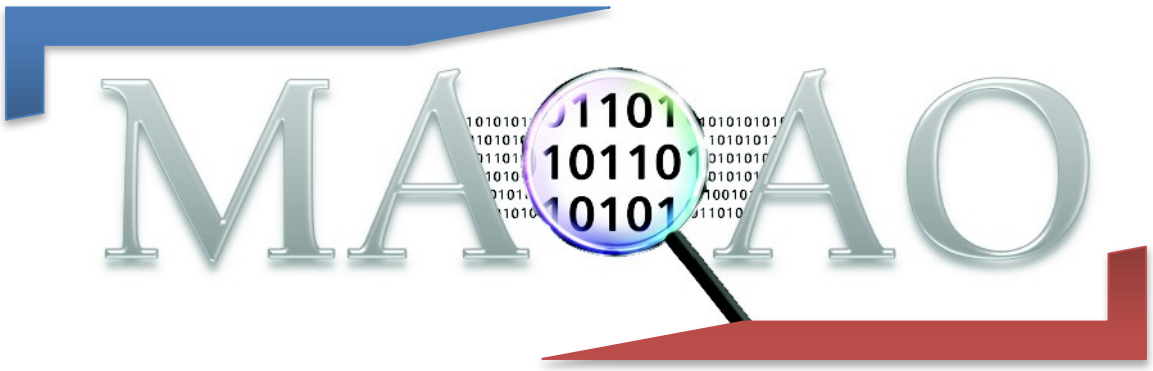


LProf

Doc version : 2.4.0, July 2026

1 Introduction

MAQAO Lightweight Profiler (LProf) is the MAQAO module which allows to easily profile an application to detect hot functions and loops in two steps:

1) Data collection using sampling

LProf uses hardware counters to profile large-scale parallel applications (2,000+ cores) with a very low overhead. A stress test showed that LProf could profile a code running on 1,000 nodes with 112 cores each (112,000 threads total).
It is also possible to provide a custom list of hardware counters to sample.

2) Data display

LProf output allows to quickly identify time-consuming functions and loops, observe the amount of time spent by the application between different categories (I/O, Runtime, etc…) and detect load balancing issues.

2 Running MAQAO LProf

2.1 Sequential Run Command

```
maqao lprof -- <application> [arg1 arg2 ...]
```

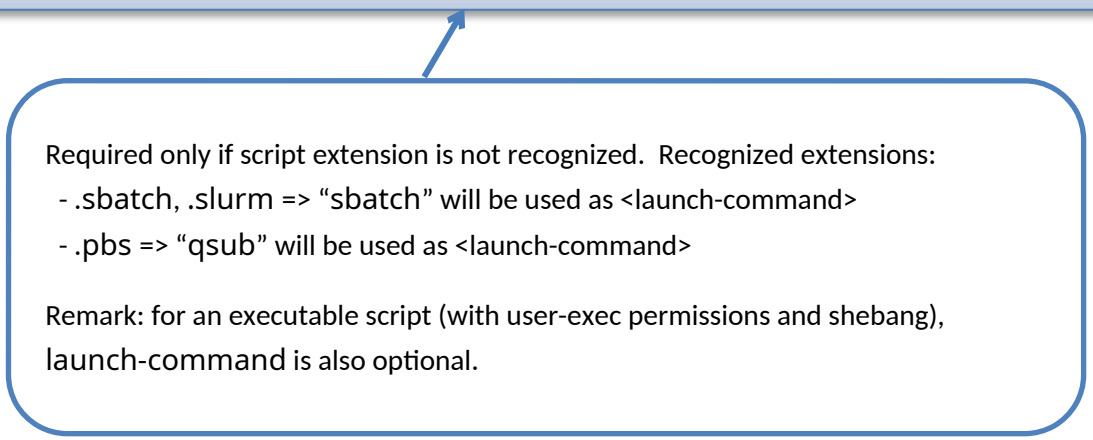
2.2 Parallel Run Command

Interactive runs:

```
maqao lprof --mpi-command="mpirun –n <NB_PROCESSES>" \
-- <application> [args]
```

Runs with launch script (typically to submit a job):

```
maqao lprof --launch-script=<script> [--mpi-command="mpirun..."] \
[--launch-command=<command to run launch-script>] -- <application>
```



In jobscript, application executable and its arguments have to be replaced by **<run_command>**.

```
$ cat jobscript.sh
…
mpirun -n 4 <run_command> # instead of mpirun -n 4
<application> [args]
# <mpi_command> <run_command> # if mpi-command used
```

For LProf to set correct internal settings, it is recommended to set **--maximum-processes-per-node** if known and > 1.

