ie
SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY

OPERATING SYSTEMS & PARALLEL COMPUTING

MacOS Examples

You need help ?

- **The Linux equivalent of HELP is `man` (manual)**
  - Use `man -k <keyword>` to find all commands with that keyword
  - Use `man <command>` to display help for that command
    - Output is presented a page at a time. Use `b` for to scroll backward, `f` or a space to scroll forward and `q` to quit

Common command

- `pwd` - **print (display) the working directory**
- `cd <dir>` - **change the current working directory to *dir***
- `ls` - **list the files in the current working directory**
- `ls -l` - **list the files in the current working directory in long format**

- `who` **or** `w`
  - List who is currently logged on to the system
- `whoami`
  - Report what user you are logged on as
- `ps`
  - List your processes on the system
- `ps aux`
  - List all the processes on the system
- `echo "A string to be echoed"`
  - Echo a string (or list of arguments) to the terminal

Who's Logged On Right Now?

- The `w` command lists all users logged on right now

```
 5:16pm  up 2 days,  8:46,  1 user,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU  WHAT
neale    ttyp0    websurfer.reston  4:28pm  1.00s  0.52s  0.18s  w
```

Execute Next commands

- Get help on the `ls` command
- Find out who else is on the system
- What is your current directory
- Redirect the output of the `ls -l /` command to `ls.output` and see what you get

Execute Process Command

- **ps -ef | more**
- **ps aux**
- **ps –e f** **NOT WORKING in MacOS**
- **top (tape “q” to quit the process)**

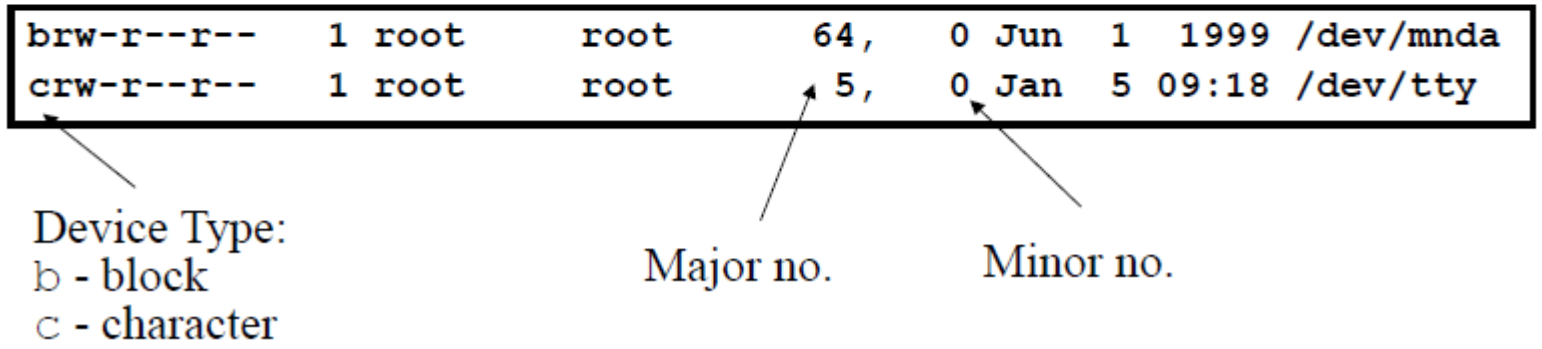
Linux Device Handling

- **Devices are the way Linux talks to the world**
- **Devices are special files in the `/dev` directory (try `ls /dev`)**

/dev/ttyx	TTY devices
/dev/hdb	IDE hard drive
/dev/hdb1	Partition 1 on the IDE hard drive
/dev/dasda	ECKD/CKD/FBA DASD
/dev/dasda1	Partition 1 on DASD
/dev/null	The null device (“hole”)
/dev/zero	An endless stream of zeroes
/dev/mouse	Mouse (not /390)

Device and Drivers

- **Each `/dev` file has a major and minor number**
  - Major defines the device type
  - Minor defines device within that type
  - Drivers register a device type



