AHMEDABAD UNIVERSITY

Global Education at Local Cost, Context and Ethos™

School of Engineering and Applied Science

OPERATING SYSTEMS PROJECT REPORT

Process Migration

Instructors :
Prof. Sanjay Chaudhary
Prof. Mansukh Savaliya

Mentors :
Aditya Parikh
Kaivalya Shah

Team Members :
Anshul Jethvani (201501037)
Meet Patel (201501074)
Jay Modi (201501059)

Introduction :

The concept of a process is not introduced in this report as processes are a well-known design in operating systems. Process Migration refers to the mobility of executing (or suspended) processes in a distributed computing environment. Usually, this term indicates that a process uses a network to migrate to another machine to continue its execution there. Sometimes the term is used to describe the change in execution from one processor to another processor within the same machine. In layman’s terms - process migration is programs migrating between machines.

Process migration is an essential technique in distributed computing to increase effectiveness of process execution as it is a specialized form of process management where processes are moved from one computing environment to another. Nowadays Process Migration used widely. On multicore machines (multiple cores on one processor or multiple processors) process migration happens as a standard part of process scheduling. It is quite easy to migrate a process within a given machine, since most resources (memory, files, sockets) do not need to be changed, only the execution context (primarily program counter and registers) are subject to change.

Brief description :

In this project, we have created a simulational environment to implement Preemptive Process Migration where we have defined a server as a central entity to manage process migration and process information check on multiple clients (possibly one). This server can check the status of all clients and can control the process migration between them i.e. any process from any client can be migrated to any other client. As this is a simulation, we have created processes instead using real processes for migration purpose.

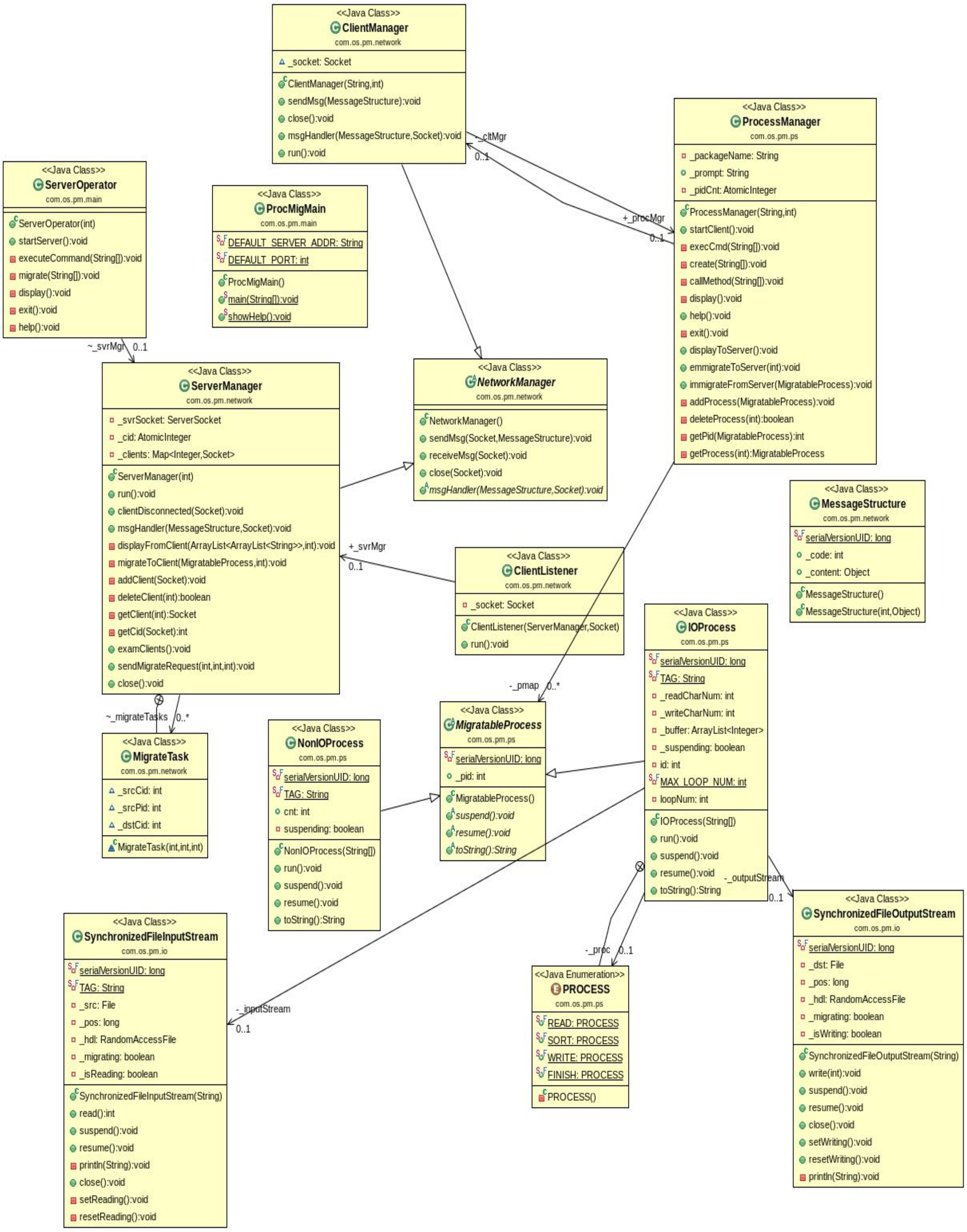
The main idea to design and implement process migration using Java, is because of its persistency i.e. Serialization and Deserialization in packaging/unpackaging objects. Also it has a rich and easy to use Socket library API for network communication to send/receive packaged object stream. Apart from this, Java provides thread management, concurrency mechanism to synchronize shared resources access between threads.