2.3 Kernel samples exclusion

Starting from 2025.2.x, kernel samples are collected by default if related permissions are granted (possible to force them excluded with `-exclude-kernel`). Previously that was the opposite, with `-include-kernel` to force collecting kernel samples.

To grant related persmissions:

- If sysctl kernel.perf_event_paranoid returns 2 or more, this step must be performed first:

```
$ sudo sysctl -w kernel.perf_event_paranoid=1
# lost after reboot
$ sudo sh -c ‘echo kernel.perf_event_paranoid=1 >>
/etc/sysctl.d/local.conf
# persists after reboot
```

- Alternative with Linux 5.8 or later:

```
$ sudo setcap cap_perfmon=ep `which maqao`
```

2.4 Options (collect step)

To list all options along with their descriptions:

```
maqao lprof --help
```

Options in gold color can be used to mitigate sampling overhead.
Options in light red can be used to override default behavior and workaround profiling issues.

Main options (collect step)		
Name	**Short Description**	**Values**
`--exclude-kernel`	No effect with the 'no-perf' engine. Force kernel sampling exclusion (avoids warning when missing permissions)	(no value)
`-mc/--mpi-command=`	Specify command for interactive MPI run or replacement value for <mpi_command> in job script	Ex: “`mpirun -n 4`”
`-ls/--launch-script=`	Script used to launch application. If it is executable (user-exec perm. and shebang), LProf directly executes it, otherwise Lprof executes <launch-command> <launch-script>	Path to script (string)
`-lc/--launch-command=`	Interpreter (e.g bash, python) or batch submission command to launch <launch-script> (as prefix). If omitted, guessed from	Ex: “`sbatch`”

	<launch-script> extension	
`--stdin-path`	Defines a file for redirection to stdin.	path to a stdin-redirection file
`--sampling-rate=`	Number of collected samples per second	• `highest` (2000 Hz, `btm=off` recommended) • `high` (1000 Hz, avoid `btm=stack`) • **`medium` (200 Hz, default)** • `low` (50 Hz) • `lowest` (10 Hz)
`-ldi/library-debug-info=`	Scan debug information into all or specified (provided list) library(ies) to get loops details	`on` (all) | **`dbg-inf` (default)** | `off` | list of libraries (`'lib1, lib2, …'`)
`--start-after-seconds=`	Setup sampling start delay	• **0 (default)** • -1: never starts • n positive integer: delay
`--stop-after-seconds=`	Setup sampling stop delay	• **-1 (default): stops with application end** • n positive integer: delay
`--pause-resume-at-SIGTSTP`	Each time LProf receives SIGTSTP (from the target application or CTRL+Z), sampling is paused (if running) or resumed (if paused).	(no value)
`--stdout-start-keywords=`	Each time LProf detects a keyword from stdout, sampling is started (no effet if already running)	Comma-separated list of words
`--stdout-stop-`	Each time LProf	Comma-separated list of words

`keywords=`	detects a keyword from stdout, sampling is stopped (no effet if already paused)	
`-btm=`	Select backtraces (callchains) collection method	• **`fp` (default on AArch64, recompile application with `-fno-omit-frame-pointer`)** • **`stack` (default on x86-64, higher overhead but no need to recompile application)** • `branch` (not really callchains but branch history, HW-dependent) • `off` (no callchains, lowest overhead)

