Отчет по лабораторной работе 6
По предмету “Анализ алгоритмов”
По теме “Конвейер”

Фирсова Дарья ИУ7-56

2018

Введение

В лабораторной работе изучается параллельный алгоритм конвейера. Используются очереди и средства синхронизации потоков. Конвейер можно использовать, когда задачу можно разбить на части, при этом каждый поток выполняет свою часть. После выпоолнения очередной задачи, результат вычислений передается на следующую "ленту".
Цель лабораторной работы: составить алгоритм конвейра, использовать общие очереди и мьютексы, записать работу поэтапно в файл. В конвейере использовать 3 уровня.

1 Аналитеская часть

В данном разделе приведены алгоритмы умножения.

1.1 Описание алгоритмов

Конвейер состоит из трех уровней и трех очередей. На первом уровне элемент поступает из первой очереди, и создаются два других числа возведением исходного в 2 и 4 степень. Затем оба элемента поступают во вторую очередь. На втором уровне элемент забирается элемент из второй очереди, возводится в степень 6 и 8, поступает в 3 очередь. На третьем уровне к полученному элементу прибавляется число и "лента"завершает работу.

2 Конструкторская часть

В данном разделе представлена схема конвейера.

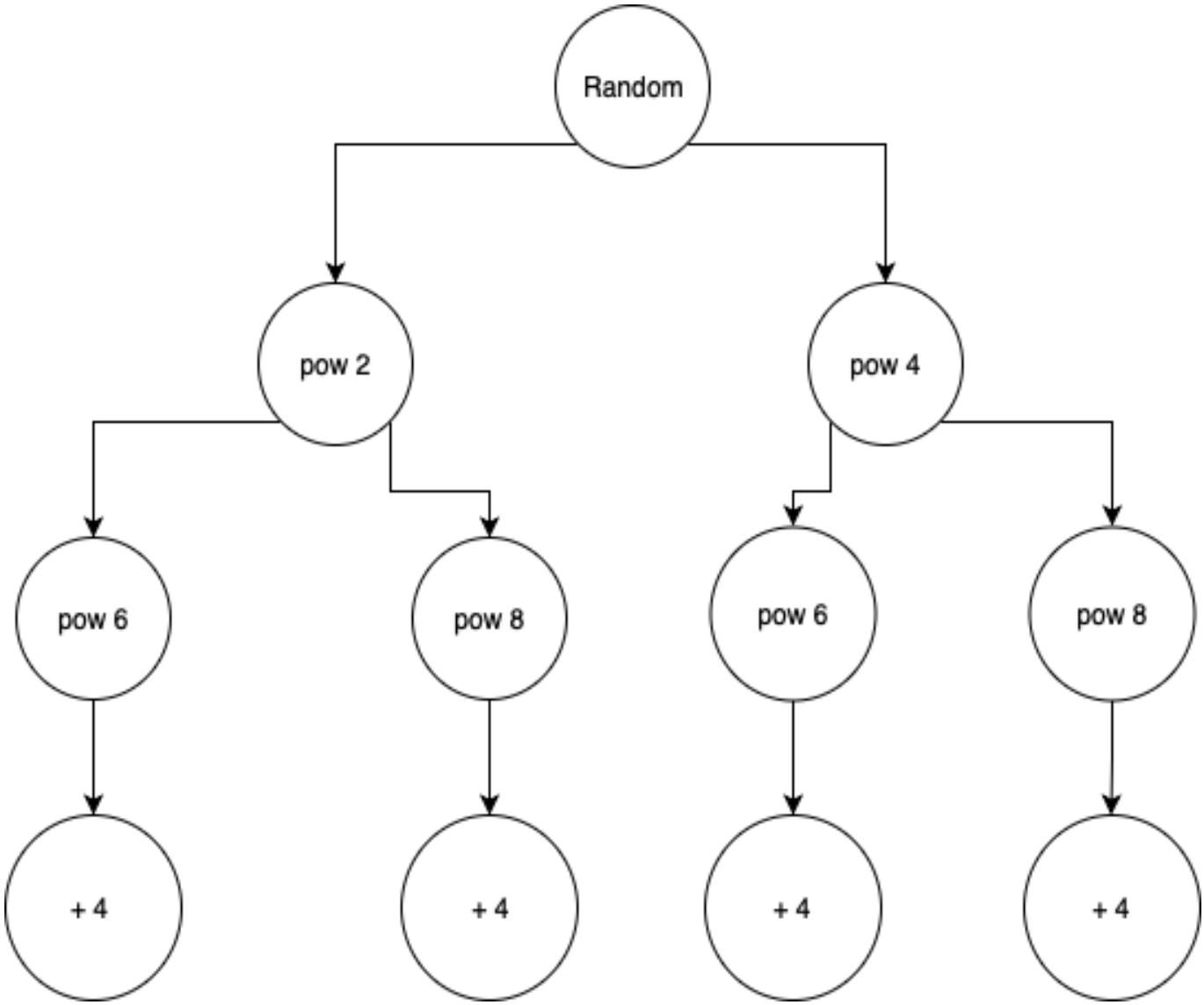

Рис. 1: Схема конвейра

3 Технологическая часть

В этом разделе приведена реализация функций, указан язык программирования и необходимые модули.

3.1 Средства реализации

В данной работе использовался язык Python 3.6, в среде Pycharm. Для измерения времени использовался модуль time, измерения производились в секундах. Для создания потока использовался модуль threading.

3.2 Листинг кода

```
import threading
import time
import queue
import random


def level1():
    i = 1
    time_end = time.time() - timeBegin
    while time_end < 3:
        if not queue1.empty():
            while(not queue1.empty()):
                deleted = queue1.get()
                tmp1 = pow(deleted, 2)
                tmp2 = pow(deleted, 4)
                time.sleep(0.00001)
                print('write', tmp1, i)
                print('write', tmp2, i)
                mutex2.acquire()
                queue2.put(tmp1)
                queue2.put(tmp2)
                mutex2.release()
                i = i+1
        time_end = time.time() - timeBegin
    print('Level1 stop')


def level2():
    global timeBegin
    #time.sleep(0.001)
    time_end = time.time() - timeBegin
    print(time_end)
    i = 1