Architecture / Model / Diagrams :

The classes developed are:

- ProcMigMain: It contains main function and it decides whether the node will act as server or client.
- ServerOperator: It executes the commands for different operations of server.
- NetworkManager: It is an abstract class which contains methods to send, receive and handle messages on clients and servers.
- ServerManager: It is used to connect new clients with running server and to handle the coming messages.
- ClientListner: On server side, it receives messages coming from different clients.
- ClientManager: This is used by client to connect with server and to receive and handle messages coming from server.
- MessageStructure: This is the structure of the message which is to be sent.
- ProcessManager: It is used for executing different commands on client side and it also contains ClientManager.
- MigratableProcess: This is an interface which is to be implemented by all processes.
- NonIOProcess: It is a migratable process which increases value of counter variable by one at every 0.5 seconds.
- IOProcess: It is a migratable process which continuously reads from a file, sorts its data and writes it to another file.
- SynchronizedFileInputStream: It is a class to facilitate migrating process with open file.
- SynchronizedFileOutputStream: It is a class to facilitate migrating process with open file.

Class diagram:



Technical Specifications / Details :

- Multiple clients are supported. Server can check the status of all clients and can control the process migration between them.

- Server can migrate a running migratable process any time with an input command.

- Network communication scalable. There are generalized message structure and dispatcher in the framework. You can add any type of message to the framework conveniently.

- Load balancing Algorithms or Migration Policies can be added easily in ServerOperator class which is loosely connected to server.

- **Threads** have been used extensively for better functioning of the application on client side (each process created by client) as well as server side.
    1. ServerManager thread: to accept client connections
    2. ClientListener threads: to communicate with newly joined client.

- **Concurrency** is being effectively managed by wisely using java keywords like ‘volatile’ and ‘synchronized’. As we are treating a client’s each process as threads, there will be many threads running concurrently. To suspend or resume these threads (when called to migrate), we are using a volatile variable through which we can allow multiple threads to access that variable simultaneously and make the implementation easy. Moreover, we have created volatile Maps, one to keep track of process id and process object and second to keep track of client id and its socket which is also being used by multiple threads i.e. multiple processes from different clients and hence using volatile Maps makes concurrent management **graceful**.

- **Exceptions** have been handled and appropriate error messages have been displayed wherever possible using Java’s Exception Handling mechanism. This takes care of possible **aborts and faults**.

Algorithms/Flow charts :

Migration Algorithm :

- Client creates a process and registers the process info to ProcessManager.

- ClientManager runs the process in new thread.

- Server asks for information of all processes running on all clients.

- ServerOperator requests migration of a process with given ProcessID from client with given SrcClientID to client with given DestClientID.

- ServerManager asks the ProcessManager to emigrate process to server.

- ProcessManager suspends the process and sends it to the server.

- ServerManager sends process to destination client.

- At destination client, ClientManager resumes the process and starts it in a new thread.

Implementation :

Source Code:
https://drive.google.com/open?id=166FoXkpqAC5Gv6fl8s9DVbWtKAP3oHxZ

Deploy and Run :

1) Open three Terminals In one Terminal, type “server” to become a server. And you will get an echo showing the default IP and port (localhost:1234) .

