ONE-View

How to generate and use reports with ONE-View

Version 1.5

January 2019

MAQAO Tutorial series

www.maqao.org

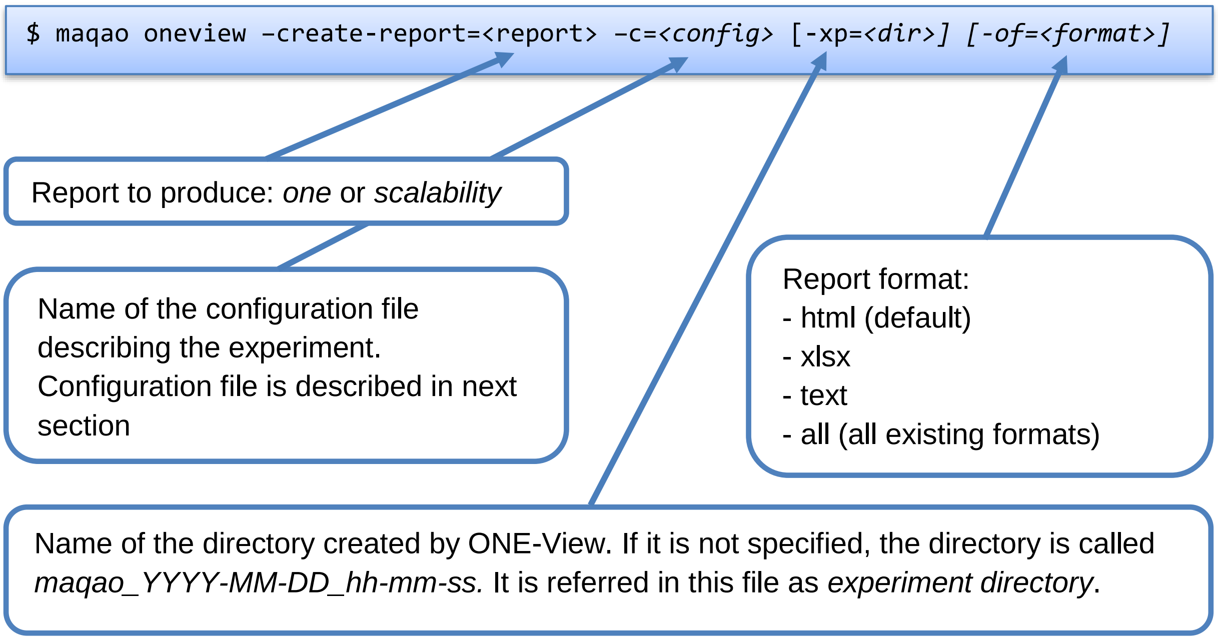
1 Introduction

ONE-View is the MAQAO module in charge of driving all other MAQAO modules in order to produce reports aggregating results from all these tools. It automatizes the execution of other modules (such as LProf or CQA) to generate reports in HTML pages or XLSX data sheets.

ONE-View offers several built-in reports combining both static and dynamic approaches to get an overview of the application performance. This document details reports ONE and SCALABILITY, which use MAQAO modules LPROF (a dynamic profiler) and CQA (a static code analyser).

2 Running ONE-View

To generate a report using ONE-View, the default command is:



```
$ maqao oneview –create-report=<report> –c=<config> [-xp=<dir>] [-of=<format>]
```

Report to produce: *one* or *scalability*

Name of the configuration file describing the experiment.
Configuration file is described in next section

Report format:
- html (default)
- xlsx
- text
- all (all existing formats)

Name of the directory created by ONE-View. If it is not specified, the directory is called *maqao_YYYY-MM-DD_hh-mm-ss.* It is referred in this file as *experiment directory*.

It is also possible to provide the necessary parameters to ONE-View from the command line. It can be used when a small set of variables in the configuration file are needed. All existing parameters for the configuration of the experiment are available in sections 3.2 and 3.3.

To list all options for ONE-View:

```
$ maqao oneview --help
```

The report *ONE* is the simpler and faster report. It combines a profiling of the application using LPROF module with CQA static analysis on loops and functions. The report *scalability* contains all data from the report *ONE* and additional data generated using several profiling of the application with different values for the number of processes and the number of threads.

3 Filling the Configuration File

3.1 Creating the configuration file

To generate a template of configuration file:

```
$ maqao oneview -create-config[=<file>]
```

Name of the generated configuration file. If it is not specified, the configuration file is called *config.lua* and it is created in the current directory.

