



# **CS 342 Operating System**

## **Project #1**

### **Processes, IPC, and Threads**

İlke Doğan 21702215

Section 1

## ***TABLE OF CONTENTS***

<b>Introduction</b>	<b>2</b>
<b>Computer Specifications</b>	<b>2</b>
<b>Graphs</b>	<b>2</b>
<b>Conclusion</b>	<b>5</b>

### **Introduction**

The fundamental goal of the project is computing several statistics for integer data sets. While realizing this application program, the server and client-side is set. Server, Client, and the User are communicated over the requests. These requests are sent over POSIX (Portable Operating System Interface for Unix). There are specific request prompts such as count, avg. These requests are gathered from the client and after this request is fulfilled, another request is waited by the client.

Two different versions of this application programming, whose purpose is shared, were made. To implement child processes, the first method was to use the pipe logic and the second was to use the thread method. The result I expected before running the experiment was to expect the thread implementation to be faster than the pipe logic.

### **Computer Specifications**

Windows 10

Intel® Core™ i7-9750H CPU @ 2.69GHz

16.0 GB

The code is tested by the Oracle Virtual Box, Ubuntu(64-bit) with a text editor. The terminal with C language compiler helps run both server and client-side at the same.

### **Graphs**

Based on the instructions given in the project manual, the test cases are created. Firstly, the application asks for one of the requests in the list and then, user types in the command prompt. For the first case, the application is designed by the pipe. In that situation, the results are given below in Figure 1 and Table 1.

<b>Requests</b>	avg	avg 100 3000	count	count 75 340000	range 20000 25000 50	max
<b>Time 'milliseconds</b>	660 milliseconds	737 milliseconds	893 milliseconds	1066 milliseconds	1306 milliseconds	1479 milliseconds

Table 1- Pipe Version Test Case

```

ilke@ilke-VirtualBox: ~/Downloads/project1_part1
ilke@ilke-VirtualBox:~/Downloads/project1_part1$ ./statclient
mq created, mq id = 3
mq maximum msgsize = 8192
mq opened, mq id = 4
Enter the request code: avg
-----
Sent message -> commandType = 1
-----
Received response val -> id = 540055.687500
-----
The program working time: 660 milliseconds.
Enter the request code: avg 100 3000
-----
Sent message -> commandType = 1
-----
Received response val -> id = 2986.000000
-----
The program working time: 737 milliseconds.
Enter the request code: count
-----
Sent message -> commandType = 0
-----
Received response val -> id = 300.000000
-----
The program working time: 893 milliseconds.
Text Editor request code: count 75 340000
-----
Sent message -> commandType = 0
-----
Received response val -> id = 93.000000
-----
The program working time: 1066 milliseconds.
Enter the request code: range 20000 25000 50
-----
Sent message -> commandType = 3
-----
Received integer array with size 3:
22632 23748 24592
-----
The program working time: 1306 milliseconds.
Enter the request code: max
-----
Sent message -> commandType = 2
-----
Received response val -> id = 1048480.000000
-----
The program working time: 1479 milliseconds.

```

Figure 1- Pipe Version Test Case

Based on the instructions given in the project manual, the test cases are created. The second application asks for one of the requests in the list and then, user types in the command prompt. For the second case, the application is designed by the threads. In that situation, the results are given below in Figure 2 and Table 2.

<b>Requests</b>	avg	avg 100 3000	count	count 75 340000	range 20000 25000 50	max
<b>Time 'milliseconds'</b>	555 milliseconds	617 milliseconds	681 milliseconds	739 milliseconds	818 milliseconds	888 milliseconds

Table 2- Thread Version Test Case

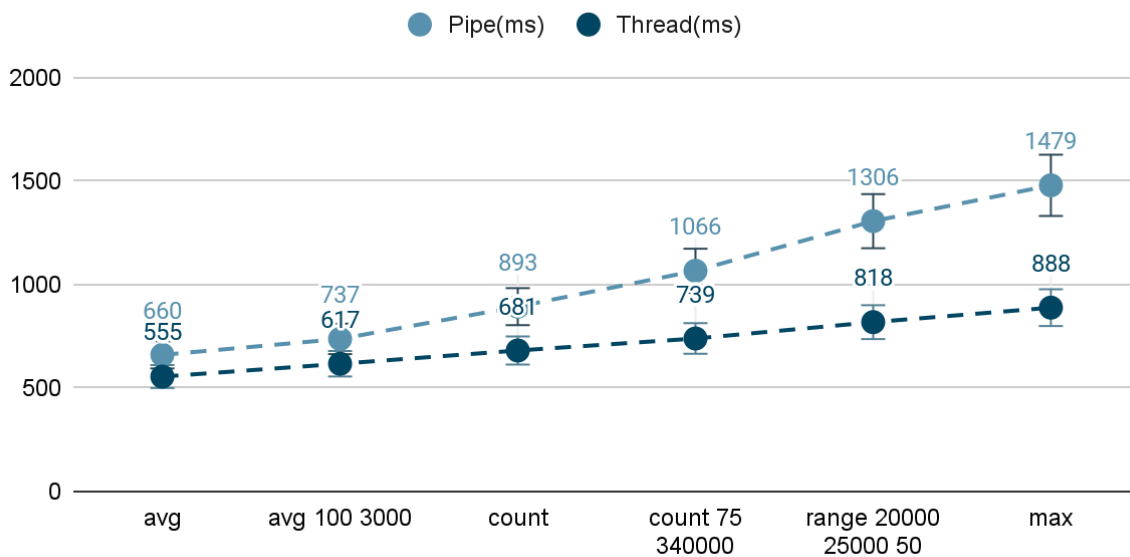
```

ilke@ilke-VirtualBox: ~/Downloads/project1_part2
ilke@ilke-VirtualBox:~/Downloads/project1_part2$ ./statclient
mq created, mq id = 3
mq maximum msgsize = 8192
mq opened, mq id = 4
Enter the request code: avg
-----
Sent message -> commandType = 1
-----
Received response val -> id = 551496.375000
-----
The program working time: 555 milliseconds.
Enter the request code: avg 100 3000
-----
Sent message -> commandType = 1
-----
Received response val -> id = 2986.000000
-----
The program working time: 617 milliseconds.
Enter the request code: count
-----
Sent message -> commandType = 0
-----
Received response val -> id = 300.000000
-----
The program working time: 681 milliseconds.
Enter the request code: count 75 340000
-----
Sent message -> commandType = 0
-----
Received response val -> id = 233.000000
-----
The program working time: 739 milliseconds.
Enter the request code: range 20000 25000 50
-----
Sent message -> commandType = 3
-----
Received integer array with size 3:
22632 23748 24592
-----
The program working time: 818 milliseconds.
Enter the request code: max
-----
Sent message -> commandType = 2
-----
Received response val -> id = 1048480.000000
-----
The program working time: 888 milliseconds.
Enter the request code:

```

Figure 2- Thread Version Test Case

## Comparison of running time of child process usage and thread usage.



## Conclusion

In conclusion, I have proven the conclusions envisaged at the beginning with the results I have obtained in the experiments. The execution time of threads is shorter than that of pipes. As a result, threads are much faster. In fact, the last example has almost half the speed compared to the pipe.