2) In the other two, type client src_ip scr_port and they will connect to that server.

3) In Client A, type “create NonIOProcess” to create a migratable process without IO operations.

4) In Client B, type “create IOProcess input.txt output.txt” to create a migratable process with IO operations.
   (*Note: input.txt and output.txt should have absolute path.)

5) In server, type “ps” to show all the running processes on all clients. In the two clients, type “ps” to show all the local running processes.

6) In server, type “migrate 1 0 0” to migrate the IO process (pid 0) on Client B (cid 1) to Client A (cid 0).

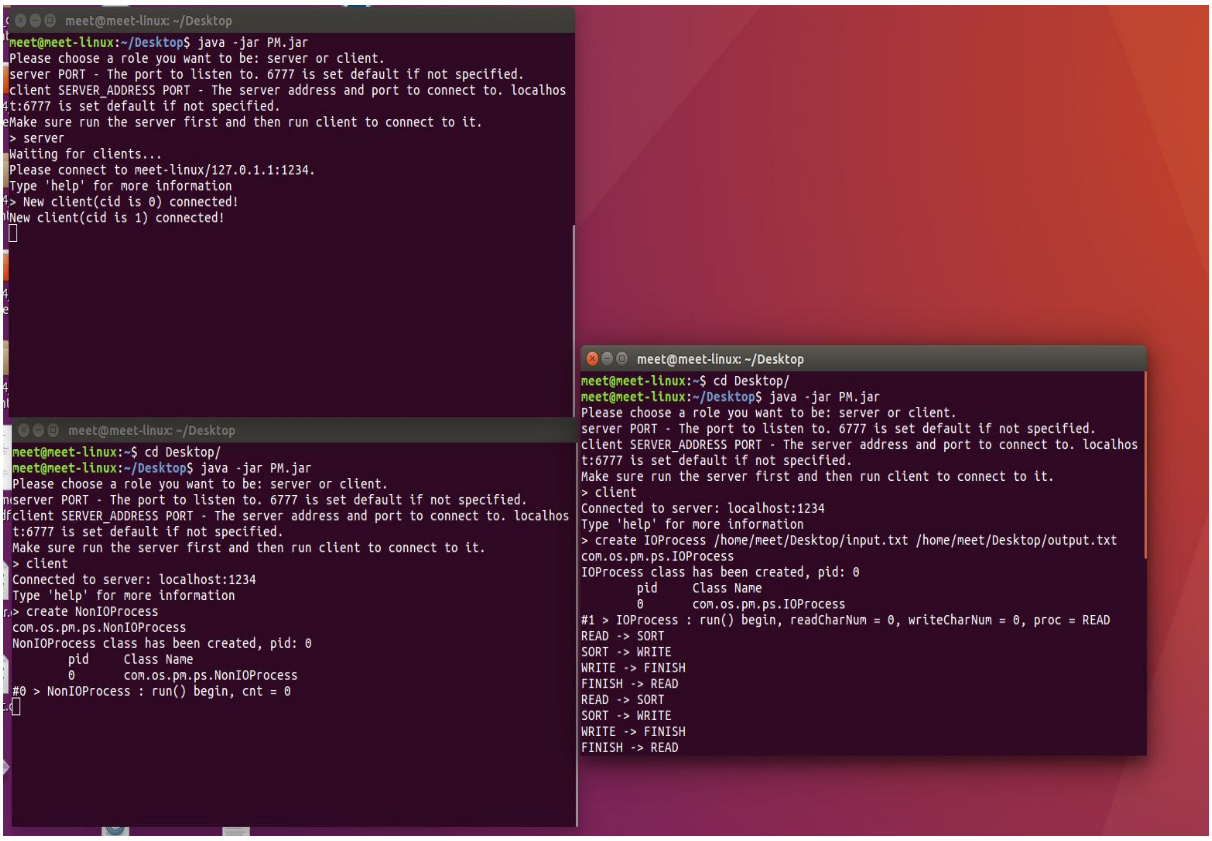
7) In server, type “migrate 0 0 1” to migrate the non-IO process (pid 0) on Client A (cid 0) to Client B (cid 1).

8) In server, type “ps” to show all the running process on all clients after migration. In the two clients, type “ps” to show all the local running processes.

9) Wait for IOProcess to finish on Client A. When it shows “JOB FINISHED”, the IOProcess finishes all the IO operations.

10) In server, type “exit” to exit server. All other clients connected to this server will exit automatically.