Advanced/other Options (collect step)		
Name	**Short Description**	**Values**
`--lprof-path`	Enforces path to an alternative MAQAO for the collect step. Run `maqao lprof -h` for more details	Path to alternative MAQAO executable compiled for the compute node architecture
`--cpu-clock-MHz`	[perf-* engines] Override the "cpu-clock" perf-event rate (in MHz) measured by a calibration loop.	integer value
`--replace`	Overwrites an already existing output directory (reuse it). Remark: no effect on a not yet existing directory.	(no value)
`-tpp/--maximum-threads-per-process`	[perf-high-ppn only] Maximum number of concurrent threads per process. Default is OMP_NUM_THREADS. Used to set buffers and files size.	integer value
`-allow-extra-`	[perf-high-ppn only] Allows monitoring of threads above	(no value)

`threads`	maximum-threads-per-process.	
`--ppn/ maximum-processes-per-node`	Recommended if ppn > 1	Ex on single node: `lprof mpi-command="mpirun -n 32" ppn=32`
`--processes-whitelist`	Collect samples only for some processes (useful for large scale runs)	Comma-separated list of strings: • `cluster-master` (first observed process) • `node-masters` (first observer process in each node) • `PID<rank>` (process with a given MPI rank)
`--interposer-compiler`	[ADVANCED] Override the command (for instance `'clang'`) to compile the LD_PRELOAD interposer	Compilation command or "`none`"
`--poller-bind`	[ADVANCED, perf-low-ppn only] Set to "close" to bind poller threads to the same CPU as the ring-buffer. May improve or degrade performance	For the moment, only "`close`"
`--maximum-buffer-megabytes`	Allow to override Lprof memory footprint (default is 50 MB per CPU)	Maximum amount per node (Megabytes)
`--maximum-tmpfiles-megabytes`	Limit total temporary files size to X Megabytes per node. Default is 100 MB per CPU (HW thread).	Integer value
`-e/--evts`	Provide custom list of events to sample (CF `maqao --list-events`)	evt1_name@sample_period, … or evt1_code@sample_period, …
`--cnt-evts`	[EXPERIMENTAL] Provide a custom list of events (CF	Ex: `cnt-evts=RAPL_ENERGY_CORES,UNC_M_`

	`maqao --list-events`) to profile (counting). Use only dynamic PMU events (not counted by the CPU cores PMUs), which requires 0 or negative paranoid level (`sudo sysctl -w kernel.perf_event_paranoid=0`).	`CAS_COUNT_IMC0.RD`
`-p/--evts-profiles`	Use ready-to-use lists of events. Not yet supporting more than one profile.	string
`--cnt-evts-profiles`	[EXPERIMENTAL] Use ready-to-use lists of events (counting). Presently supported: ENERGY, DRAM_READS and DRAM_WRITES	string
`--cnt-metrics`	[EXPERIMENTAL] Counting metrics. Presently supported: - ENERGY_{PKG,DRAM} (add ENERGY into cnt-evts-profiles) - DRAM_{READS,WRITES} (add DRAM_{READS,WRITES} into cnt-evts-profiles)	string
`--max-callchain-length`	Maximum callchain length (default: 20), useful to reduce btm=stack overhead.	Positive integer
`--stack-size`	Size (in bytes) of stack to dump on samples (default: 8192). Using a smaller size (typically 4096) reduces profiling overhead but may cut (or loose) callchains. Using a bigger size (typically 16384) increases profiling overhead but should guarantee minimal callchains loss.	Positive integer

`--mmap-pages`	Overrides autotuned number of mmap pages for ring buffer payload.	Positive integer
`--collect-calls-info`	Collects source file/line information for callchain nodes (calls)	on (default)/off
`--engine`	Use another perf-events based sampling engine. With 4 or less processes, if you encounter troubles with the default (`perf-low-ppn`) engine, try to `perf-high-ppn` engine.	• `perf-low-ppn` (selected by default when perf-events are available with max 4 processes per node) • `perf-high-ppn` (selected by default when perf-events are available with more than 4 processes per node) • `no-perf` (selected by default when perf-events are not available)
`--include-sleep-time`	[no-perf only] Include sleep time (walltime).	(no value)
`--keep-external-threads`	[perf-high-ppn engine only] Profile threads with a different command line than the monitored application.	on/off (default)
`--keep-indirect-threads`	[perf-high-ppn engine only] Profile threads that are not direct children of the monitored application.	on (default)/off
`--ignore-signals`	[no-perf and perf-high-ppn engines] Prevents signals from being interpreted as termination	comma-separated list of integers

