Mihai Adrian

Iasi 2022

1. MapReduce

MapReduce este o paradigmă de programare pentru calculul distribuit. Acest model implică, în general, existenţa unui nod de procesare cu rol de coordonator (sau master) şi mai multe noduri de procesare cu rol de worker.

2. Descrierea aplicației

Programul poate primi ca date de intrare o cale a unui director-sursă, din care se va face citirea şi memorarea conţinutului fiecărui fişier şi o cale a unui director-destinaţie, în care vor fi afişate rezultatele execuţiei programului. Aceste două căi pot să nu fie specificate, lucru ce are efect lansarea în execuţie a programului cu date de intrare prestabilite. Pentru stocarea datelor intermediare s-a folosit un folder numit "inter-files".

Datele de ieșire vor fi afișate în directorul specificat în rularea aplicației, sau în directorul prestabilit în caz contrar. Conținutul directorului destinație va cuprinde un fișier text care va conține toate cuvintele unice existente în fiecare fișier al directorului sursă și fișierul în care a fost găsit.

Exemplu:

Result.txt

< preferably, { 3.txt: 2, 19.txt: 2, 15.txt: 1} >
< suspense, { 7.txt: 4, 10.txt: 5, 12.txt: 1} >
< soothing, { 17.txt: 2, 13.txt: 2, 18.txt: 1, 11.txt: 1, 10.txt: 1, 2.txt: 3, 1.txt: 2, 15.txt: 1} >
< strata, { 8.txt: 1} >

Pentru o rulare optimă este recomandată folosirea a minim 2 procese.

3. Implementarea aplicației

Aplicația este structută în modul următor:

- Procesul n-1 (n = numărul de procese) este coordonatorul (master).
- Procesele 0…n-2 sunt desmenati drept worker.

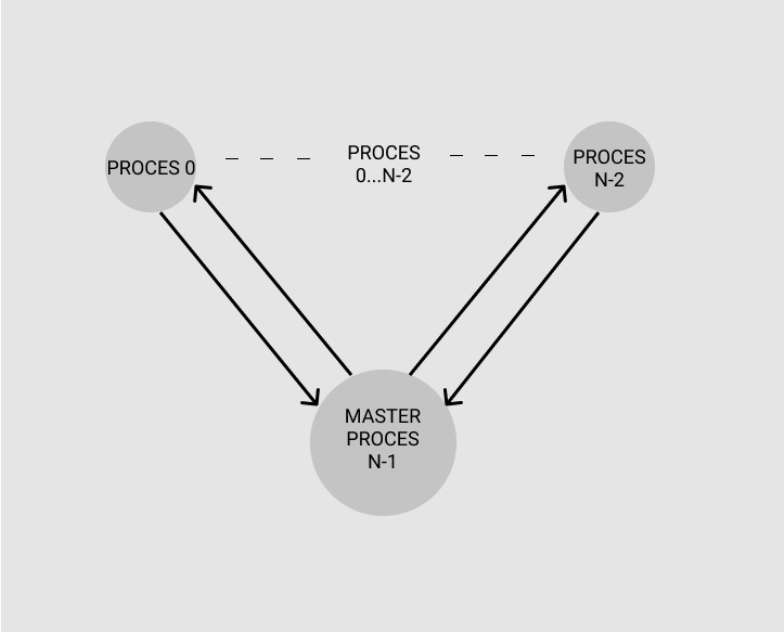

Figura 1. Comunicarea intre procese

Se primesc datele de intrare, apoi master-ul va împărți fiecărui process în mod egal fișierele din care se va face citirea. În caz că numărul de fișiere nu se împarte exact la numărul de procese, restul fișierelor vor fi împărțite câte unul începând cu primul process.

```
input_file_names = next(walk(test_files_location), (None, None, []))[2]
stop_index = 0
for i in range(0, master):
    files_per_process = int(len(input_file_names) / workers)
    if len(input_file_names) % workers > 0:
        if i < len(input_file_names) % workers:
            start_index = stop_index
            files_per_process += 1
            stop_index = start_index + files_per_process
        else:
            start_index = stop_index
            stop_index = start_index + files_per_process
    else:
        start_index = stop_index
        stop_index = start_index + files_per_process
    print("Process - " + str(i) + "\tStart: " + str(start_index)
          + "\tStop: " + str(stop_index) + "\tTotal: " + str(files_per_process))

    comm.send([test_files_location, input_file_names[start_index:stop_index]], dest=i, tag=0)
```