The template contains all available fields and it is fully documented. This document details all fields in next subsections. The configuration file uses the Lua syntax; in particular, commented lines are prefixed with "--" (two dashes).

3.2 Main fields

- *binary*: Path to the binary to analyse
- *dataset*: Path to a directory containing the application dataset. If filled, this directory is copied into the experiment directory so it must be as small as possible. The experiment directory must not be created in the dataset directory.
- *run_command*: A string detailing how the binary must be run. In this string, the binary is referred as *<binary>.* This substring is automatically replaced by the correct binary name when ONE-View needs to run any version of the binary.
  If the application does not need any parameter, the field has "*<binary>*" as value. If the application needs two parameters, *-a=5* and *--b*, the field has "*<binary> -a=5 --b*" as value.
- *run_directory*: A string detailing where the binary should be run. Some applications must be run from a specific directory, most of the time a directory related to the dataset directory. This field is used to specify this path. The substring *"<dataset>*" can be used to represent the path to the dataset directory located in the experiment directory and it is automatically substituted by the real path by ONE-View during runs.
- *mpi_command*: A string detailing the MPI command to use to run the application. If MPI should not be used, this string must be empty. If MPI is used, this field must contain the call to *mpirun* or *mpiexec* with all its parameters, except the application and its own parameters.

For example, if a binary needs the following command to be run:

```
$ mpirun -n 4 ./my_app 250 -output=./log.out
```

The corresponding configuration file contains:

```
binary = "./my_app"
run_command = "<binary> 250 -output=./log.out"
mpi_command = "mpirun -n 4"
```

- *omp_num_threads:* A number specifying how many OpenMP threads must be used for the experiment. If the number is greater than one, the environment variable OMP_NUM_THREADS is defined before executing the application.
- *batch_command:* When the cluster uses a batch manager, this variable describes how to use it. If a script is needed, it must be referred as *<batch_script>.*
- *batch_script:* Path to a script used by a batch manager. The script must be adapted to ONE-View by using the code *<run_command>* instead of the classic binary execution command. For example, a batch script adapted for ONE-View for SLURM can be:

```
#! /bin/bash

#SBATCH SETTINGS
#SBATCH –J myJob
…

#APPLICATION SETTINGS
module load …
export MY_VAR …

#RUN THE APPLICATION
# mpiexec –n 16 ./my_app
<run_command>
```

- *scalability_params*: When scalability report is generated, describes all experiments to run. It is a table containing one entry per experiment, with following fields:
  - *nb_processes* – Number of processes for the experiment. Default is 1. It substitutes *<number_processes>* in the *batch_script* or the *mpi_command* fields for scalability runs.
  - *nb_threads* – Number of threads for the experiment. Default is 1. It substitutes *omp_num_threads* for scalability runs.
  - *binary* – Path to a binary if a specific version must be used with the configuration. Default value is the path defined in the *binary* field.

3.3 Secondary fields

- *external_libraries*: An optional table describing dynamic libraries to analyse in addition of the binary. Most of the time, linked libraries are system libraries that cannot be easily optimized so it is not interesting to analyse them but sometimes, the application to optimize is not an executable but a library. Each entry in the table is a string with the name of a library to analyse.
- *source_code_location*: An optional string detailing where the source code is located. It is needed to localize the source code of your application if it is not at the location defined in debug data (which is set when compiling the application).
- *maximal_path_number*: A number indicating the maximal number of paths in the control flow graph a loop can have. Loops with a greater number of paths will not be analysed.

- *excluded_areas*: A table describing areas (loops or basic blocks) which must be skipped from analysis. The table is a set of sub tables containing three fields: *type* which can be either *loop* or *block*; *id* which is the identifier of the area to exclude; *module* which is either *binary* (the default value) when the area is in the main binary or an external library name (as defined in the field external_libraries).
- *number_nodes:* Number of nodes to use to run the application on the cluster. Can be referred as <*number_nodes*> in the *batch_script* or the *mpi_command* fields.
- *number_processes:* Number of processes to use to run the application. Can be referred as <*number_processes*> in the *batch_script* or the *mpi_command* fields.
- *number_tasks_nodes:* Number of tasks per nodes to use to run the application. Can be referred as <*number_tasks_nodes*> in the *batch_script* or the *mpi_command* field.

