PPPD - Lab. 05

Zadanie punktowane, lab 05, grupa B, 2022/2023, autor: Piotr Wolszakiewicz

Uwaga: w rozwiązaniu zadania nie można używać list.

Temat: Baza danych ze zmienną liczbą plików

Zadanie inspirowane algorytmem Consistent hashing.

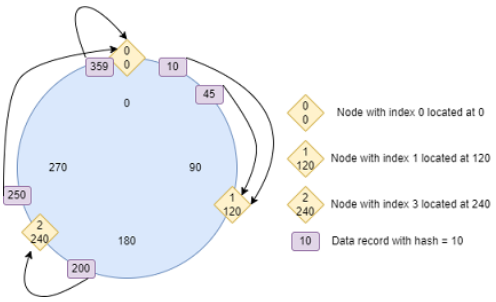
Treść zadania

Zadanie polega na zaimplementowaniu systemu do przechowywania danych w wielu plikach. Liczba plików w trakcie działania programu może się zmieniać, co nie powinno zmieniać ilości już zapisanych danych. Podczas usuwania/dodawania pliku, będziemy potrzebowali odpowiednio poprzenosić dane. Przypisanie rekordu danych do pliku odbywa się w następujący sposób:

- Wszystkie dostępne pliki rozmieszczamy na kole 0° - 360°. W naszym przypadku będziemy operowali maksymalnie trzema plikami (oznaczanymi dalej jako plik0, plik1, plik2). Pliki rozmieszczamy w sposób statyczny, mianowicie plik0 umieszczamy w 0°, plik1 - 120°, plik2 - 240°.
- Następnie wyliczamy wartość `hash` rekordu danych (jako reszta z dzielenia identyfikatora wiersza przez 360).
- Potem znajdujemy plik, który występuje za wyliczonym hash-em (tzn. jest położony najbliżej wyliczonego `hash-a` patrząc zgodnie do ruchu wskazówek zegara) i do niego dopisujemy nasz rekord danych w nowej linii (na końcu pliku).

Przypisanie wiersza do pliku

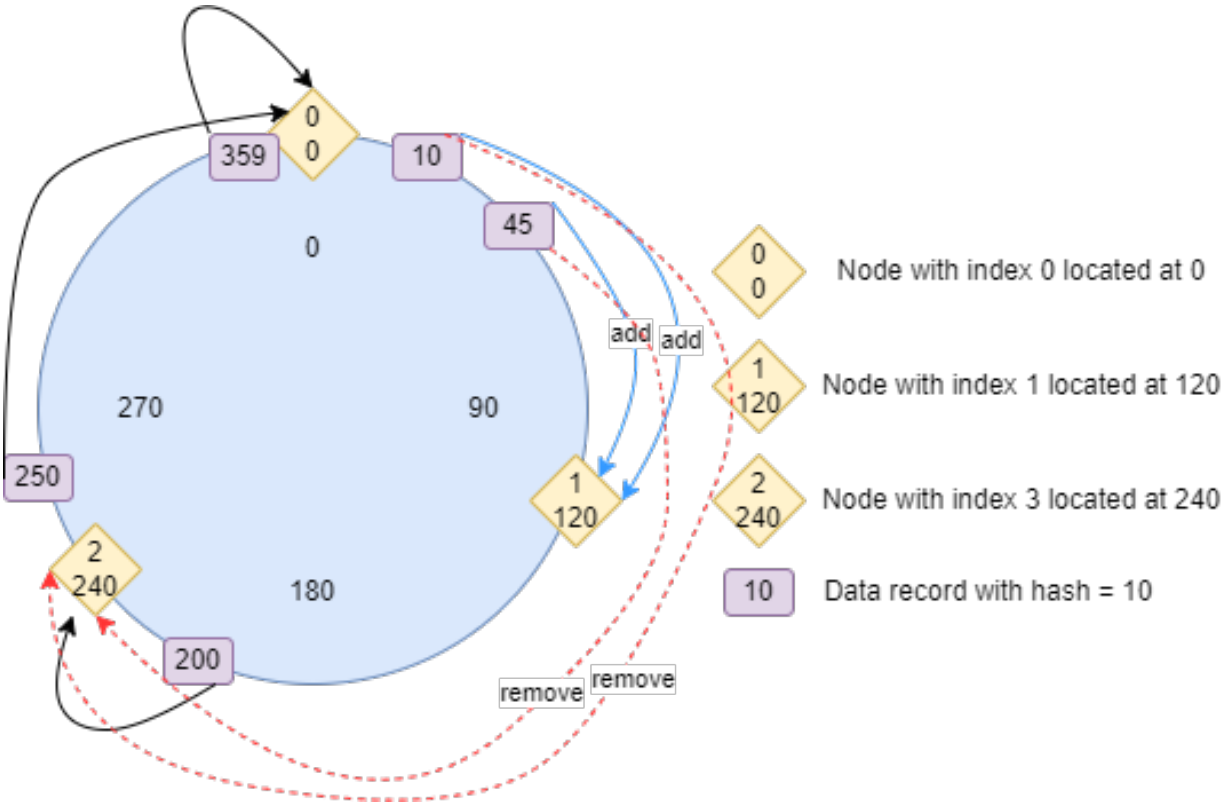
W poniższym przykładzie mamy przypadek z trzema plikami. Mamy również dane, dla których wyliczyliśmy wartości hash jako: 10, 45, 200, 250, 359. Wiersze:
10, 45 - lądują w `file1`
200 - ląduje w `file2`
250, 359 - lądują w `file0`