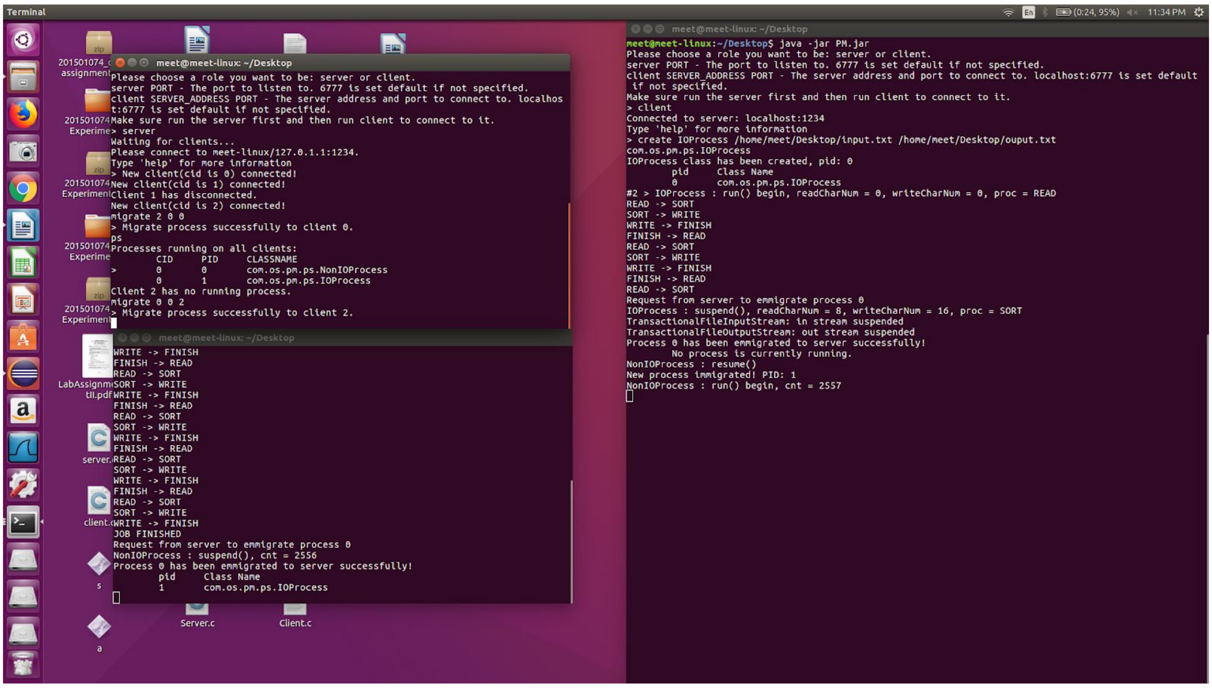
ScreenShots :


meet@meet-linux: ~/Desktop
meet@meet-linux: ~/Desktop
meet@meet-linux: ~/Desktop

```
meet@meet-linux:~/Desktop$ java -jar PM.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
New client(cid is 1) connected!
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar PM.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create NonIOProcess
com.os.pm.ps.NonIOProcess
NonIOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.NonIOProcess
#0 > NonIOProcess : run() begin, cnt = 0
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar PM.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create IOProcess /home/meet/Desktop/input.txt /home/meet/Desktop/output.txt
com.os.pm.ps.IOProcess
IOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.IOProcess
#1 > IOProcess : run() begin, readCharNum = 0, writeCharNum = 0, proc = READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
```



```
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
New client(cid is 1) connected!
Client 1 has disconnected.
New client(cid is 2) connected!
migrate 2 0 0
> Migrate process successfully to client 0.
ps
Processes running on all clients:
        CID     PID     CLASSNAME
>       0       0       com.os.pm.ps.NonIOProcess
        0       1       com.os.pm.ps.IOProcess
Client 2 has no running process.
migrate 0 0 2
> Migrate process successfully to client 2.
```

```
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
JOB FINISHED
Request from server to emmigrate process 0
NonIOProcess : suspend(), cnt = 2556
Process 0 has been emmigrated to server successfully!
        pid     Class Name
        1       com.os.pm.ps.IOProcess
```

```
meet@meet-linux:~/Desktop$ java -jar PM.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhost:6777 is set default
 if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create IOProcess /home/meet/Desktop/input.txt /home/meet/Desktop/ouput.txt
com.os.pm.ps.IOProcess
IOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.IOProcess
#2 > IOProcess : run() begin, readCharNum = 0, writeCharNum = 0, proc = READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
SORT -> WRITE
WRITE -> FINISH
FINISH -> READ
READ -> SORT
Request from server to emmigrate process 0
IOProcess : suspend(), readCharNum = 8, writeCharNum = 16, proc = SORT
TransactionalFileInputStream: in stream suspended
TransactionalFileOutputStream: out stream suspended
Process 0 has been emmigrated to server successfully!
        No process is currently running.
NonIOProcess : resume()
New process immigrated! PID: 1
NonIOProcess : run() begin, cnt = 2557
```