The following fields are not used by report ONE and reserved for future releases:

- *filter*
- *freqencies*
- *profile_start*
- *additional_hwc*
- *decan_multi_variant*
- *is_sudo_available*

4 Reading Reports

Reports are generated in *<experiment_directory>/REPORTS/* as *<binary>_<report>.<format>*, where *<binary>* is the analysed binary, *<report>* is the value of the parameter *–create-report* and *<format>* the value of the parameter *format*.

4.1 HTML Output

HTLM reports can be read using Mozilla Firefox, Google Chrome and Microsoft Edge web browsers. The main file is **index.html**, located in *<experiment_directory>/RESULTS/<report>_html/*. All tabs have a menu located at the top of the tab which can be used to navigate between tabs. All tabs are described in next subsections. On most on tabs, there are one or several symbols ? that display help when the cursor is over them.

4.1.1 Global

The file **index.html** is the report index and it presents four sections:

- **Experiment Summary**, on the top left, describes some parameters about the run
- **Configuration Summary**, on the top right, describes parameters from the configuration file
- **Global Metrics**, on the bottom left, presents some metrics summarizing the application performances. Metrics are highlighted from green to red to signal best to worse performance. Each metric is described by a tooltip appearing when the cursor is above the metric name. Metrics *Clean*, *FP Vectorised* and *Fully Vectorised* can be clicked to change the chart according to the selected metric. These metrics detailed some potential speedups obtainable under specific conditions, according to the number of loops modified to satisfy the condition. Charts present speedups obtainable if the assembly code is modified to satisfy some conditions:
    - **If Code Clean** means that all instructions which do not perform floating-point computation or memory accesses are deleted. Instructions used to handle the loop control flow are not included in the instructions set to remove;
    - **If FP Vectorized** means that all instructions performing floating-point computation are vectorized;
    - **If Fully Vectorized** means that all instructions performing floating-point computation and all memory accesses are vectorized.

These charts allow to know if the application can be optimized and what speedup can be achieved

- **Speedup charts**, on the bottom right, details which loops give better speedups if they are optimized. When the symbol ⏎ appears, it can be click to display the summary speedup chart.

In the *scalability* report, an additional section is available at the page bottom, detailing the global efficiency of the application and the speedup for each run of the scalability analysis. Both metrics are computed using the first experiment describes in *scalability_params* as reference. An array with all chart values can be displayed by expanding the **Detailed Speed-Up and Efficiency** section.

4.1.2 Application

The **Application** tab shows several charts available by clicking on the corresponding menu entry on the left. Menu entries whose name started by *Scalability* are only available in *scalability* report. Other entries are available with any report.

4.1.2.1 Application Categorization

It details the percentage of the application spent in various categories. The section **Detailed Application Categorization** can be expanded to reveal a table with all data for each thread, process and node. An example is available with Figure 1 - Application Categorization. In this example, there is about 20% of the application time spent in MPI runtime (not MPI parallel sections), 70% in the application code (it include parallel regions) and 10% in two other categories.

