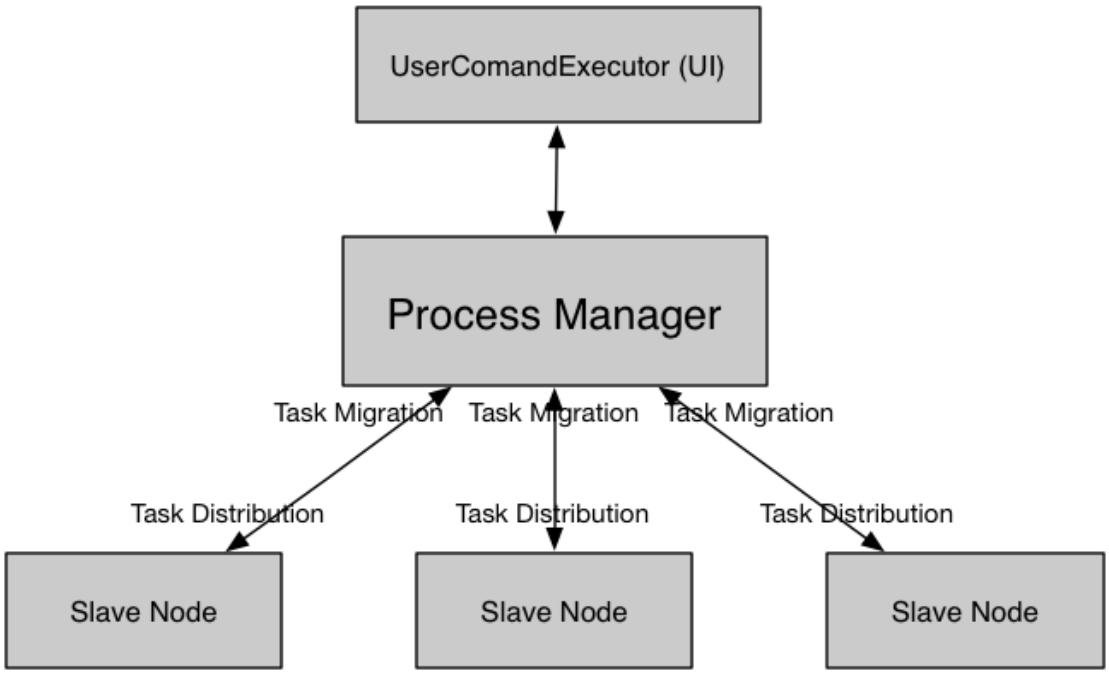
Report

Note: For an overall system description please refer to /description.txt. Review /manul.txt for the usage description of the framework.

1. Architecture

The overall design of the system is the same as the following diagram.



A UI program is taking the user input and send it to the process manager. The process manager is polling for user commands. There is an arbitrary number of slaves running that also actively listen to the process manager's command.

2. Design Choice and implementation

We chose polling architecture over event-driven as is more direct and clear. To implement this, the process manager and every slave node are all running a ServerSocket that is actively listening for others to connect. When a connection is opened, the manager/slave takes the commands from the connection and behave accordingly. The migratable process is sent between manager and slaves by sockets and can thus be executed elsewhere. When the manager receives a "migrate" command it randomly migrates the process to a slave other than the slave that originally runs the process.

3. Our own migratable objects

We have created two additional migratable objects. The FactorialProcess takes numbers from an input file and calculate the product of the numbers. The ReverseWordList process reverse the words

within a line from a input file and output it to a output file. The arguments of these two process are both <InputFile> <OutputFile>. When have provided sample files for running these process which are:

“in.txt, out.txt” for ReverseWordList

“numIn.txt, numOut.txt” for FactorialProcess

.

4. Own Testing

We have tested process migration on our own migratable processes and the given GrepProcess and they are successfully migrated between slaves.

5. Your Testing

Please refer to manual.txt for instructions on framework testing.