Test Result :

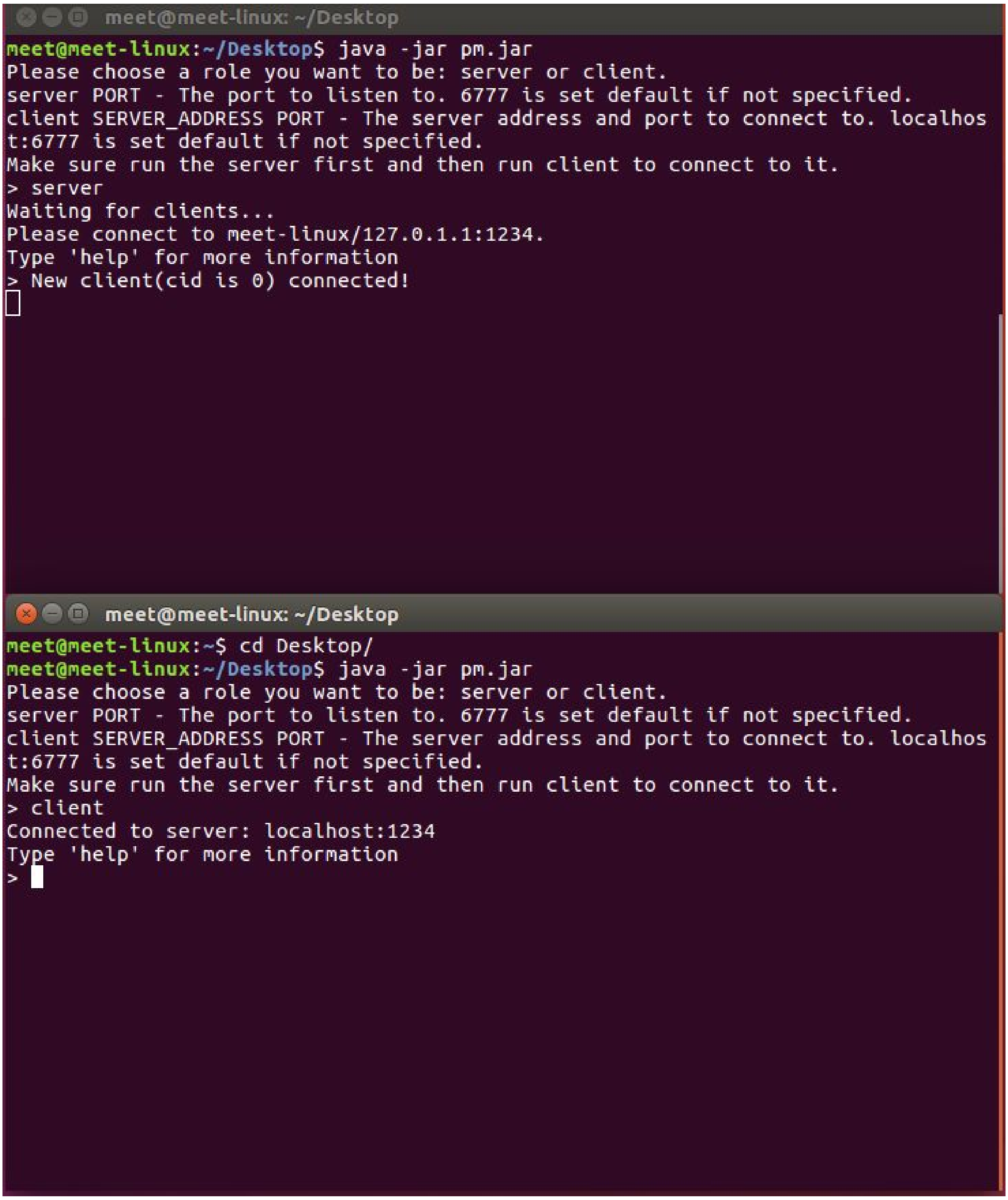
The result consist of three parts:

1. Migration status: In step 5, you can see the NonIOProcess running on Client A and IOProcess on Client B. In step 8, you can see NonIOProcess running on Client B and IOProcess on Client A. This means that the two processes have been migrated to each other.

