# *Work flow and Analysis:*

At the start I just want to start with the basic evaluation and analysis, like what is the problem and what is going inside the program. So, simply first executed the program several times and noted the results.

| Number of Workers | Package Planned | Actually delivered |
|---|---|---|
| 1 | 2000 | 2000 |
| 2 | 2000 | 3028 |
| 3 | 2000 | 3835 |
| 4 | 2000 | 3423 |
| 5 | 2000 | 3182 |
| 6 | 2000 | 3481 |

Second Phase(lower value):

| Number of Workers | Package Planned | Actually delivered |
|---|---|---|
| 1 | 500 | 500 |
| 2 | 500 | 764 |
| 3 | 500 | 942 |
| 4 | 500 | 780 |
| 5 | 500 | 843 |
| 6 | 500 | 871 |

***My first observation is the problem:***

AFTER SEVERAL CHECK, I found there is no problem with the NUMBER OF PACKAGES, There is a problem with the NUMBER OF WORKERS.

My first observation is the problem arises when the number of workers greater than 1 (when there is one worker the packages are correctly delivered)

***What are all may be the reason:***

1. First reason may be the same package is being delivered by 2 or more times by different workers.

2. The logger failed to maintain the package delivered information correctly

3. Even after delivering the package, the package delivered status didn't update by the worker.

4. Same package assigned to 2 or more workers (need to check already assigned or not while assigning packages) ((((CORRECT))))

5. Threading is used here, so there is a chance for "race condition".

***SOLUTION:***

After checking all the possibilities, The 4th reason is correct. While assigning the packages to the worker, they are checking only if the package is delivered or not. But actually, sometimes the package is already assigned to someone and the worker is busy with delivering that package. So the status of the package(delivered) was not changed, as a result, the same package will be assigned to another worker again. This is the actual problem acquiring here.

So simply check, whether the package is already assigned to someone or not. If yes, move to the next package. If no, then assign that package to the coming worker.

Now the package problem is fixed

| Package planned | Actually delivered | Time taken (sec) |
|---|---|---|
| 20 | 20 | 1 |
| 50 | 50 | 2 |
| 500 | 500 | 4 |
| 3000 | 3000 | 10 |
| 5000 | 5000 | 19 |

## *Optimization:*

Now it is taking 13-18 sec to deliver 5000 packages . So, optimization is required.(depending upon the CPU usage in my system)

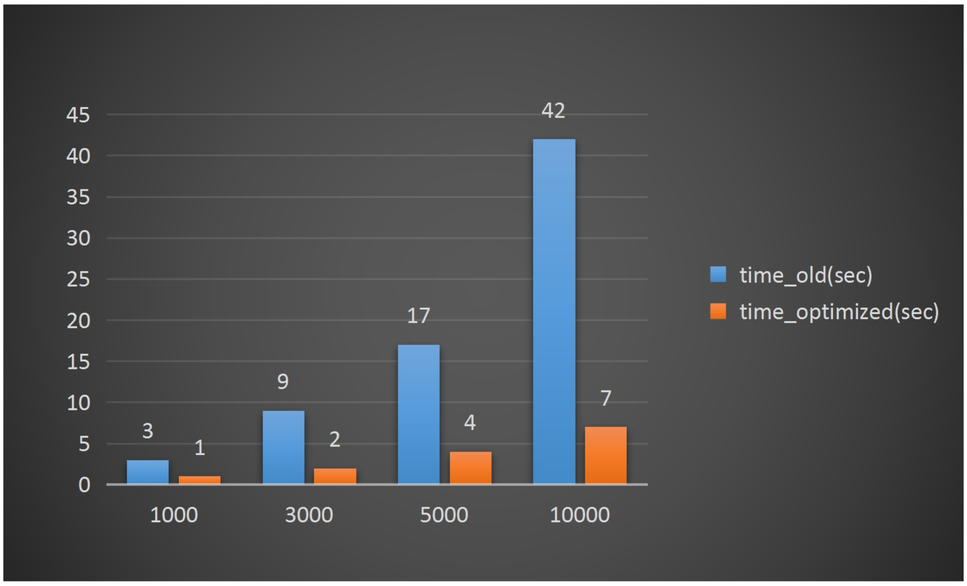
Did basic optimization - like eliminated the unwanted for loops in the code.

***Elimination of for-loops:***

- ✓ While assigning the package to the worker there is no need to loop over all the packages available, so just get the package by indexing.
- ✓ While getting the number of packages left to be delivered, there is no need for again loop over all the packages available and check the delivered status to count the not delivered packages, So keep track of the number of packages delivered and simply return that count.

***Result:***

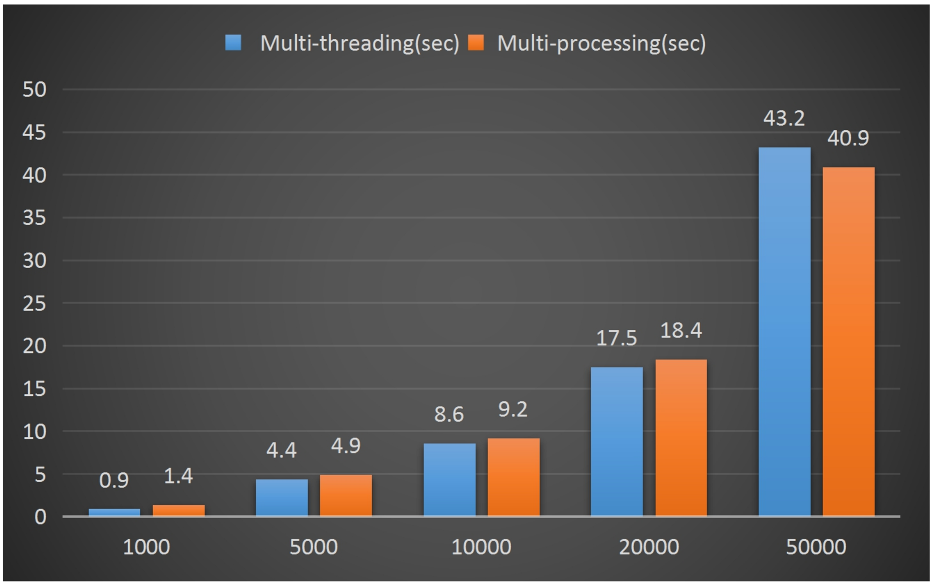
After eliminated is 2 for loops it is now taking 4 sec for 5000 package with 6 workers.



Further optimization --> thinking why multi-threading? Can't I use some other technology or method?

I can use multi-processing in the place of multi-threading. The main difference between the multi-processing and multi-threading in python is threading uses a single GIL(Global Interpreter Lock) for the whole process. Multi-processing uses separate GIL for each process.

The comparison between the Multi-processing and multi-threading is done, and we can see the results below.



## *Conclusion:*

In the end, after all, comparisons and modifications, I changed the programming a little bit and cleared the bug. To increase the execution time of the program, I eliminated the unwanted loops and variables.