RSYNC; OCTAVE + GNU PARALLEL; HTOP TUTORIAL

1) **RSYNC** is common easy to use package for synchronizing content of files and directories. Using rsync one can send only differences between files instead of whole information – it reduces used network bandwidth and speeds up information spreading across different knots.

2) synchronizing current directory with remote directory (with ssh_keygen one can do it via commands) - upload data on remote:

```
#rsync -u -v –progress -e ssh * user@IPv4:/home/user/remotePathToDirectory
```

3) synchronizing some directory with remote directory - upload data on remote:

```
#rsync -u -v –progress -e ssh /home/user/localPathToDirectory
user@IPv4:/home/user/remotePathToDirectory
```

3) synchronizing remote directory with some local directory – download data from remote:

```
#rsync -u -v –progress -e ssh user@IPv4:/home/user/pathToDirectory
/home/user/localPathToDirectory
```

1) **HTOP** is useful package for system monitoring via terminal:

```
#sudo apt-get install htop

#htop
```

2) closing system monitor is via F10 hotkey – please note, that it is configurable in some way.

1) one can combine **GNU PARALLEL** (please consider citation of authors publication) and **OCTAVE** to provide easy-to-program parallel execution script. There are some other solution like f.e. built-in Octave-parallel package, but it has some limitations. Personally I use those simplified receipt:

-on each computer there could be the same username with exchanged ssh-keys

-on each computer there could be the same directory path to data (if small enough), or mounted network folder (consider checking the RAID tutorial, and samba server tutorial; for high-density computation consider non-synchronous start of program commads for better basic network balance)

-provide tested M-script f.e. for processing single image and rsync it on each computer

-profile script execution (f.e. use htop) - especially if there are some memory – space costly operations for calculation of possible parallel executions on each computer (some computer could have smaller number of cores per avaiable memory for that partiular script execution)

-write M – script (or use BASH commands-script) for generating each command text string; processing single file with arguments - commandsOctave.txt

```
octave-cli
        -p /home/user/pathToScriptDirectory
        -p /home/user/pathToSomeOtherUsedByScriptDirectory
        --eval ”scriptName( ‘argFilepath1’, argumentScalar1 )”; pause( 0.1 );

octave-cli
        -p /home/user/pathToScriptDirectory
        -p /home/user/pathToSomeOtherUsedByScriptDirectory
        --eval ”scriptName( ‘argFilepath2’, argumentScalar2 )”; pause( 0.1 );
```

-generate each computer address list with coresponding CPU physical cores; homeClusterList.txt – 2 cores on local computer and 12 cores on remote computer:

```
2/:

12/user@IPv4
```

-populate gnu parallel with commands file and server list; use all cores(for 3 cores use : -j3) save results to logFile.txt:

```
#parallel --slf /user/home/pathToDirectory/homeClusterList.txt -j+0 <
commandsOctave.txt &>> logFile.txt
```

P.S. There will be better L3 memory size per core if one disable HyperThreading technology on Intels CPU’s. Benefits are only in some data processing situations.