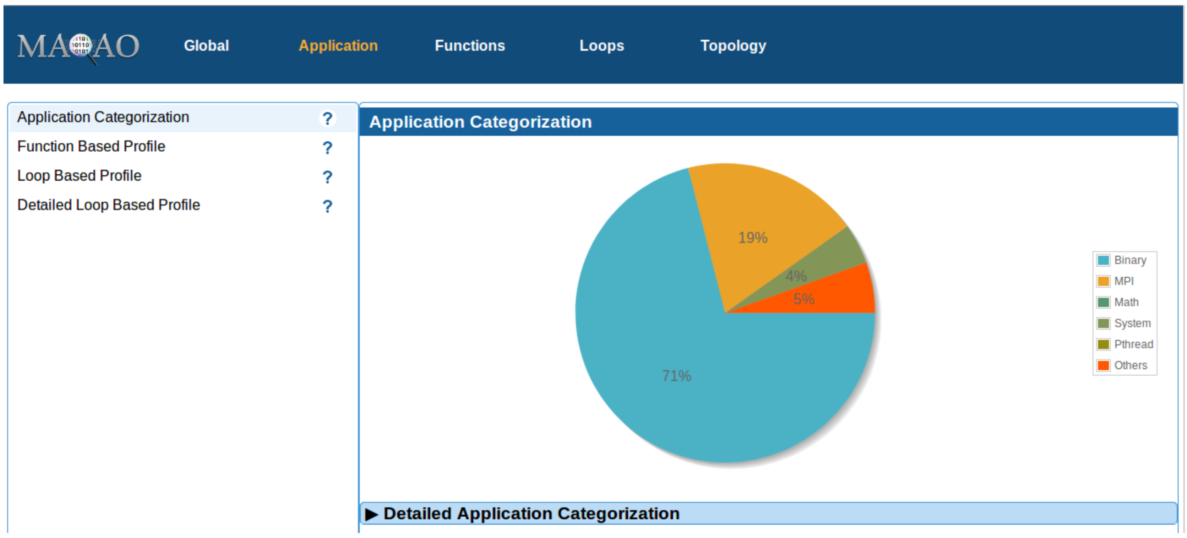


Figure 1 - Application Categorization

4.1.2.2 Function Based Profile

It presents a profiling of the application at the function level. Functions are grouped by coverage in buckets and for each bucket, three metrics are available:

- The number of function in the bucket,
- The total coverage of the bucket,
- The cumulated coverage with all previous buckets

An example is available with Figure 2 - Function Based Profile. The example presents an application containing five functions whose coverage is greater than 8%, which represents 54% of the total time.

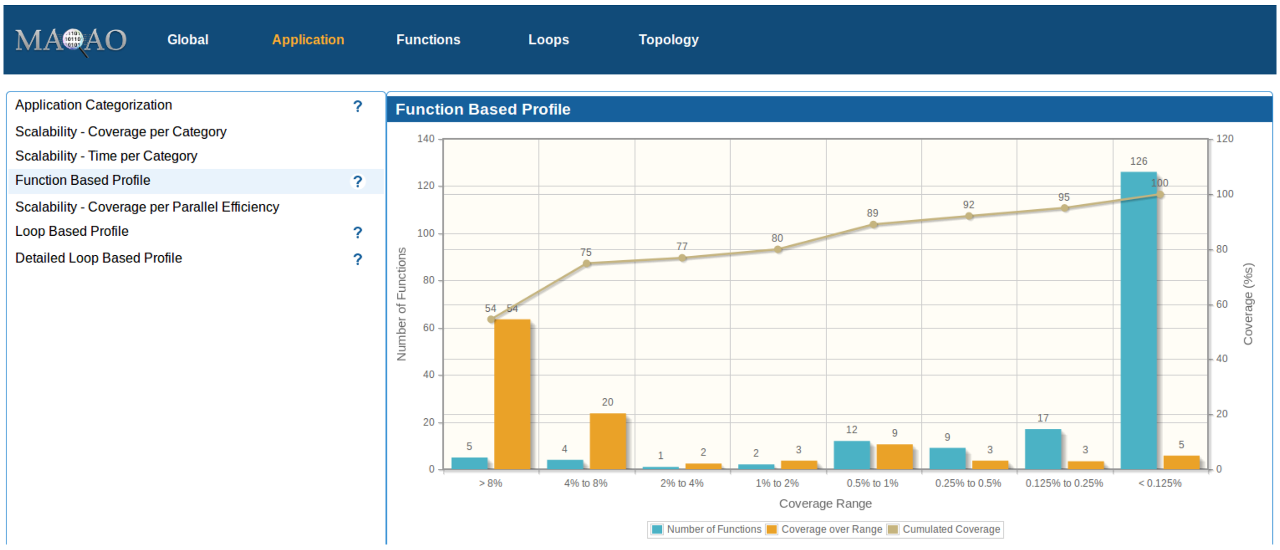


Figure 2 - Function Based Profile

4.1.2.3 Loop Based Profile

It presents a chart similar than the Function Based Profile described in section 4.1.2.2, but at the loop level, and using only innermost loops. An innermost loop is a loop which does not contain any loop.

4.1.2.4 Detailed Loop Based Profile

It presents a breakdown of loops types (innermost, outermost and inbetween) of the chart presented in section 4.1.2.3. An outermost loop contains at least another loop and an in between loop contains at least one loop and is contained in another loop. Figure 3 - Detailed Loop Based Profile shows an example of this chart.