Special Files - /proc

- **Information about internal Linux processes are accessible to users via the `/proc` file system (in memory)**

/proc/cpuinfo	**CPU Information**
/proc/interrupts	**Interrupt usage**
/proc/version	**Kernel version**
/proc/modules	**Active modules**

NOT WORKING in MacOS

```
cat /proc/cpuinfo
vendor_id       : IBM/S390
# processors    : 1
bogomips per cpu: 86.83
processor 0: version = FF, identification = 045226, machine = 9672
```

Check CPU Info

- sysctl -n machdep.cpu.brand_string
- sysctl -a | grep machdep.cpu
- sysctl -a | grep machdep.cpu | grep core_count
- sysctl -a | grep machdep.cpu | grep thread_count

File System

- **You can view what file systems are mounted using either:**
  - mount
  - df –h
  - cat /etc/fstab

mount

- Mounts a file system that lives on a device to the main file tree
- Start at Root file system
  - Mount to root
  - Mount to points currently defined to root
- `/etc/fstab` used to establish boot time mounting

```
/dev/dasda1     /              ext2    defaults,errors=remount-ro 0 1
/dev/dasdb1     /bin           ext2    defaults,errors=remount-ro 0 1
/dev/dasdc1     /usr           ext2    defaults,errors=remount-ro 0 1
/dev/dasdd1     /usr/local     ext2    defaults,errors=remount-ro 0 1
/dev/dasde1     /usr/man       ext2    defaults,errors=remount-ro 0 1
/dev/dasdf1     /home          ext2    defaults,errors=remount-ro 0 1
/dev/dasdg1     swap           swap    defaults    0       0
none            /proc          proc    defaults    0       0
```

Environment Variables

- **Using Environment Variables:**
  - `echo $VAR`
  - `cd $VAR`
  - `cd $HOME`
  - `echo "You are running on $SYSTEMNAME"`
- **Displaying - use the following commands:**
  - `set` (displays local & environment variables)
  - `export`
- **Variables can be retrieved by a script or a program**

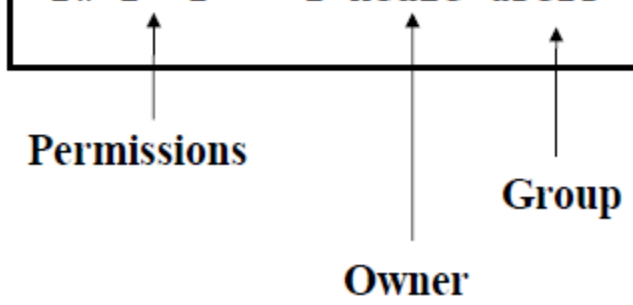
Creating file and directories

- **Files can be created in a number of ways**
  - The output of a command
  - Being edited using vi or your favorite editor
  - By using the `touch` command which creates an empty file or updates the modification and access time information of an existing file
- **Directories are created using the `mkdir` command**

File Permissions

- **The long version of a file listing (`ls -l`) will display the file permissions:**

```
-rwxrwxr-x  1 rvdheij  rvdheij      5224 Dec 30 03:22 hello
-rw-rw-r--  1 rvdheij  rvdheij       221 Dec 30 03:59 hello.c
-rw-rw-r--  1 rvdheij  rvdheij      1514 Dec 30 03:59 hello.s
drwxrwxr-x  7 rvdheij  rvdheij      1024 Dec 31 14:52 posixuft
:
-rw-r--r--  1 neale users       1039 2009-09-10 12:47 a.a
drwxr-xr-x  5 neale users       4096 2011-08-16 20:34 benchmark
drwxr-xr-x  2 neale users       4096 2009-07-30 08:55 bin
drwxr-xr-x  3 neale users       4096 2009-05-16 12:17 BINUTILS
-rw-r--r--  1 neale users       3776 2012-02-24 09:32 bluefin.cs
```



File Commands

- `cp <fromfile> <tofile>`
  - Copy from the <fromfile> to the <tofile>
- `mv <fromfile> <tofile>`
  - Move/rename the <fromfile> to the <tofile>
- `rm <file>`
  - Remove the file named <file>
- `mkdir <newdir>`
  - Make a new directory called <newdir>
- `rmdir <dir>`
  - Remove an (empty) directory