	signals. Allows to adapt no-perf and perf-high-ppn to various runtimes. Remark: for ignored signals also specified in set-exit-signals or set-abort-signals, evaluation order is set-abort-signals, set-exit-signals and then ignore-signals.	
`--set-exit-signals`	[no-perf and perf-high-ppn engines] Interpret signals as normal application exit. Allows to adapt no-perf and perf-high-ppn engines to various runtimes. Remark: for exit signals also specified in ignore-signals or set-abort-signals, evaluation order is set-abort-signals, set-exit-signals and then ignore-signals.	comma-separated list of integers
`--set-abort-signals`	[no-perf and perf-high-ppn engines] Interpret signals as abnormal application exit. Allows to adapt no-perf and perf-high-ppn engines to various runtimes. Remark: for abort signals also specified in ignore-signals or set-exit-signals, evaluation order is set-abort-signals, set-exit-signals and then ignore-signals	
`--legacy-maps`	[ADVANCED] Use only if unknown functions coverage is high for executable or libraries. Collect maps via legacy method (out of perf-events) after <legacy-maps> milliseconds and fallback to them in case of	Positive integer (number of milliseconds)

	unresolved addresses.	
`--collect-CPU-time-intervals`	[ADVANCED] [perf-low-ppn and perf-high-ppn engines] Collect per-thread CPU-time intervals. Allows to trace when and where (CPU) threads was running, and display them by adding -verbose at display step.	(no value)
`--disable-libs-links`	[ADVANCED] No more make unique (using links) files in <node>/libs directories.	(no value)

2.5 Collect step hints

In case of many application processes (typically MPI ranks), use:

- `collect-calls-info=off` to limit LProf memory footprint when dumping to disk source file/line for each call listed in callchains
- `processes-whitelist=node-masters` to limit LProf sampling to first observed rank within each node

3 Display

The two common display modes are text (default) and HTML.

3.1 Concepts

LProf relates *code regions* contributions to *system-levels*. User must then specify which code regions he is interested in and at which system level/granularity.

3.1.1 Code regions (hotspots)

From bigger to smaller:

- *Application:* set of *modules*
- *Module:* set of *functions*
- *Function:* set of *loops*
- *Loop:* set of *blocks*
- *Block:* basic block (compilation concept)

3.1.2 System levels

From bigger to smaller:

- *Cluster:* set of *nodes* (machines)
- *Node:* set of (system) *processes*
- *Process:* set of (system) *threads*
- *Thread*

3.2 Text Output

3.2.1 Functions Hotspots

To display summary view (at cluster level):

```
maqao lprof –df xp=<EXPERIMENT_DIRECTORY>
```

hotspots' level: functions

experiment directory (name or path) containing data to display

```
#########################################################################################################################################
#       Function Name       |       Module       |     Source Info     | Coverage (%) | Time Min(s) [TID] | Time Max(s) [TID] | Time w.r.t Walltime (s) #
#########################################################################################################################################
#  binvcrhs                 | bt-mz.A.4          | solve_subs.f:206    | 23.90        | 1.98 [12705]      | 2.58 [12693]      | 2.25                    #
#  z_solve_._omp_fn.0       | bt-mz.A.4          | z_solve.f:45        | 13.89        | 1.14 [12693]      | 1.48 [12692]      | 1.31                    #
#  matmul_sub               | bt-mz.A.4          | solve_subs.f:56     | 13.39        | 1.12 [12699]      | 1.48 [12693]      | 1.26                    #
#  y_solve_._omp_fn.0       | bt-mz.A.4          | y_solve.f:45        | 13.14        | 1.02 [12693]      | 1.56 [12698]      | 1.24                    #
#  x_solve_._omp_fn.0       | bt-mz.A.4          | x_solve.f:48        | 12.39        | 0.84 [12706]      | 1.42 [12705]      | 1.16                    #
#  compute_rhs_._omp_fn.0   | bt-mz.A.4          | rhs.f:33            | 12.12        | 0.98 [12698]      | 1.28 [12707]      | 1.14                    #
#  matvec_sub               | bt-mz.A.4          | solve_subs.f:27     | 3.67         | 0.26 [12699]      | 0.46 [12698]      | 0.34                    #
```