2. Process status: In step 7, we can see an echo from NonIOProcess as following , “NonIOProcess : suspend(), cnt = xxx” on Client A. After migration, we can see an echo like “NonIOProcess : run() begin, cnt = yyy”, where yyy = xxx + 1. This means the process, suspended before migration resumes as expected, after migration.

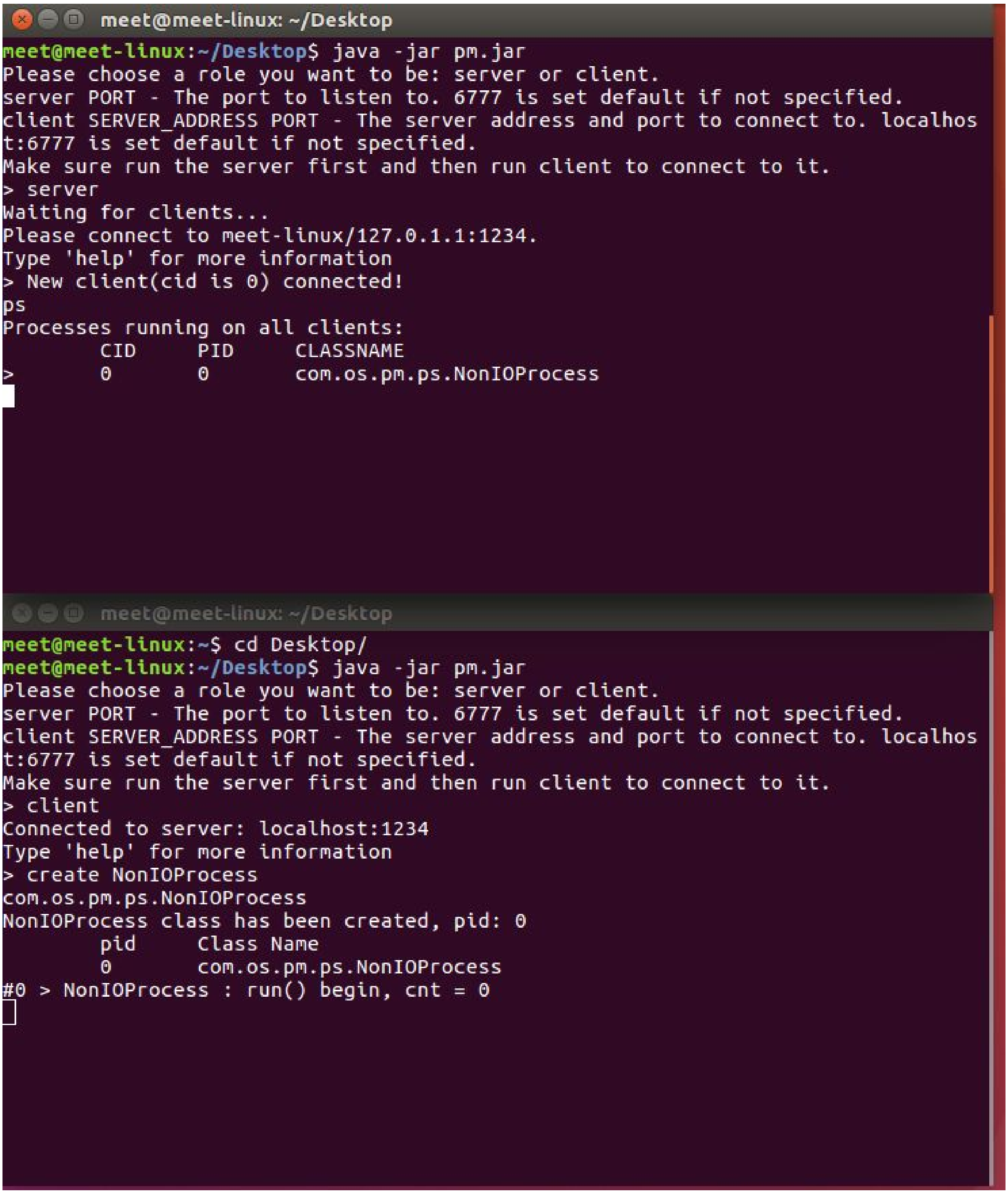
3. IO status: In IOProcess, we read a shuffled alphabet one character by one character from input.txt, sort them and write them to output.txt, and repeat this procedure 10 times. So, open output.txt, and you can see 10 set of ordered alphabets in it. This means the IO operations works fine during migration.

