Homework 2

Simon Sun ys3031

**Overview**

In this homework, I use the dataset [Seattle Library Checkout Records](https://www.kaggle.com/seattle-public-library/seattle-library-checkout-records), and tried to find records that have specific attributes. I optimize my model by implementing **Fusion**.

**Task**

The task is to filter all library checkout records with “ItemType” equal to “jcbk” and “Collection” equal to “ncpic”.

**Implementation**

I implemented multi-thread in Python 3. Each computation node (operator) is represented as a thread. The structure before fusion is like:

*Reading Thread -> Buffer 1 -> Thread A -> Buffer 2 -> Thread B -> Output*

The Reading Thread continuously reads data from dataset and sends data to Thread A. Thread A filters data, sends only those data with attribute “ItemType” equal to “jcbk” to Thread B. Thread B filters data and sends only those data with attribute “Collection” equal to “ncpic” to the Output. Also, I use buffers, implemented by LinkedList, to store data that has beed processed by previous operator and is waiting to be processed by next step. Obviously, this will generate a communication cost.

I implement fusion by merge Thread A and Thread B into one thread – Thread Fused. After fusion, the structure looks like:

*Reading Thread -> Buffer 1 -> Thread Fused -> Output*

Each line of the data (tuple) is marked with a timestamp whenever it enters a thread or leaves a thread. The communication cost is defined as the timestamp a tuple enters an operator minus the timestamp it leaves the previous operator. The operator cost is defined as the timestamp a tuple leaves an operator minus the timestamp it enters this operator. The throughput is defined as 1 / (total cost), where total cost is the time from this tuple is read to it is output.

**Result**

According to the paper, the performance of Fusion is measured by plotting a graph, with x-axis being (operator cost)/(communication cost) and y-axis being throughput.

I am sorry that there is a weird bug in my matplotlib.pyplot package, and it could only generate one series in one graph. That is, it cannot plot the throughput both before fusion and after fusion in one graph. So, I have to plot them in two different plots.