Rysunek 1: Consistent hashing illustration

Operacja dodania pliku

Podczas dodawania `pliku` potrzebujemy najpierw znaleźć miejsce na kole w którym `plik` najlepiej umieścić. W naszym zadaniu zawsze stosujemy stałe rozmieszczenie - `plik0` - 0, `plik1` - 120, `plik2` - 240. Drugim elementem jest przeanalizowanie wszystkich danych zapisanych w następnym `pliku` i przeniesienie wybranych do nowego. Na poniższym przykładzie dodajemy `plik1` na miejsce 120. Przeglądamy elementy z `plik2` i przenosimy element `10` i `45` do `plik1`. Element `200` pozostaje w `plik2`.



Rysunek 2: Consistent hashing illustration

Wymagania

Twoim zadaniem jest napisanie programu, który działa na zasadzie stałej interakcji z użytkownikiem. Mianowicie oczekuje na wprowadzenie akcji, którą użytkownik chce wykonać. Potem podejmuje działanie odpowiadające wybranej akcji, wyświetla wynik i czeka na kolejną akcję. Program kończy działanie po wybraniu akcji odpowiadającej za wyjście. Akcje mogą być wykonywane w dowolnej kolejności. Program powinien być odporny na sytuacje, gdzie akcja wymaga, aby przed nią była wykonana inna akcja. Przy starcie programu i każdorazowo po wybraniu nieprawidłowej akcji program powinien wyświetlić listę dostępnych akcji.
W funkcji main powinny być przechowywane zmienne: `file0_exists`, `file1_exists` i `file2_exists` jako wartości `bool`, które przechowują informację czy dany `plik` istnieje. W niej również powinna znajdować się obsługa wczytania akcji, wypisania możliwych akcji (np. poprzez wywołanie funkcji), a po każdej akcji nr 3 wydrukowanie stanu plików (patrz funkcja `print_files_state` i `get_file_name`).

Możliwe akcje to:

1. Wygeneruj wiersz danych
   W ramach tej akcji napisz funkcję `generate_row()`, która zwraca krotkę w postaci `row_id`, `login`, `level`, gdzie

- `row_id` - liczba o długości 5 cyfr, gdzie każdą z cyfr losujemy oddzielnie a następnie sklejamy np. Jeśli

wylosowaliśmy cyfry: 5,3,4,5,0 to row_id to 53450. Należy obsłużyć sytuację gdy jako pierwsze wypadnie 0.

- `login` - losowo wygenerowany ciąg znaków `a-z` (jest 26 liter pomiędzy a-z) o długości 5. Patrz pomocnicze funkcje `join_letter` do sklejania liter i `number_to_letter` do zamiany liczb 0-25 na odpowiadającą im literę. Litery znajdujące się w odległości 2 nie mogą się powtarzać. czyli `aabcd` jest prawidłowe, ale `abacd` już nie bo `a` występuję pod indeksem 0 i 2.
- `scale` - jest losowo wygenerowaną wartością z `[unsatisfactory, poor, satisfactory, good, very_good]`, przy czym unsatisfactory - wypada z prawdopodobieństwem 20%, poor - 30%, satisfactory - 30%, good - 15%, very_good - 5%.