Other Screenshots (only NonIOProcess) :



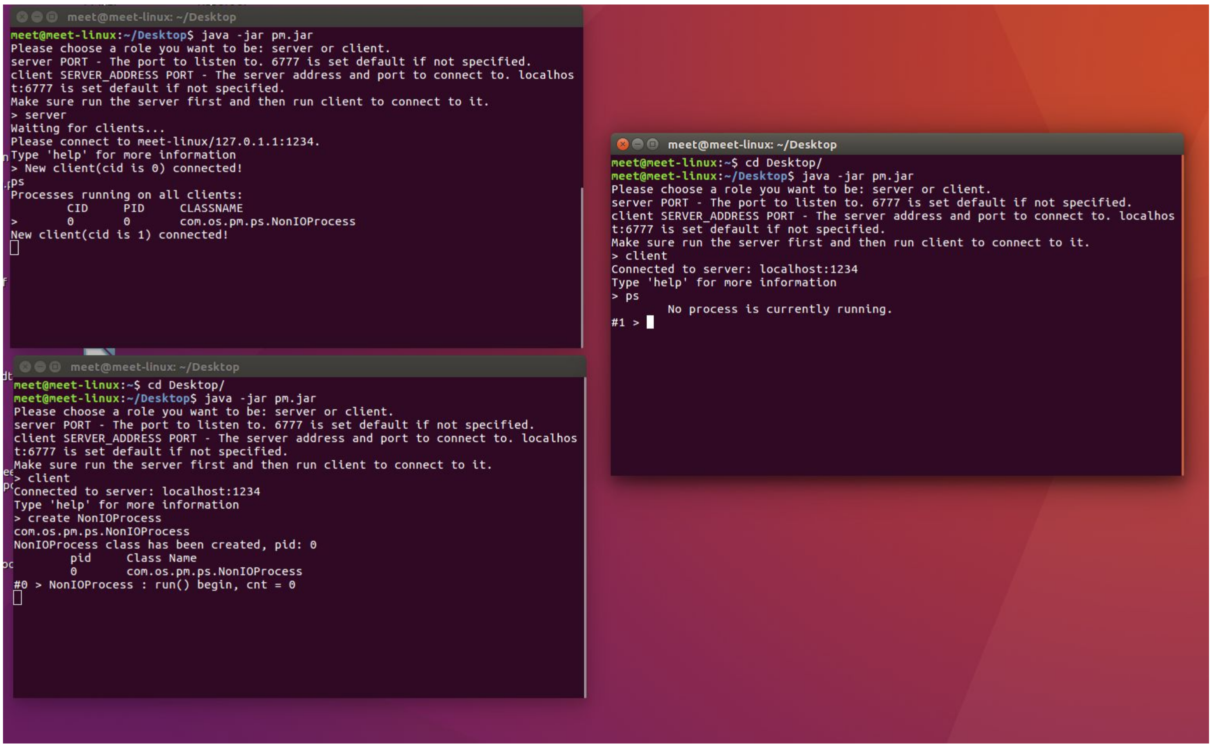
```
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> 
```



```
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
ps
Processes running on all clients:
        CID     PID     CLASSNAME
>       0       0       com.os.pm.ps.NonIOProcess
```

