The `gnuplot` tool is used to plot all figures in the papers. For each script `.gnu` provided, the figure is obtained by doing `gnuplot script.gnu`. Gnuplot is easily installed on Ubuntu as followed:

```
sudo apt update
sudo apt install gnuplot
```

Memory Expes

> (expes/mem_conso)

- script `data_cdf/run.sh` (note: `run_local.sh` allows to collect data from the host without the VM, this is possible since the hypercalls do not influence the memory consumption) allows to collect data on working set and security distances for all the applications and all the frequencies, and each result is placed in ../data_cdf/freq_$f/$app_output
- script `data_cdf/extract_wss.sh` retrieves the wss for each app an each technique an places the output in data_cdf/plot/$app_wss
- script `data_cdf/extract_cdf.sh` gives the cdf data for each freq in data_cdf/freq_$f/plot/$app_cdf
- script `data_cdf/extract_pattern.sh`: data_cdf/plot/subpages contains the raw data for blackscholes and each frequency; the script extracts from these files, the pattern and number of subpages for each pattern and places the cdf in data_cdf/plot/distrib_patternxx

Security Distances

> (expes/mem_conso/data_cdf)

- The script `cdf.gnu` plots in **Fig. 8** the CDF of the security distances depending on the protection policy. Data used are in ./distance_secu_slim_gp_default/ for the default policy, and in ./freq_$i/ folders for our custom policy (*i* represents the protection frequency).

Memory Consumption

> (expes/mem_conso/.)

- `wss.data` summarizes data from ./freq_$i/ that are used by `conso2.gnu` to plot **Fig. 9**.

Optimum

> (expes/mem_conso/optimum/)

- From Fig.9, we get the optimum frequencies for each app, and compile Slimguard with each freq. `optimum.data` contains the summarized data and `optimum.gnu` plots **Fig. 10**

Perfs Expes

> (expes/perfs)

Basic Costs

> (expes/perfs/basic costs)

- ./KONE-GP/ contains the file with the raw data for the mmap time (a simple grep | awk allows to compute the mean)
- ./LeanGuard/mmap_madvise/ contains the mmap and madvise raw times for LeanGuard (**Table 1**)
- Data are summarized in `basic_costs.data` and `basic_costs.gnu` allows to plot the Average execution time of malloc() in **Figure 11**

Microbench

> (expes/perfs/with_hypercalls_subinfo/microbench)

- Here, we collect the time of hypercalls and page faults, with (`./Xen_optimized`) and without (`./Xen_notoptimized`) page walk optimization in Xen . Summarized data are in `worse_both.data` and the secript `worse.gnu` plots in **Fig.12** the Time of page fault handling w and w/o optimization.
- Raw data are in `./worse & ./best` folders, of `/Xen_optimized & ./Xen_notoptimized` dir, obtained by doing a `grep worse/best` from files $i in each forlder. Each $i represents the size' order of the pool
- The script `median.awk` allows to compute the median of data conatined in each file using `awk -f median.awk $file`

Execution times

> (expes/perfs/with_hypercalls_subinfo/exec_times)

To evaluate the overhead of LeanGuard compared to SlimGuard, we measure the execution time of PARSEC applications under Slim and Lean; for Lean, we leave the hypercalls in Linux, but we comment, in Xen, their corresponding code (to avoid the emulation). Data results for each allocator are in $allocator/$app_time. `overhead.data` summarizes the data and the script `overhead.gnu` plots the Performance overhead in **Fig.13**.

Concurrency

> (expes/perfs/with_hypercalls_subinfo/concurrency)

We measure the time of page fault handling (with SPP emulation) for each application when they run alone (see ../parsec_apps/PFs/), and when they run concurrently with others. Each folder $i contains the raw data for each concurrent application involved in the experiment (*i* is the number of concurrent applications run).

The mean page fault time of each application, when running alone, is computed in `../parsec_apps/PFs/$app_PF`, and the mean page fault time when running concurrently is in each folder ./$i. Grouped data are in `concurrency.data` and the script `concurrency.gnu` plots, in **Fig.14**, the overhead of varying the number of concurrent applications.

Background Expes

> (expes/background)

In data/distances/, we have the average security distances for each protection frequency and for each Slimguard class.

In data/freq_1/ and data/freq_others/, we have the raw data of the experiments, that give the real and theroretical mem consumption while varying the protection frequency.

- `blackscholes.data` summarizes the results for the blackscholes app, and `blackscholes.gnu` plots **Fig. 3**
- `parsec_mem_waste.data` summarizes the results for the memory waste of all applications and `parsec_mem_waste.gnu` plots **Fig. 2**