Figure 1 - LProf Output: Summary View (Functions)

To display view for a lower system level, use -dn (resp. dp, dt) for node (resp. process, thread). For instance, to display thread view:

```
maqao lprof –df xp=<EXPERIMENT_DIRECTORY> -dt
```

Thread ID

Hostname

Process ID

Walltime

```
                                    BACRI - PROCESS #12693
                                  Thread #12693 - 9.40 second(s)
#########################################################################################################################################
#       Function Name       |       Module       |     Source Info     | Coverage (%) | Time w.r.t Walltime (s) | CPI ratio #
#########################################################################################################################################
#  binvcrhs                 | bt-mz.A.4          | solve_subs.f:206    | 27.45        | 2.58                    | 0.68      #
#  matmul_sub               | bt-mz.A.4          | solve_subs.f:56     | 15.74        | 1.48                    | 0.69      #
#  compute_rhs_._omp_fn.0   | bt-mz.A.4          | rhs.f:33            | 12.77        | 1.20                    | 0.99      #
#  z_solve_._omp_fn.0       | bt-mz.A.4          | z_solve.f:45        | 12.13        | 1.14                    | 0.71      #
#  x_solve_._omp_fn.0       | bt-mz.A.4          | x_solve.f:48        | 11.91        | 1.12                    | 0.74      #
#  y_solve_._omp_fn.0       | bt-mz.A.4          | y_solve.f:45        | 10.85        | 1.02                    | 0.67      #
#  matvec_sub               | bt-mz.A.4          | solve_subs.f:27     | 2.98         | 0.28                    | 0.62      #
```

Figure 2 - LProf Output: Thread View (Functions)

3.2.2 Loops Hotspots

To display summary view (at cluster level):

```
maqao lprof -dl xp=<EXPERIMENT_DIRECTORY>
```

hotspots' level: loops

experiment directory (name or path) containing data to display

```
#################################################################################################
#  Loop ID  |    Module    |       Function Name       |      Source Info      |    Level    |
#################################################################################################
#  112      |  bt-mz.A.4   |  x_solve_._omp_fn.0       |  x_solve.f:146-309    |  Innermost  |
#  139      |  bt-mz.A.4   |  z_solve_._omp_fn.0       |  z_solve.f:146-309    |  Innermost  |
#  118      |  bt-mz.A.4   |  y_solve_._omp_fn.0       |  y_solve.f:145-308    |  Innermost  |
#  119      |  bt-mz.A.4   |  y_solve_._omp_fn.0       |  y_solve.f:55-137     |  Innermost  |
#  140      |  bt-mz.A.4   |  z_solve_._omp_fn.0       |  z_solve.f:55-137     |  Innermost  |
#  113      |  bt-mz.A.4   |  x_solve_._omp_fn.0       |  x_solve.f:57-139     |  Innermost  |
#  78       |  bt-mz.A.4   |  compute_rhs_._omp_fn.0   |  rhs.f:4-238          |  Innermost  |
```

Figure 3 – LProf Output: Summary View (Loops)

The above figure is truncated. In the actual output, four more columns are available on the right (same as functions mode):

Coverage (%), Time Min (s), Time Max (s) and Time w.r.t Walltime (s).

