Working Set Expes

> (expes/mem_conso)

- In Fig.9 we get the optimum frequencies for each app, and compile slimguard with each freq and generate lib_app.so
- scritp `wss/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 wss/freq_$f/$app_output
- script `wss/extract_wss.sh` retrieves the wss for each app an each technique an places the output in wss/plot/$app_wss (*conso2.gnu ==>Fig.9*)
- script `wss/extract_cdf.sh` gives the cdf data for each freq in wss/freq_$f/plot/$app_cdf (*cdf.gnu ==>Fig.8*)
- script `wss/extract_pattern.sh`: in wss/plot/subpages we have raw data for blackscholes and each frequency, the script extracts from these files, the pattern and number of subpages in each pattern and places the cdf in wss/plot/distrib_patternxx (pattern.gnu)

Perfs Expes

> (expes/perfs) ==> ./

Microbench

> (expes/perfs/with_hypercalls/microbench)

- Here, we collect the time of hypercalls and page fault, with (`./Xen_optimized`) and without (`./Xen_notoptimized`) page walk optimization in Xen (*worse.gnu ==> Fig.12 in §5.3-SPP Table Walk Optimization*)
- raw data are in `./wore & ./best` 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 a data into a given file using `awk -f median.awk $file`

Execution times

> (expes/perfs/with_hypercalls/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). Result for each allocator are in $allocator/$app_time (*overhead.gnu ==> Fig.13*)

Concurrency

> (expes/perfs/with_hypercalls/concurrency)

We measure the time of page fault handling (with leave activated the emulation) for each application when they run alone (see ./parsec/), and when they run in concurrency with other ones. Each folder $i contains the raw data for each concurrent application involved in the experiment.

What we do is computing the mean page fault time of each application in `./parsec/$app`, and then compute their mean page fault time in each folder $i, and finally compute the overhead (*concurrency.gnu ==> Fig.14*)