Analysing with ONE view

```
$ maqao oneview -R1 [options] -- cmd
```

cmd corresponds to the name of the executable to analyse and its parameters.

ONE View general options:

- -xp=*exp dir*: If specified, the results will be stored in directory exp dir. If omitted, directory maqao_<timestamp> will be created in the current directory.
- --output-format=html(default)|xlsx|text|all: Output format.
- --with-scalability=[strong(default)|weak]: Toggles scalability mode. The multiruns_params array must be filled in the configuration file.

Using a configuration file for ONE View:

- --config=*cfg path*: Uses file cfg path to retrieve options. Options in cfg path are similar to the execution options described below (without '--' and replacing '-' with '_') and declared as Json variables ("option": "value" or "option": number). Other variables can be referenced by enclosing them in brackets (<>). For instance: mpi_command="mpirun -n <number_processes>"
- --create-config=*sample cfg*: Generates empty configuration file. If sample cfg is omitted, "**config.json**" will be created in the current directory.
- --create-config-template=*template*: Generates a sample configuration file. If template is omitted, templates for standard analysis cases (sequential, parallel, scalability, …) will be created in the current directory.

Parallel execution options:

- --mpi-command=*mpi cmd*: MPI runtime invocation. Will prepend cmd.
- --number-processes-per-node=*num*: Number of MPI tasks (or processes) per node (recommended if known and num > 1)

Batch scheduling execution options:

- --batch-script=*script path*: Path to job scheduler script. The script must have been modified to replace the application executable and its arguments with keyword "<run_command>".
- --batch-command=*batch_cmd*: Command for invoking the job scheduler, using keyword <batch_script> to reference script path.

Viewing reports:

- Text reports are displayed directly on the console output.
- HTML: open **exp dir/RESULTS/<executable_name>_one_html/index.html** in a browser to display the HTML reports.
- XLSX reports are in file **exp dir/RESULTS/<executable_name>_one_0_0.xlsx**
- The path to the reports is displayed at the end of ONE View analysis.

Sample invocations of ONE View

- Command line on interactive MPI run
  - Original line:
```
$ OMP_NUM_THREADS=2 mpirun -n 4 ./my_exe my_args
```
  - Corresponding ONE View invocation:
```
$ OMP_NUM_THREADS=2 maqao oneview -R1 --number-processes=4\
--mpi-command="mpirun -n <number_processes>" \
--number-processes-per-node=2 -- ./my_exe my_args
```
- Command line for job scheduler script (script must be edited to replace original command line with keyword "<run_command>")
```
$ maqao oneview -R1 --batch-script="script.job" \
  --batch-command="my_jobsched <batch_script>"
```
- Using ONE View configuration file
```
$ maqao oneview --create-config=my_config.json
{edit my_config.json to fill all required variables}
$ maqao oneview -R1 --config=my_config.json
```
- Compare existing ONE View reports
```
$ maqao oneview --compare-reports --inputs=exp_dir1,exp_dir2,…
```

Advanced: Invoking LProf / CQA separately

Profiling with MAQAO LProf

If ***exp_dir*** is omitted, a directory named ***maqao_lprof_<timestamp>*** will be created.

- Sequential / OpenMP profiling
```
$ maqao lprof [-xp=exp_dir] -- ./foo arg1 arg2 …
```
- MPI / hybrid profiling
```
$ maqao lprof [-xp=exp_dir] --mpi-command="mpirun -n 32 -ppn 4" \
  ppn=4 -- ./foo arg
```
- Displaying profiling results
```
$ maqao lprof -xp=exp_dir -df # Functions profiling results
$ maqao lprof -xp=exp_dir -dl # Loops profiling results
```

Analysis with CQA

id1, *id2*, *id3* … are the numerical loop identifiers returned by **LProf**.

- Analysing a given loop or set of loops
```
$ maqao cqa ./my_app -loop=id1,id2,id3…
```
- Analysing all innermost loops in a given function or set of functions
```
$ maqao cqa ./my_app -fct-loops="regexp"
```
- Analysing the body of a given function or set of functions
```
$ maqao cqa ./my_app -fct-body="regexp"
```

regexp is a regular expression: *foo* matches "foo1", "foo" or "afoo", while *^bar$* matches "bar" only