Figura 2. Împărțirea fișierelor pe procese

Fiecare process desemnat worker va aplica algoritmul MapReduce pentru fisierele asignate de catre master, memorând rezultatul obţinut în fișierele temporare notate cu prima literă din fiecare cuvânt găsit.

```
def mapper_function():
    print(f" <{rank}> started mapping")
    data = comm.recv(source=master, tag=0)
    inter_files_location = "inter-files"
    test_files_location = data[0]
    files = data[1]
    for file in files:
        with open(os.path.join(test_files_location, file), errors="ignore") as f:
            try:
                text = f.read()
                words = re.findall(r"[\w']+", text)
                for word in words:
                    while True:
                        try:
                            with open(os.path.join(inter_files_location, word.lower()[0] + ".txt"), 'a') as word_file:
                                word_file.write(f"< {word.lower()} , {file} >\n")
                            break
                        except:
                            time.sleep(0.01)
            except:
                print(file)
    print(f" <{rank}> finished mapping")
    comm.send("FINISHED", dest=master, tag=0)
```

Figura 3. Maparea fișierelor și scrierea în fișiere temporare.

După terminarea procesului de mapare, workeri vor trimite către master un mesaj de terminare. După primirea mesajului de terminare de la toți workeri, procesul master va împărți fișierele intermediare către workeri și va trimite datele necesare inceperi procesului de reducere.

```
    inter_files_names = next(walk(inter_files_location), (None, None, []))[2]
    stop_index = 0
    for i in range(0, master):
        files_per_process = int(len(inter_files_names) / workers)
        if len(inter_files_names) % workers > 0:
            if i < len(inter_files_names) % workers:
                start_index = stop_index
                files_per_process += 1
                stop_index = start_index + files_per_process
            else:
                start_index = stop_index
                stop_index = start_index + files_per_process
        else:
            start_index = stop_index
            stop_index = start_index + files_per_process
        print("Process - " + str(i) + "\tStart: " + str(start_index)
              + "\tStop: " + str(stop_index) + "\tTotal: " + str(files_per_process))

        comm.send([result_files_location, inter_files_names[start_index:stop_index]], dest=i, tag=0)
```

Figura 4. Împărțirea fișierelor intermediare.

În cadrul procesului de reducere se parcurg toate fișiere intermediare iar fiecare worker crează câte un dicționar pe care îl va scrie în fișierul de destinație după terminarea parcurgerii. Dacă fișierul destinație este deschis de către alt proces, procesul curent rămâne într-o buclă până când se poate accesa fișierul destinație.

```
words_count = {}
for file in files:
    with open(os.path.join(inter_files_location, file), errors="ignore") as f:
        lines = f.readlines()
        for line in lines:
            try:
                word = line.split()[1]
                source_file = line.split()[3]

                if word in words_count:
                    if source_file in words_count[word]:
                        words_count[word][source_file] += 1
                    else:
                        words_count[word][source_file] = 1
                else:
                    words_count[word] = {source_file: 1}
            except:
                print(file)
while True:
    try:
        with open(os.path.join(result_files_location, result_file_name), 'a') as output_file:

            for word in words_count:
                output = f"<{word}, " + "{ "
                for source_file in words_count[word]:
                    output += f"{source_file}: {words_count[word][source_file]}, "
                output = output[:-2] + "} >\n"
                output_file.write(output)
        break
    except:
        time.sleep(0.01)

print(f" <{rank}> finished reducing")
comm.send("FINISH", dest=master, tag=0)
```

Figura 5. Reducerea fișierelor temporare într-un singur fișier rezultat.

Bibliografie

[1] MapReduce -
https://www.talend.com/resources/what-is-mapreduce/

[2] What is a MapReduce? - Michael Kleber
https://sites.google.com/site/mriap2008/what_is_mapreduce.pdf

[3] MPI for Python -
https://mpi4py.readthedocs.io/en/stable/intro.html