Report

The *client-server* interaction was simulated using Scala Actors. In my implementation I defined the following actors:

- ***Server***: executes the median filter algorithm on demand on the image received in the message
- ***Client***: creates server sub-actor when receives a message to simulate a "request" to the server and process different images concurrently

```
Client Actor --> Server SubActors
                          |
                           - - -> Serial Execution
                          |
                          - - -> Concurrent Execution
```

The algorithm functionality is encapsulated in the singleton object ***MedianFilter***. This contains a method for serial execution and another with an optimized version using Scala concurrent library. These functions are used by the server to process the images.

The optimized approach benefits from Scala concurrency using Futures. The idea is to apply the median filter algorithm in different pieces of the image at the same time. This implementation processes the image in 4 partitions, it waits for all the partitions to be concluded and then it writes the resulting image.

Execution Time Results:

- Serial execution: 1.1184684 sec
- Concurrent execution: 0.5814112 sec

There is a difference of 0.53 sec between the two approaches, therefore we can conclude that there is a 52% improvement in performance when using concurrency.