Change File Permissions

- **Use the `chmod` command to change file permissions**
  - The permissions are encoded as an octal number

User			Group			Other		
Read r	Write w	Execute x	Read r	Write w	Execute x	Read r	Write w	Execute x
400	200	100	40	20	10	4	2	1

```
chmod 0755 file  # Owner=rwx Group=r-x Other=r-x
chmod 0500 file2 # Owner=r-x Group=--- Other=---
chmod 0644 file3 # Owner=rw- Group=r-- Other=r--

chmod +x  file   # Add execute permission to file for all
chmod u-r file   # Remove read permission for owner
chmod a+w file   # Add write permission for everyone
```

More Commands

- `awk` - a file processing language that is well suited to data manipulation and retrieval of information from text files
- `chown` - sets the user ID (UID) to owner for the files and directories named by pathname arguments. This command is useful when from test to production

```
chown -R apache:httpd /usr/local/apache
```

More Commands

- `diff` - attempts to determine the minimal set of changes needed to convert a file specified by the first argument into the file specified by the second argument
- `find` - Searches a given file hierarchy specified by path, finding files that match the criteria given by expression

Search Command

- `grep` - **Searches files for one or more pattern arguments. It does plain string, basic regular expression, and extended regular expression searching**

```
find ./ -name "*.c" | xargs grep -i "fork"
```

In this example, we look for files with an extension "c" (that is, C source files). The filenames we find are passed to the xargs command which takes these names and constructs a command line of the form: `grep -i fork <file.1>…<file.n>`. This command will search the files for the occurrence of the string "fork". The "-i" flag makes the search case insensitve.

Kill Process

- **kill - sends a signal to a process or process _group**
  - You can only kill your own processes unless you are root

```
UID         PID   PPID   C STIME TTY          TIME CMD
root       6715   6692   2 14:34 ttyp0    00:00:00 sleep 10h
root       6716   6692   0 14:34 ttyp0    00:00:00 ps -ef
[root@penguinvm log]# kill 6715
[1]+  Terminated              sleep 10h
```

Replace String

- **sed - applies a set of editing subcommands contained in a script to each argument input file**

```
find ./ -name "*.c,v" | sed 's/,v//g' | xargs grep "PATH"
```

This finds all files in the current and subsequent directories with an extension of c,v. sed then strips the ,v off the results of the find command. xargs then uses the results of sed and builds a grep command which searches for occurrences of the word PATH in the C source files.

Archive command

- **`tar` - manipulates archives**
  - An archive is a single file that contains the complete contents of a set of other files; an archive preserves the directory hierarchy that contained the original files.

```
tar -tzf imap-4.7.tar.gz
imap-4.7/
imap-4.7/src/
imap-4.7/src/c-client/
imap-4.7/src/c-client/env.h
imap-4.7/src/c-client/fs.h
```

Viewing Files

- `cat` **"Concatenate"**
- `more` **Display one page at a time**
- `less` **Variant of** `more`
- **Editors**
  - `vi` Visual editor, the default
  - `the` XEDIT/KEDIT/ISPF clone
  - `xedit` X windows text editor
  - `emacs` Extensible, Customizable Self-Documenting Display Editor
  - `pico` Simple display-oriented text editor
  - `nedit` X windows Motif text editor

Examples Thread & Processes with Python

- Test test_thread.py
- Test test_thread2.py
- Test test_process1.py
- Test test_process2.py

