LProf

Version 1.8
November 2022

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 (2000+ cores) with a very low overhead.
It is also possible to provide a custom list of hardware counters to sample.

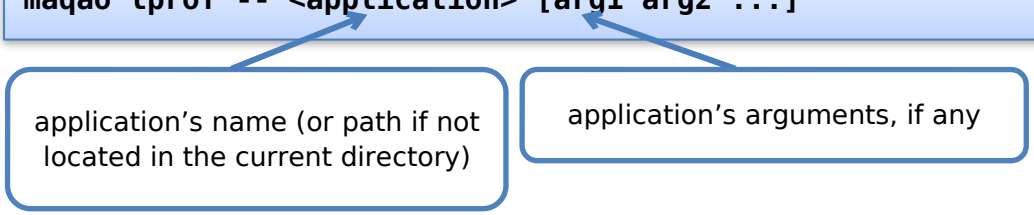
2) Data display

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 ...]
```

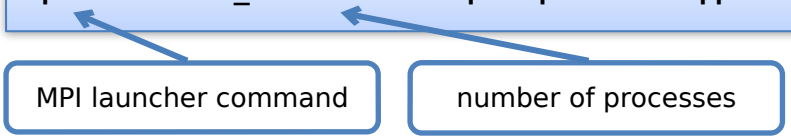


2.2 Parallel Run Command

WARNING! Invocation of LProf in MPI has changed in old versions of MAQAO.

MAQAO Version 2.4.4 and below: the command line begins with the MPI command.

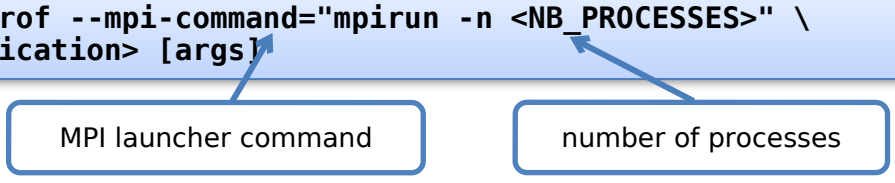
```
mpirun -n <NB_PROCESSES> maqao lprof -- <application> [arg1 arg2 ...]
```



From 2.4.5 version and higher: the *--mpi-command* option is required for interactive runs and *--batch-script*, for batch runs.

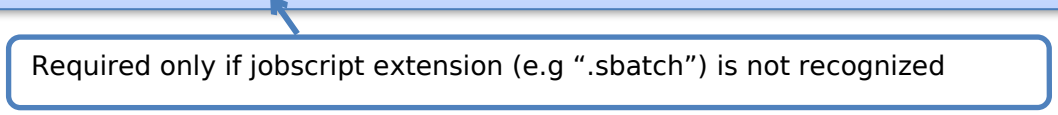
Interactive runs:

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



Batch runs:

```
maqao lprof --batch-script=<jobscript> [--mpi-command="mpirun..."] \
[--batch-command=<submission command>] -- <application> [args]
```



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
```

Since 2.12.0, you can (and must) inform Lprof about the maximum number of processes per node (if greater than 1), allowing it to set correct internal settings: `--maximum-processes-per-node`

Starting from 2.14.5, it is autodetected when missing but it is still recommended to set `--maximum-processes-per-node` if known and > 1.

2.3 Kernel samples exclusion

Since 2.12.0, kernel samples are not collected by default (recent Linux distributions do not allow this by default). To collect them:

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
```

If `sysctl kernel.perf_event_paranoid` returns 1 or less:

```
$ maqao lprof --include-kernel …
```

2.4 Options

To list all options along with their descriptions:

```
maqao lprof --help
```

Common Options (collect step)		
Name	**Short Description**	**Values**
`-xp=`	Specify the experiment directory	Directory's name (string)
`--mpi-command=`	Specify command for interactive MPI run or replacement value for <mpi_command> in job script	Ex: "`mpirun -n 4`"
`--ppn/maximum-processes-per-node=`	Since 2.12.0, mandatory when using `--mpi-command` Optional but recommended starting from 2.14.5 if ppn > 1	Ex on single node: `lprof mpi-command="mpirun -n 32" ppn=32`
`--batch-script=`	Jobscript to submit to job scheduler	Path to jobscript (string)
`--batch-command=`	Command used to submit jobs, required if jobscript extension is not recognized. Currently recognized: .sbatch and .pbs	Ex: "`sbatch`"
`--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)
`-ug=`	Control (*i.e.* pause/resume) measurement via a	on (CTRL+Z) | **`off` (default)** or a delay in seconds

	signal (Ctrl+Z) or via a countdown	
`-ldi=`	Scan debug information into all or specified (provided list) library(ies) to get loops details	on (all) | **`off` (default**) or list of libraries (**`'lib1, lib2, …'`**)
`-btm=`	Select backtraces (callchains) collection method	- **`fp` (default, recompile application with `-fno-omit-frame-pointer`)** - `stack` (higher overhead but no need to recompile application) - `branch` (not really callchains but branch history, HW-dependent) - `off` (no callchains, lowest overhead)

Advanced Options (collect step)		
Name	**Short Description**	**Values**
`--maximum-buffer-megabytes=`	Allow to override Lprof memory footprint (default is 50 MB per CPU)	Maximum amount per node (Megabytes)
`--engine=`	Use another perf-events based sampling 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)
`--evts=`	Provide custom list of events to sample (CF `maqao --list-events`)	evt1_name@sample_period, … or evt1_code@sample_period, …

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/cluster 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

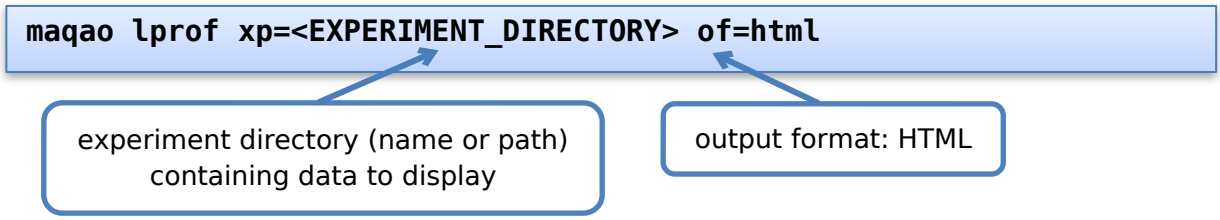
BACRI - PROCESS #12693
Thread #12693 - 9.40 second(s)

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 HTML Output

3.3.1 Generation of HTML results



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

3.3.2 Interpretation of the Results

Refer to the Oneview tutorial: http://maqao.org/release/MAQAO.Tutorial.ONEVIEW.pdf