    while time_end < 3:
        if not queue2.empty():
            while(not queue2.empty()):
                mutex2.acquire()
                deleted = queue2.get()
                mutex2.release()
                tmp1 = pow(deleted, 6)
                tmp2 = pow(deleted, 8)
                mutex3.acquire()
```

```
            queue3.put(tmp1)
            queue3.put(tmp2)
            mutex3.release()
            f.write('{} {}\n'.format(tmp1, i))
            f.write('{} {}\n'.format(tmp2, i))
            i = i + 1
        time_end = time.time() - timeBegin
    print('LEVEL2 STOP')


def level3():
    global timeBegin
    #time.sleep(0.005)
    time_end = time.time() - timeBegin
    i = 1
    while time_end < 3:
        if not queue3.empty():
            mutex3.acquire()
            deleted = queue3.get()
            mutex3.release()

            result = deleted + 4
            f.write('{} {}\n'.format(result, i))
            i = i + 1
        time_end = time.time() - timeBegin
    print('LEVEL 3 STOP')


if __name__ == '__main__':
    f = open('log.txt', 'w')
    threads = []
    queue1 = queue.Queue()
    for i in range(1,50):
        queue1.put(random.randint(1, 100))
    queue2 = queue.Queue()
    queue3 = queue.Queue()
    timeBegin = time.time()
    mutex1 = threading.Lock()
    mutex2 = threading.Lock()
    mutex3 = threading.Lock()
    t1 = threading.Thread(target=level1)
    t2 = threading.Thread(target=level2)
    t3 = threading.Thread(target=level3)
    threads.append(t1)
    threads.append(t2)
    threads.append(t3)
    for thread in threads:
        thread.start()

    for thread in threads:
        thread.join()
```

Листинг 1. Реализация алгоритмов.

4 Экспериментальная часть

В данном разделе будут приведены примеры работы алгоритмов и произведены замеры времени. Тестирование производилось на компьютере с процессором Intel Core i5 (I5-6267U) и оперативной памятью 8 Гб.

4.1 Примеры работы

Результат работы алгоритма находится в файле log.txt

4.2 Вывод

В файле лога можно увидеть, что значения на каждом этапе вычисляются параллельно. На вход подаются 50 случайных значений, выходных значений в 4 раза больше. В результате работы программы вычисляются степени числа.

5 Заключение

В лабораторной работе разработаны алгоритмы параллельного вычисления для разработки конвейера. Были изучены механизмы работы очередей, мьютексов и потоков. Для составления отчета использован Latex.