2. Wyszukaj i wypisz uprzednio zapisany wiersz. W ramach tej akcji prosimy użytkownika o podanie `row_id`. Odszukujemy w naszym zbiorze danych czy istnieje taki wiersz i wypisujemy jego zawartość, bądź informację że nie ma wiersza o podanym `row_id`
   W ramach tej akcji należy napisać funkcję:

- `find_row(searched_row_id, file0_exists, file1_exists, file2_exists)`, gdzie `searched_row_id` - to identyfikator poszukiwanego wiersza,
  file0_exists, file1_exists i file2_exists to wartości `bool`, które przechowują informację czy dany plik istnieje. Funkcja zwraca krotkę: `row_id`, `login`, `scale`, bądź wartość `None` jeśli wiersza nie znaleziono. Uwaga funkcja powinna przeszukać tylko odpowiedni plik. Patrz pomocnicza funkcja `parse_line`, która dla linii z pliku zwraca krotkę: row_id, login, scale
- `get_file_for_row(row_id, file0_exists, file1_exists, file2_exists)`, Jest to pomocnicza funkcja, która na podstawie hash-a wyliczonego z `row_id`, zwraca id pliku do którego przynależy dany `row_id` - (0, 1, bądź 2). Hash wyliczamy jako wartość `row_id` modulo 360.

3. Dodaj plik o podanym id (0-2)
   W ramach tej akcji należy:

- wczytać i zwalidować numer pliku do dodania (0-2) - `file_id`
- napisać funkcję `can_add_file`, która przyjmuje wszystkie potrzebne parametry do sprawdzenia czy dany `file_id` może być dodany, a zwraca wartość `bool`. Pliku nie można dodać jeśli plik o podanym id już istnieje. W takim przypadku odpowiedni komunikat powinien zostać wypisany.
- napisać funkcję `get_next_file_id(file_id, file0_exists, file1_exists, file2_exists)`, gdzie: `file_id` - identyfikator pliku który dodajemy.
  `file0_exists, file1_exists, file2_exists` - wartości bool mówiące czy dany plik już istnieje czy nie.
  Funkcja zwraca identyfikator pliku, z którego dane należy ewentualnie poprzenosić do dodawanego pliku.
- napisać funkcję `add_file(file_id, file0_exists, file1_exists, file2_exists)` gdzie: `file_id` - identyfikator pliku który dodajemy Funkcja ta powinna przeczytać dane z odpowiedniego istniejącego `pliku` i ewentualnie dopisać je do nowego. Powinna również usunąć dane które zostały przeniesione. Podpowiedź: aby zmodyfikować plik, możesz przeczytać całą jego zawartość do zmiennej, usunąć plik z dysku i stworzyć od nowa z nową zawartością. Patrz pomocnicze funkcje `remove_file_from_disk` oraz `get_line_starting_from_index` dzięki której podzielisz zawartość całego pliku na linie. Poniżej znajduje się przykład użycia.

4. Wyjdź z programu

Ustawienia startowe

- Proszę ustawić `seed` dla funkcji `random` wartością `2000`
- W funkcji `main` przechowuj informacje czy pliki istnieją `file0_exists`, `file1_exists`, `file2_exists`. Na start dostępny jest tylko `file2` - patrz dostarczony plik `file2.txt` (umieść w tym samym katalogu co `main`). Dostępny jest również plik `2022-IAD-05B_stud.py` ze szkieletem programu do dokończenia.

Pomocnicze funkcje

```
import os
import os.path

def remove_file_from_disk(file_name):
    if os.path.exists(file_name):
        os.remove(file_name)

def number_to_letter(number):
    return chr(number + ord('a'))

def join_letter(base, letter):
    return base + letter

def parse_line(line, separator=" "):
    line_items = line.split(separator)
    if len(line_items) != 3:
        return None
    return int(line_items[0]), line_items[1], line_items[2]

def get_file_name(file_id):
    """Zwraca nazwę pliku na podstawie identyfikatora pliku"""
    return f"file{file_id}.txt"

def print_files_state(files_count=3):
    """Wypisuje stan plików"""
    for file_id in range(files_count):
        file_name = get_file_name(file_id)
        if not os.path.exists(file_name):
            continue
        content = ''
        with open(file_name, "r") as read_file:
            print(f'-- {file_name} --: ')
            content = read_file.read()

        if content.strip() !='':
            print(content.strip())

def get_line_starting_from_index(content, start_index=0):
    """Funkcja zwraca linię startując od start_index bądź -1 jeśli nie ma"""
    if not content:
        return -1, None

    end_index = content.find('\n', start_index)
    if end_index == -1:
        return -1, None
    line = content[start_index:end_index]
    return end_index, line
```

```
# Przykłady użycia
initial ='a'
initial = join_letter(initial, 'b')
print(initial)     # ab
```

```python
print(number_to_letter(0))     # a
print(number_to_letter(25))    # z

# iterowanie linia po lini
file_content = '99731 wbhxp satisfactory\n36790 bjduj satisfactory\n'
end_index, line = get_line_starting_from_index(content, 0)
while end_index != -1:
    print(line)
    end_index, line = get_line_starting_from_index(content, end_index+1)


# Wydrukuje odpowiednio
# 99731 wbhxp satisfactory
# 36790 bjduj satisfactory
```