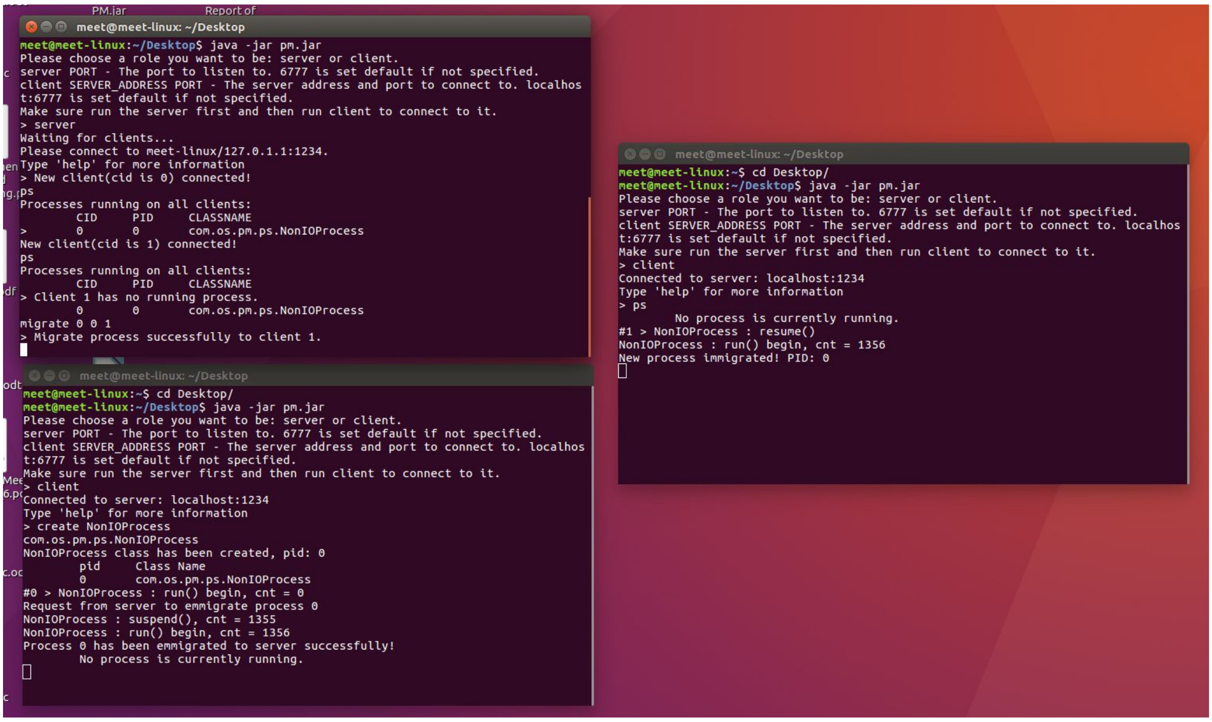
```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create NonIOProcess
com.os.pm.ps.NonIOProcess
NonIOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.NonIOProcess
#0 > NonIOProcess : run() begin, cnt = 0
```



```
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
ps
Processes running on all clients:
        CID     PID     CLASSNAME
>       0       0       com.os.pm.ps.NonIOProcess
New client(cid is 1) connected!
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create NonIOProcess
com.os.pm.ps.NonIOProcess
NonIOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.NonIOProcess
#0 > NonIOProcess : run() begin, cnt = 0
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> ps
        No process is currently running.
#1 >
```



```
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> server
Waiting for clients...
Please connect to meet-linux/127.0.1.1:1234.
Type 'help' for more information
> New client(cid is 0) connected!
ps
Processes running on all clients:
        CID     PID     CLASSNAME
>       0       0       com.os.pm.ps.NonIOProcess
New client(cid is 1) connected!
ps
Processes running on all clients:
        CID     PID     CLASSNAME
> Client 1 has no running process.
        0       0       com.os.pm.ps.NonIOProcess
migrate 0 0 1
> Migrate process successfully to client 1.
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> create NonIOProcess
com.os.pm.ps.NonIOProcess
NonIOProcess class has been created, pid: 0
        pid     Class Name
        0       com.os.pm.ps.NonIOProcess
#0 > NonIOProcess : run() begin, cnt = 0
Request from server to emmigrate process 0
NonIOProcess : suspend(), cnt = 1355
NonIOProcess : run() begin, cnt = 1356
Process 0 has been emmigrated to server successfully!
        No process is currently running.
```

```
meet@meet-linux:~$ cd Desktop/
meet@meet-linux:~/Desktop$ java -jar pm.jar
Please choose a role you want to be: server or client.
server PORT - The port to listen to. 6777 is set default if not specified.
client SERVER_ADDRESS PORT - The server address and port to connect to. localhos
t:6777 is set default if not specified.
Make sure run the server first and then run client to connect to it.
> client
Connected to server: localhost:1234
Type 'help' for more information
> ps
        No process is currently running.
#1 > NonIOProcess : resume()
NonIOProcess : run() begin, cnt = 1356
New process immigrated! PID: 0
```

References :

-> GeeksforGeeks - A computer science portal for geeks (For Java and its API related queries)
-> javatpoint & tutorials point (For Java related queries)
-> StackOverflow (For Socket Programming related queries)
-> Operating Systems: Internals and Design Principles, 7th Edition by William Stallings.