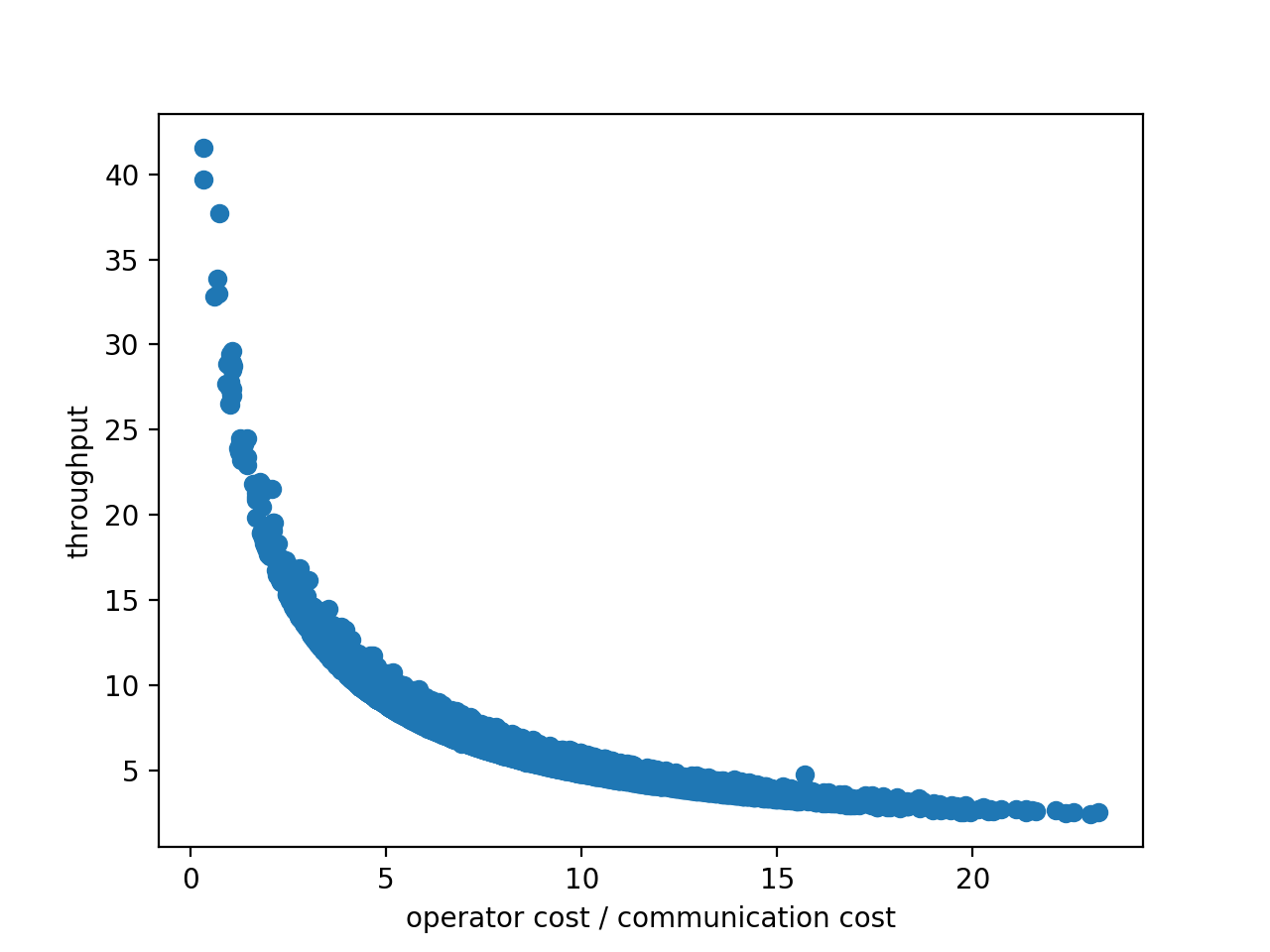
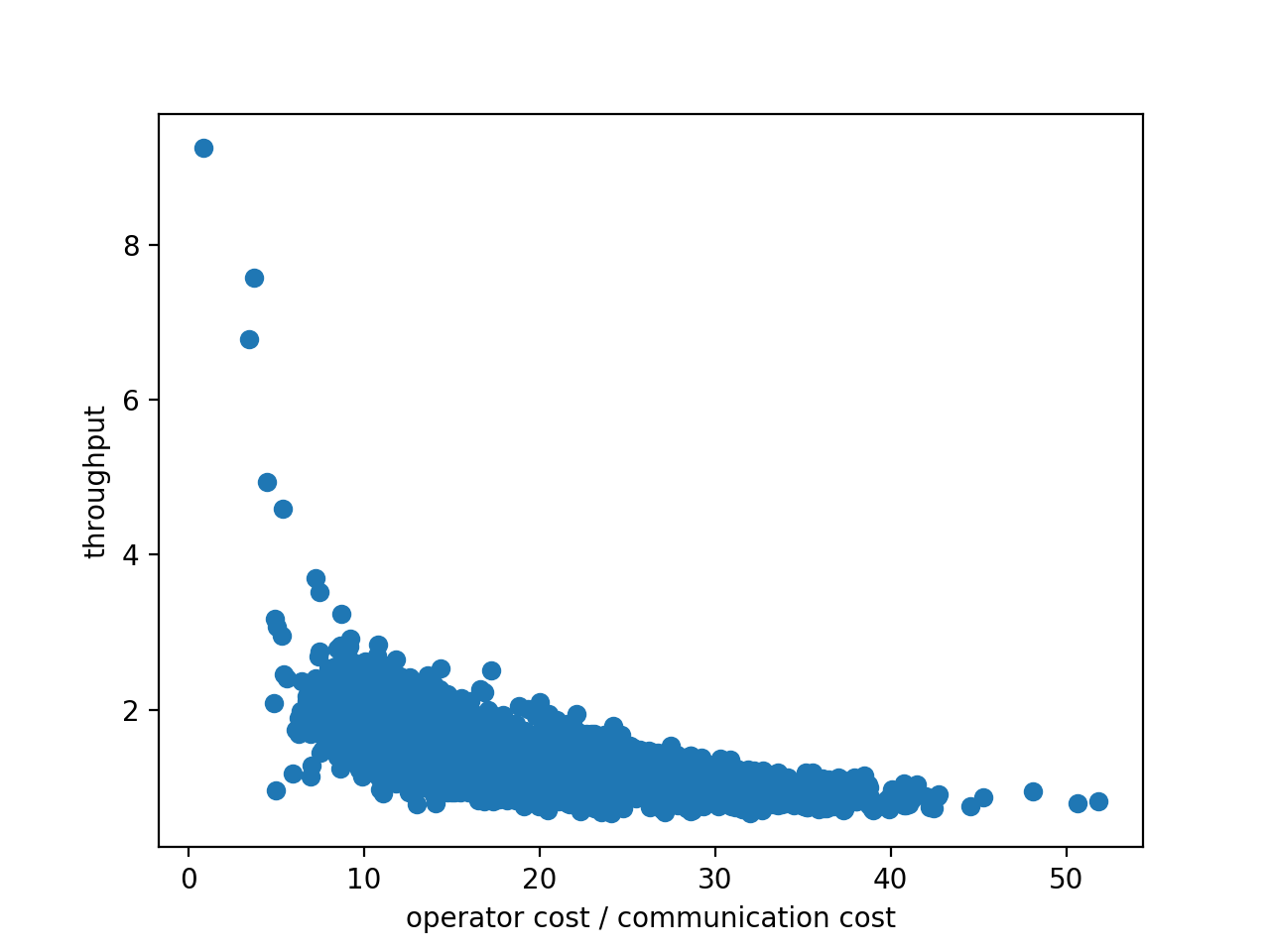


Figure 1 - before fusion Figure 2 – after fusion

**Analysis**

From the figures above, we can see that the performance of after-fusion is significantly better than that of before-fusion.

Multiple reasons may lead to this result. First, the cost of frequently acquiring locks and releasing locks may be the bottleneck of the system. Since My program runs in multi-thread environment, it is necessary to put a lock to each of the buffer. Hence, the before-fusion version has one more operator, and the cost of it is significantly increased due to one more time to acquire locks and release locks. Second, Python’s multi-thread environment is, actually, pseudo-multi-thread. That is, only one thread can access the Python interpreter at a time, and thus, one more thread means one more cost.

Unfortunately, I cannot generate the ideal graph like the paper shows. If it is able to generate data with x-axis larger than 100 or more, maybe we can see the result that is more similar to the graphs in the paper given by Professor Turaga.

You may visit

<https://github.com/sym19940907/large_data_stream_process_hw2>

to test my programs. Simply run $ python3 before\_fusion.py or $ python3 after\_fusion.py