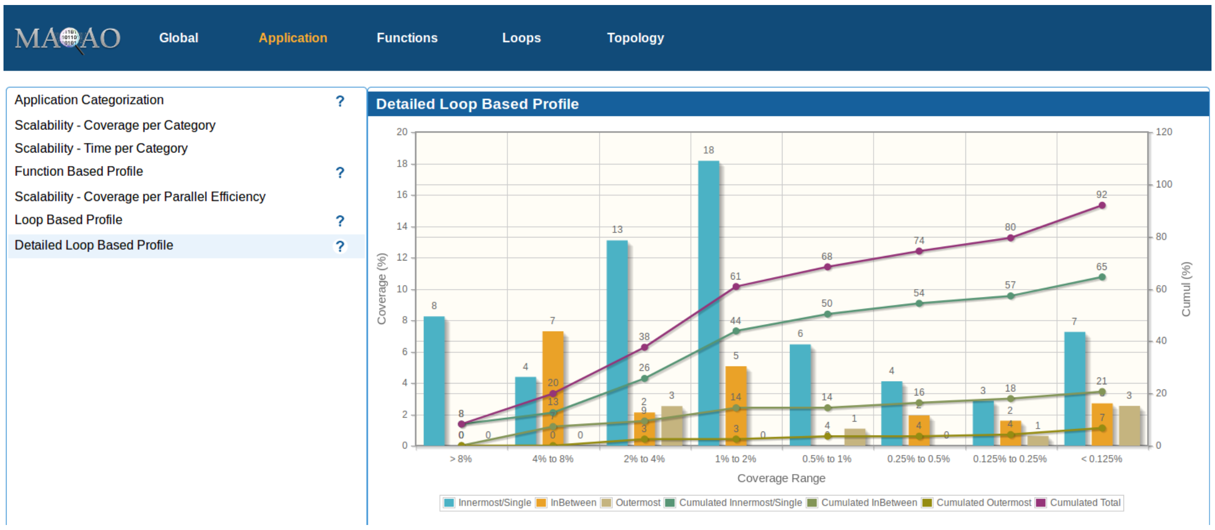


Figure 3 - Detailed Loop Based Profile

4.1.2.5 Scalability - Coverage Per Category

It presents the same data than section 4.1.2.1, but there is one bar per configuration in the scalability parameters. Configurations are formatted as *<nb_processes>-<nb_threads>* It allows to see the impact of the number of processes and threads on the different categories. An example is shown by Figure 4 - Scalability: Coverage Per Category. In the example, we can see the time spent in MPI library increase with the number of processes.

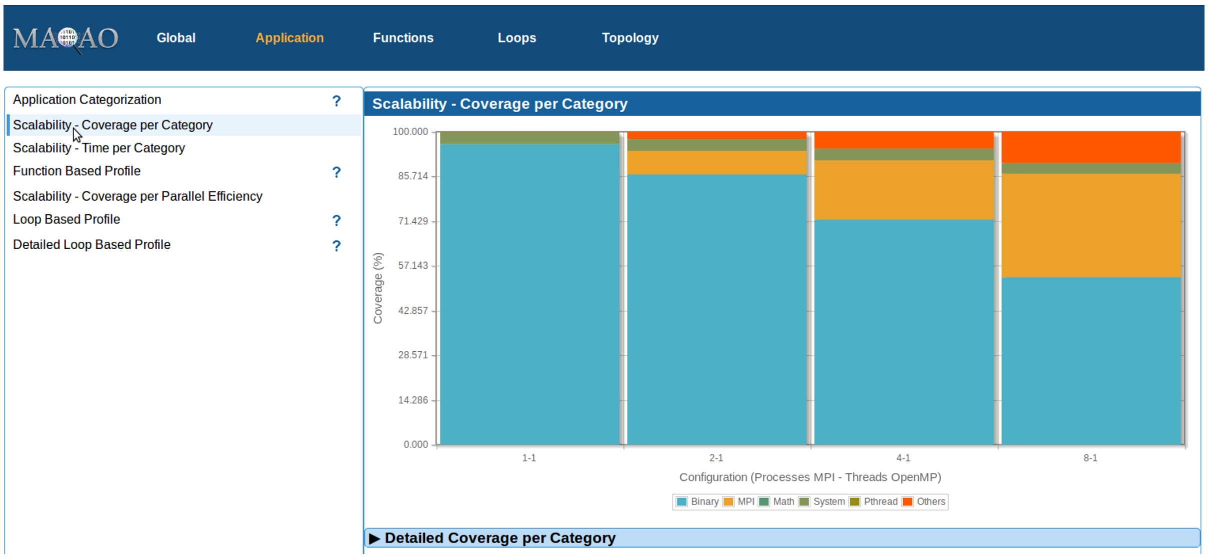

MAQAO
Global
Application
Functions
Loops
Topology
Application Categorization
Scalability - Coverage per Category
Scalability - Time per Category
Function Based Profile
Scalability - Coverage per Parallel Efficiency
Loop Based Profile
Detailed Loop Based Profile
Scalability - Coverage per Category
Coverage (%)
100.000
85.714
71.429
57.143
42.857
28.571
14.286
0.000
1-1
2-1
4-1
8-1
Configuration (Processes MPI - Threads OpenMP)
Binary MPI Math System Pthread Others
▶ Detailed Coverage per Category