As for functions, use -dn/dp/dt to select a lower system level. For instance, to display thread view:

```
maqao lprof -dl xp=<EXPERIMENT_DIRECTORY> -dt
```

Thread ID

Hostname

Process ID

Walltime

```
                                BACRI - PROCESS #12693
                            Thread #12693 - 9.40 second(s)
#########################################################################################################################################
#  Loop ID  |    Module    |       Function Name       |      Source Info      |    Level    |  Coverage (%)  |  Time w.r.t Walltime (s)  |  CPI ratio  #
#########################################################################################################################################
#  139      |  bt-mz.A.4   |  z_solve_._omp_fn.0       |  z_solve.f:146-309    |  Innermost  |  9.57          |  0.90                     |  0.87       #
#  112      |  bt-mz.A.4   |  x_solve_._omp_fn.0       |  x_solve.f:146-309    |  Innermost  |  7.87          |  0.74                     |  0.72       #
#  118      |  bt-mz.A.4   |  y_solve_._omp_fn.0       |  y_solve.f:145-308    |  Innermost  |  6.38          |  0.60                     |  0.58       #
#  119      |  bt-mz.A.4   |  y_solve_._omp_fn.0       |  y_solve.f:55-137     |  Innermost  |  3.40          |  0.32                     |  1.17       #
#  78       |  bt-mz.A.4   |  compute_rhs_._omp_fn.0   |  rhs.f:4-238          |  Innermost  |  2.98          |  0.28                     |  1.07       #
#  113      |  bt-mz.A.4   |  x_solve_._omp_fn.0       |  x_solve.f:57-139     |  Innermost  |  2.34          |  0.22                     |  0.82       #
```

Figure 4 - LProf Output: Thread View (Loops)

3.3 Display Options

Basic Options (display step)		
Name	**Short Description**	**Values**
-df/-dl	Display functions/loops	(no value)
-db	Display basic blocks (for finer granularity than loops)	(no value)
-dn	Display per-node profiles (instead of cluster by default)	(no value)
-dp	Display per-process profiles (instead of cluster by default)	(no value)
-dt	Display per-thread profiles (instead of cluster by default)	(no value)
-lec/--libraries-extra-categories	Consider specified libraries as extra categories	libraries names as given by 'ldd <application>' or 'analyzed' to add one category per library processed by ldi/library-debug-info
-of/--output-format	Output results in a file of the given format (default if omitted: console output)	html or csv
-cc/--callchain	Specify objects for callchains analysis: • exe: display the callchain (if available) for each function with a scope limited to the application. • lib: extend the callchain scope to external libraries function calls. • all: display the callchain with no limited scope (application + libraries + system calls).	exe, lib, all or off

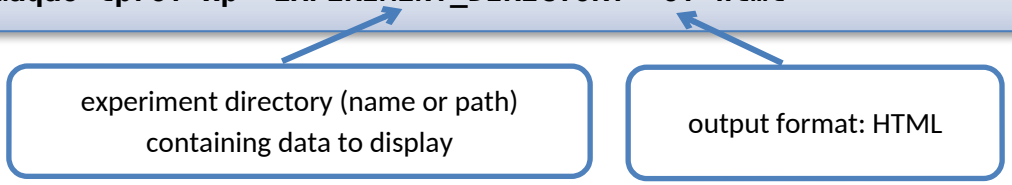
	• off: disable callchains analysis. Some OpenMP/MPI functions/loops will no more be correctly categorized. Use this only when display takes too much time/memory.	
-ct/--cumulative-threshold	Display the top loops/functions up to a given cumulated coverage (e.g: ct=50).	integer between 0 and 100

Basic Options (other options)		
Name	**Short Description**	**Values**
-display-walltime	While building node's results, display processes's duration. Combine with -verbose to display, for each thread, time elapsed after process's start and before process's end	(no value)
-display-CPU-time	While building node's results, display processes's user and system CPU time (getrusage)	(no value)
-display-raw-events	Use raw events instead of default metrics (coverage, walltime...). Possible to select only one or two items, for instance "samples,events".	comma-separated list of {percentage, samples, events}

3.4 HTML Output

3.4.1 Generation of HTML results

```
maqao lprof xp=<EXPERIMENT_DIRECTORY> of=html
```



This command generates an 'index.html' file into the *<EXPERIMENT_PATH>/html/* directory. Open this file into a web browser to see the results.

3.4.2 Interpretation of the Results

Refer to the Oneview tutorial:
https://maqao.org/documentation/MAQAO.Tutorial.ONEVIEW.pdf