Punktacja

Komunikacja z użytkownikiem (tzn. wszystkie wczytywania danych i wypisywanie informacji) powinny znajdować się w funkcji `main`. Za poszczególne elementy można uzyskać następującą liczbę punktów:

- Poprawnie zaimplementowana i wywołana funkcja `generate_row`, parametry wyjściowe i logika losowania - `2pkt`
- Poprawnie zaimplementowana i wywołana funkcja `get_file_for_row` - `2pkt`
- Poprawnie zaimplementowana i wywołana funkcja `find_row` - `2pkt`
- wczytanie identyfikatora pliku do usunięcia i poprawnie zaimplementowanie funkcji `can_add_file` - `1pkt`
- Poprawnie zaimplementowana i wywołana funkcja `get_next_file_id` - `1pkt`
- Poprawnie zaimplementowana i wywołana w main funkcja `add_file` - `2pkt`
- Każdy z wymienionych etapów wymaga, aby były stworzone odpowiednie funkcje do tego etapu i aby były one prawidłowo wywołane w `main`. Punktacja za wywołanie funkcji jest wliczona w punkty za funkcję.

Uwaga

- Jeśli rozwiązanie nie spełnia postawionych wymagań (korzysta z list), zadanie jest oceniane na 0 punktów.
- Jeśli program się nie kompiluje (interpretuje), ocena jest zmniejszana o połowę.
- Jeśli kod programu jest niskiej jakości (nieestetycznie formatowanie, mylące nazwy zmiennych itp.), ocena jest zmniejszana o `2pkt`.

Przykłady interakcji użytkownika z programem

```
Możliwe akcje to:
1 - Wygeneruj wiersz danych
2 - Wyszukaj zapisany wiersz po row_id
3 - Dodaj node o podanym id (0-2)
4 - Wyjdź z programu
Podaj jaką akcję chcesz wykonać: 0
Nieprawidłowy numer akcji
Możliwe akcje to:
1 - Wygeneruj wiersz danych
```

```
2 - Wyszukaj zapisany wiersz po row_id
3 - Dodaj node o podanym id (0-2)
4 - Wyjdź z programu
Podaj jaką akcję chcesz wykonać: 2
Podaj id wiersza do odszukania: 30307
row_id:30307, login:frwtl, scale:good
Podaj jaką akcję chcesz wykonać: 1
Wygenerowano wiersz: row_id:70742, login:sueyy, scale:poor
Podaj jaką akcję chcesz wykonać: 1
Wygenerowano wiersz: row_id:28955, login:qnagx, scale:satisfactory
Podaj jaką akcję chcesz wykonać: 3
Podaj id pliku do dodania: 2
Nie można dodać pliku o id: 2
Podaj jaką akcję chcesz wykonać: 3
Podaj id pliku do dodania: 1
Dodano plik o id: 1
-- file1.txt --:
99731 wbhxp satisfactory
36790 bjduj satisfactory
90101 dqnhg unsatisfactory
12243 rztcz satisfactory
88239 aokbk good
30307 frwtl good
-- file2.txt --:
70742 sueyy poor
28955 qnagx satisfactory
19381 jonbk satisfactory
26478 nxzzm unsatisfactory
53967 eiyve unsatisfactory
Podaj jaką akcję chcesz wykonać: 3
Podaj id pliku do dodania: 0
Dodano plik o id: 0
-- file0.txt --:
19381 jonbk satisfactory
53967 eiyve unsatisfactory
-- file1.txt --:
99731 wbhxp satisfactory
36790 bjduj satisfactory
90101 dqnhg unsatisfactory
12243 rztcz satisfactory
88239 aokbk good
30307 frwtl good
-- file2.txt --:
70742 sueyy poor
28955 qnagx satisfactory
26478 nxzzm unsatisfactory
Podaj jaką akcję chcesz wykonać: 2
Podaj id wiersza do odszukania: 99999
Nie znaleziono wiersza o podanym row_id: 99999
Podaj jaką akcję chcesz wykonać: 2
Podaj id wiersza do odszukania: 30307
row_id:30307, login:frwtl, scale:good
Podaj jaką akcję chcesz wykonać: 4
```