Figure 4 - Scalability: Coverage Per Category

4.1.2.6 Scalability - Time Per Category

It presents data similar than section 4.1.2.5, but now it is a time (in seconds) and not a coverage that is displayed. It allows to see the impact of the parallelism over the application time and categories. An example is presented by Figure 5 - Scalability: Time Per Category.

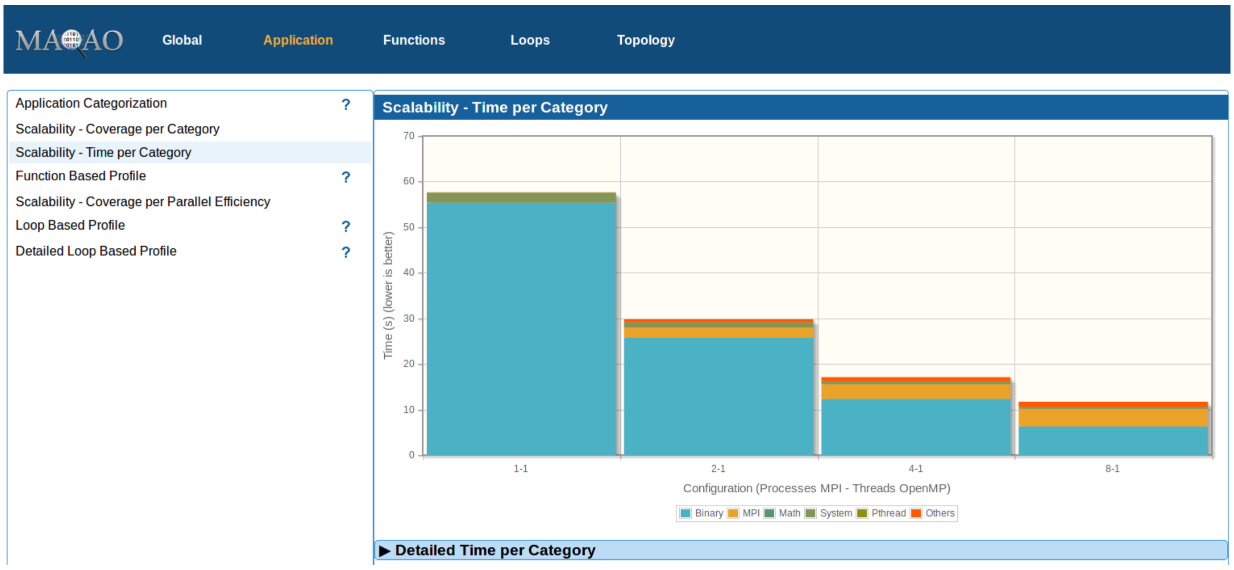

MAQAO
Global
Application
Functions
Loops
Topology
Application Categorization
Scalability - Coverage per Category
Scalability - Time per Category
Function Based Profile
Scalability - Coverage per Parallel Efficiency
Loop Based Profile
Detailed Loop Based Profile
Scalability - Time per Category
Time (s) (lower is better)
70
60
50
40
30
20
10
0
1-1
2-1
4-1
8-1
Configuration (Processes MPI - Threads OpenMP)
Binary MPI Math System Pthread Others
▶ Detailed Time per Category

Figure 5 - Scalability: Time Per Category

4.1.2.7 Scalability - Coverage Per Parallel Efficiency

The chart presents the efficiency of functions across runs of the scalability analysis. The efficiency is based on the first run described in parameters so the first bar is always in the grey color. Grey elements are functions that where not found during the first profiling. Colours varies from green for efficient functions to red for not efficient functions. An example is displayed by Figure 6 - Scalability: Coverage per Parallel Efficiency.