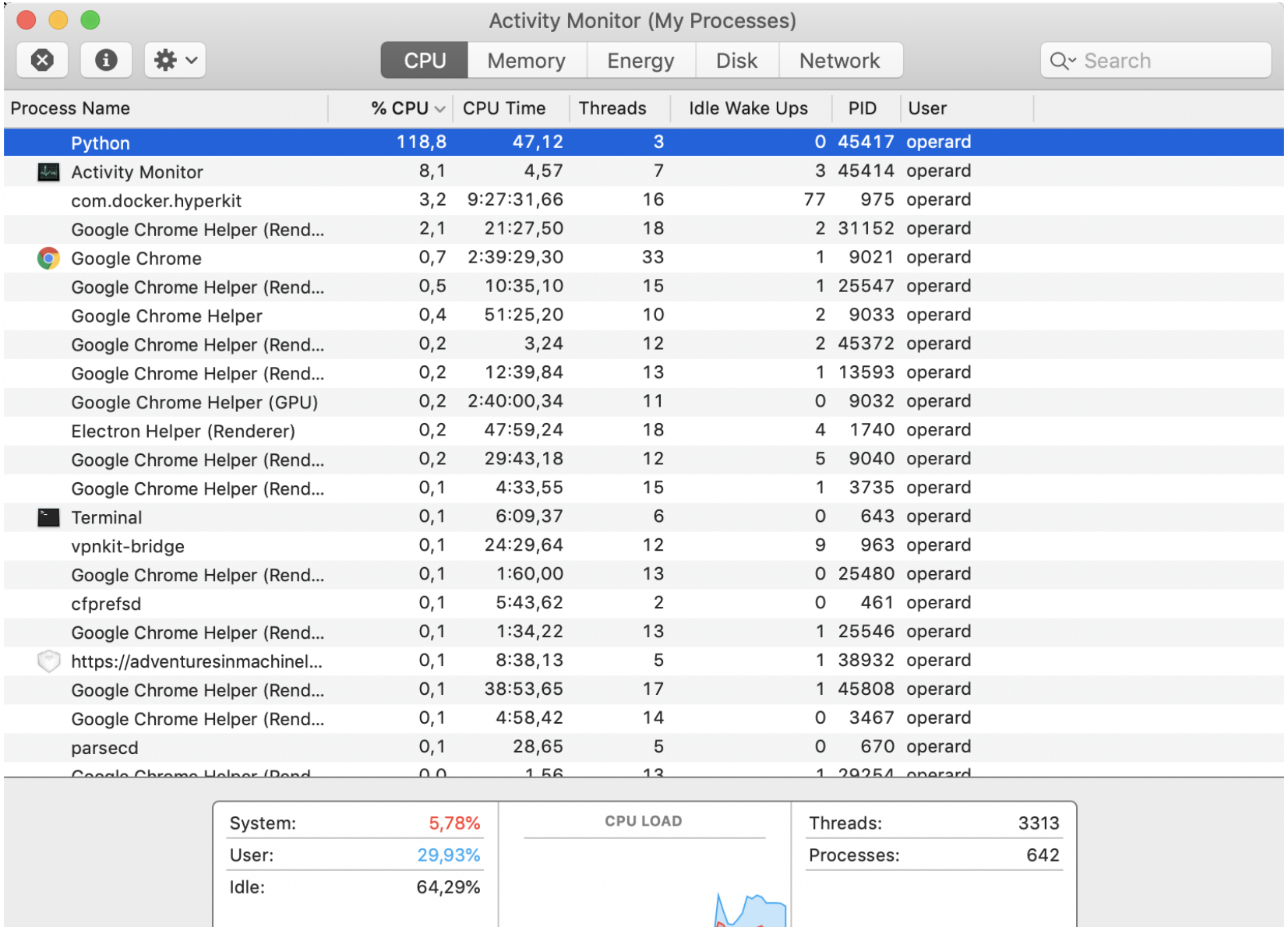
- Which conclusions ?