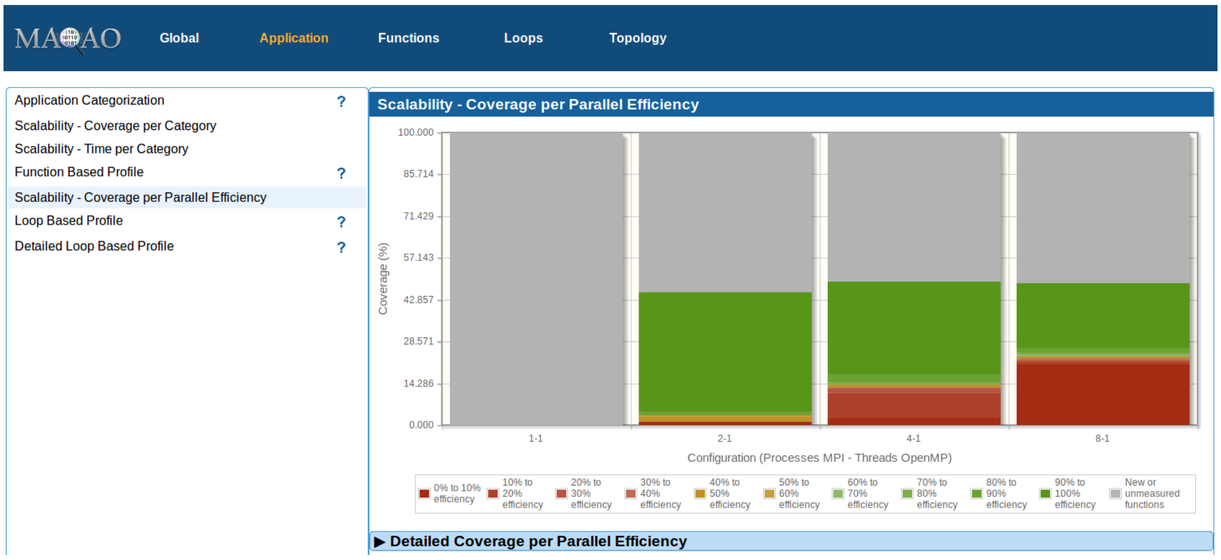


Figure 6 - Scalability: Coverage per Parallel Efficiency

4.1.3 Functions

The **Functions** tab presents a profiling of the application at the function level, listing all used functions with their coverage. By clicking on the arrow on the left of any functions, the box can be opened to reveal all profiled loops belonging to the function represented as a tree. Loops can also be opened by clicking on the left arrow. If a loop has a circle instead of an arrow, it means it is an innermost loop. All coverages are global to the application. A row can be fully expanded by clicking on the symbol + appearing on the right of the current row.

By clicking on a column header, the table is sorted according to this column.

By right-clicking on a row (either loop or function), a menu appears and allows to display several charts:

- Load Distribution: The distribution of the function / loop coverage across threads.
- Sorted Load Distribution: The distribution of the function / loop coverage descending sorted across threads.
- Load Distribution All Threads: The distribution of the function / loop coverage across threads, including 0 values for threads that do not run the function / loop.
- Scalability Report: Only available in the scalability analysis, it presents the efficiency and the speedup of the functions / loop across all experiments of the scalability analysis.

By double-clicking on a function or a loop, a new tab presenting all results for the loop is opened in the browser. Details about loop tabs are described in the subsection **Loop** and details about function tabs are described in the subsection **Function**.

In the scalability report, optional columns can be displayed by clicking on boxes in the list above the table to display efficiency and speedups from the scalability analysis.

4.1.4 Function

The **Function** is not accessible from the menu, but only from tabs **Functions** and **Loops**. This tab is split in two panels whose width can be adjusted by moving the vertical blue bar on the left or on the right. Each panel content can be changed by selecting a report in the select box. Current reports are:

- The source code is available.
- The load distributions charts.
- The CQA report. More details about CQA are available in the CQA tutorial available at http://maqao.org/release/MAQAO.Tutorial.CQA.intel64.pdf. Current path can be changed using arrows in the path selection header or by selecting a path identifier in the text box then clicking on the *OK* button.
- The function loop hierarchy with links to all its loops report.
- In the scalability report, the function scalability report.

The symbol ↗ can be clicked to open the current panel in a new browser tab. The same report cannot be opened in both panels.

4.1.5 Loops

The **Loops** tab presents a profiling of the application at the loop level, listing all analysed loops. For each loop, there is the MAQAO identifier, data about the location in the source code and the coverage with a colour associated to it. The colour is red when the loop is hot (high coverage) and it goes to blue when the loop is cold (low coverage).

Additional columns can be displayed by checking to corresponding box just above the table.

By clicking on a column header, the table is sorted according to this column.

By right-clicking on a row, a menu appears and allows to display several charts:

- Load Distribution: The distribution of the loop coverage across threads.
- Sorted Load Distribution: The distribution of the loop coverage descending sorted across threads.
- Load Distribution All Threads: The distribution of the loop coverage across threads, including 0 values for threads that do not run the loop.
- Scalability Report: Only available in the scalability analysis, it presents the efficiency and the speedup of the loop across all experiments of the scalability analysis.

By double clicking on a loop, a new tab presenting all results for the loop is opened in the browser. Details about this tab are described in the subsection **Loop**.

4.1.6 Loop

The tab **Loop** is not accessible from the menu, but only from tabs **Functions** and **Loops**. This tab contains all available data about a specific loop and is similar than **Function** tab described in section 4.1.4. Its reports are:

- The source code is available,
- The assembly code with a memory group analysis that can be displayed by clicking on the corresponding button. A memory group is a set of assembly instructions that access to a same memory region. Most of the time, it corresponds to a same source data structure.
- The load distributions charts
- The CQA report. More details about CQA are available in the CQA tutorial available at http://maqao.org/release/MAQAO.Tutorial.CQA.intel64.pdf. Current path can be changed using arrows in the path selection header or by selecting a path identifier in the text box then clicking on the *OK* button.
- A table with more advanced CQA metrics
- In the scalability report, the function scalability report.

The symbol can be clicked to open the current panel in a new browser tab. The same report cannot be opened in both panels.

4.1.7 Topology

The tab **Topology** presents the topology of the run, meaning how threads, processes and nodes used during the run are organised. The table can be expanded by clicking on the left arrow, or fully expanded by clicking on the + symbol appearing on the right of the current row.

By double-clicking on a thread row, a new tab with the thread profiling at the function level is opened.

In the scalability report, additional tables are available for each experiment.

4.2 Text Output

The text report is displayed on the terminal. It can be customized using several options:

- *--text-global [=on/off]*: Display *Global* section if parameter is *on* (default), else do not display it if *off.*
- *--text-application [=on/off]*: Display *Application* section if parameter is *on* (default), else do not display it if *off.*
- *--text-functions [=on/off]*: Display *Functions* section if parameter is *on* (default), else do not display it if *off.*
- *--text-functions-full [=on/off]*: Display all data for *Function* section if parameter is *on* (default), else do not display it if *off.*
- *--text-loops [=on/off]*: Display *Loops* section if parameter is *on* (default), else do not display it if *off.*
- *--text-loops-full [=on/off]*: Display all data for *Loops* section if parameter is *on* (default), else do not display it if *off.*
- *--text-cqa [=on/off/[module:]id1, [module:]id2]*: Display *CQA* section if parameter is *on* (default), else do not display it if *off.* Analysed loops can be filtered by giving for each loop its module (*binary* (default) or an entry in *external_libraries*) and its MAQAO identifier.
- *--text-cqa-full [=on/off/[module:]id1, [module:]id2]* Display all data for *CQA* section if parameter is *on* (default), else do not display it if *off.* Analysed loops can be filtered by giving

for each loop its module (*binary* (default) or an entry in *external_libraries*) and its MAQAO identifier.

Default output display sections *Global*, *Application*, *Functions*, *Loops* and *CQA*.

Text report sections are similar to corresponding HTML sections. CQA section is CQA reports of selected loops.

There is no special output for scalability in text output, it will be added in a future update.

4.3 XLSX Output

XLSX files can be read by several softwares: Microsoft Office Excel, LibreOffice, OpenOffice. The file contains several tabs whose content is presented in HTML section (section 4.1).

4.3.1 Experiment_Summary

This tab presents blocs **Experiment Summary**, **Global Metrics** and **Configuration Summary** described in the section 4.1.1

4.3.2 Application_Categorization

This tab presents the chart described in section 4.1.2.1.

4.3.3 Functions_Profile

This tab presents the chart described in section 4.1.2.2.

4.3.4 Function_Listing

This tab presents the table described in section 4.1.3.

4.3.5 Innermost_Loops_Profile

This tab presents the chart described in section 4.1.2.3.

4.3.6 Loops_Profile

This tab presents the chart described in section 4.1.2.4.

4.3.7 Static_Optimization_Report

This tab presents a table containing the CQA advanced metrics described in section 4.1.6 for each path of each analysed loop.

4.3.8 Potential_Speedups

This tab presents *speedup if* charts described in the section 4.1.1 with data tables.