Monitoring Example: cluster_analysis.py

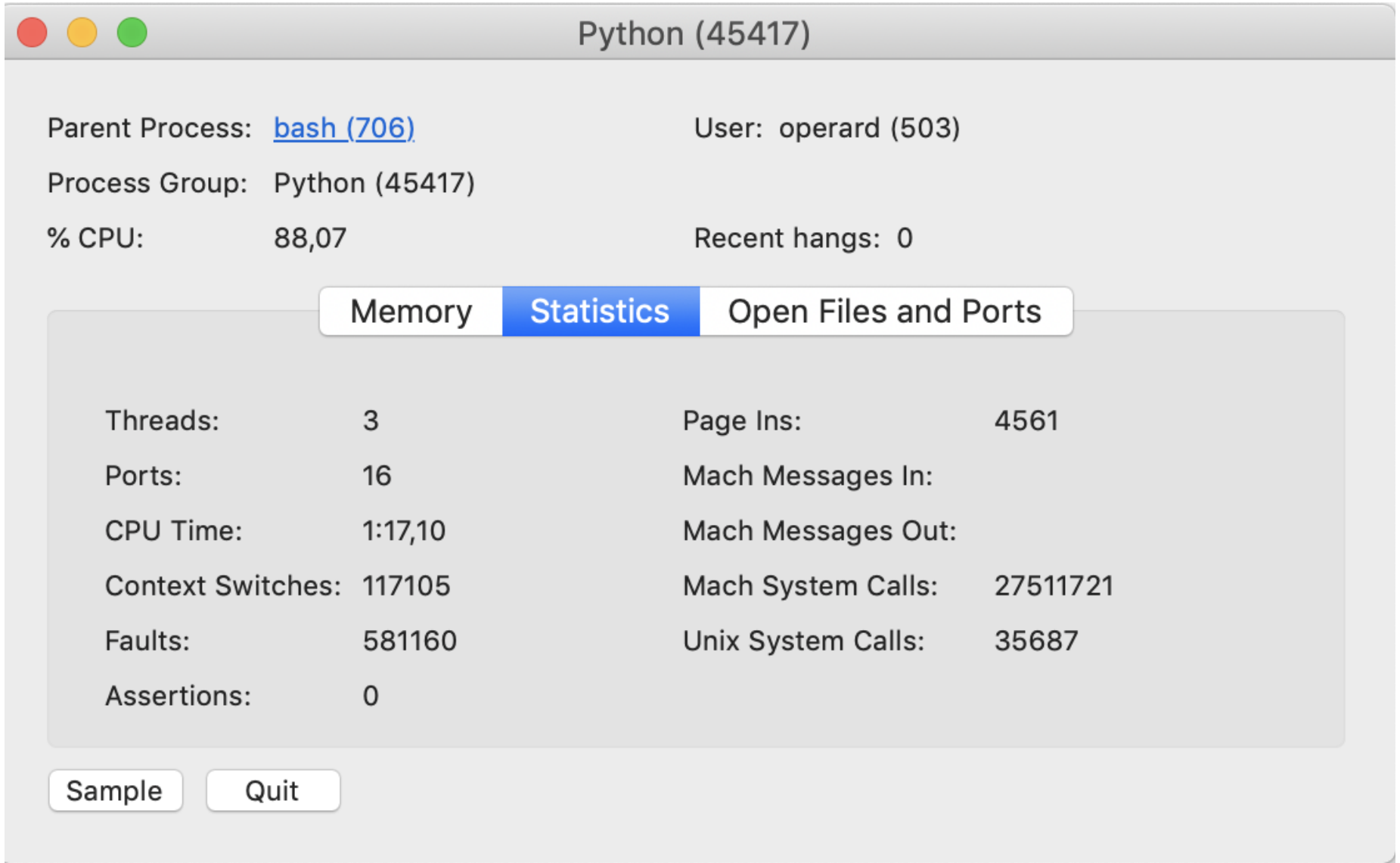
- Use Activity Monitor
- Or use “htop”: htop –t –p <PID>

- Installation/configuration python3:
  - python3 -m pip install matplotlib
  - python3 -m pip install scikit-learn
  - python3 -m pip install pandas
  - time python3 cluster_analysis.py
  -
  - brew install htop
  -

Activity Monitor: cluster_analysis.py



Activity Monitor: cluster_analysis.py



Execution in OEL7

- time python3 cluster_analysis.py
  (500000, 33)
  Execution Time: 0 hour:1 min:51 sec

  real 2m4.101s
  user 1m55.106s
  sys 0m18.814s

SCHOOL OF
HUMAN SCIENCES
& TECHNOLOGY