Using MAQAO to Analyse and Optimise an Application

Cédric Valensi, William Jalby, Mathieu Tribalat, Emmanuel Oseret, Salah Ibnamar, Kevin Camus
Exascale Computing Research
University of Versailles Saint Quentin en Yvelines
Versailles, France

***Abstract*—One of the major issues in the performance analysis of HPC codes is the difficulty to fully and accurately characterize the behaviour of an application. In particular, it is essential to precisely pinpoint bottlenecks and their true causes. Additionally, providing an estimation of the possible gain obtained after fixing a particular bottleneck would surely allow for a more thorough choice of the optimizations to apply or avoid. In this paper, we present MAQAO, a performance analysis framework aiming at providing a comprehensive human-friendly view of performance issues and also guide the user's optimization efforts on the most promising performance bottlenecks.**

I. Introduction

The evolution of the recent HPC processors has shown an increase both in terms of number of components (larger multi-core/many-cores) and in terms of mechanism complexity (advanced out of order, multilevel memory hierarchies). These trends make the task of application optimization more and more complex: not only the sources of potential performance loss have become more diverse, but several of them can occur simultaneously and with different impacts. Additionally, sorting out the sources from the consequences of performance losses, therefore identifying the correct issues to be addressed, becomes increasingly difficult.

When looking for potential improvements, code developers first need to identify the critical performance issues to be tackled, and second, to know how much performance gain can be obtained after applying a specific optimization. Developers will mostly be interested in optimizing the code for different data sets and for different configurations (number of cores, nodes, ...), therefore needing to aggregate performance views across different cases to study the performance impact and select the correct trade off. Unfortunately, this simple aggregation capacity is missing in most performance tools. Also, these tools usually provide results as a collection of metrics which require expertise in optimisation to correctly interpret.

In this paper, we present the MAQAO performance analysis framework and its performance view aggregator ONE View, which aims at precisely helping code developers in selecting, with some reasonable confidence, the most profitable optimizations. ONE View allows to automatize the analysis process and aggregate the result from different and complementary analysis modules into a set of reports providing a full assessment of the application performance behaviour and evaluations of the potential performance gains of code optimisations.

II. MAQAO Performance Analysis Framework

MAQAO (Modular Assembly Quality Analyzer and Optimizer) [1] is a performance analysis and optimisation framework operating at binary level, with a focus on core performance. Its main goal is to guide application developers along the optimization process through synthetic reports and hints. The framework mixes both dynamic and static analyses based on its ability to reconstruct high level structures such as functions and loops from an application binary. Since MAQAO operates at binary level, it is agnostic with regard to the language used in the source code and does not require recompiling the application to perform analyses. MAQAO has been successfully used to optimise academic and industrial codes.

The MAQAO modules rely on the internal representation of the binary to perform distinct and complementary analyses, which include the following:

- Profiling through two modules, LProf, a sampling-based lightweight profiler offering results at both function and loop levels, and VProf, an instrumentation-based value profiler allowing to retrieve dynamic metrics for each instance of a loop.
- Static analysis through CQA [3], a module assessing the quality of the code generated by the compiler and producing a set of reports describing potential issues, estimations of the gain if fixed, and hints on how to achieve it.
- Differential analysis through DECAN [1], a module that modifies the application binary code to evaluate the impact of families of instructions on global performance.

III. ONE View

ONE View is a MAQAO module in charge of launching all of the other performance modules according to a coherent workflow [2], formatting their output, and building the various performance views in a report available in the HTML, XLSX or text format. ONE View offers three levels of reporting, with increasing levels of complexity and associated overhead, each report level including the results of the levels below it. Another mode of operation allows ONE View to execute the application with different parallel configurations in order to estimate its scalability. The reports generated by ONE View are structured along a set of complementary views.

[1] www.maqao.org

A. Global Metrics

This view presents an estimation of the overall quality of the program with regard to performance and the possible improvements to be expected. It includes a set of global metrics and graphs presenting the speed-ups expected if performing optimisations suggested by CQA and DECAN analyses.

The global metrics aim at giving an overall view of the quality of the code. They include timings of the application and the percentage of it spent in loops, a list of standard optimisation options that were missed when compiling the application, an estimation of the complexity of the flow (average number of paths in loops), an estimation of the regularity of accesses to array elements across the whole application, and a series of speed-up predictions if a given optimisation could be applied to every loop in the file, along with the number of loops to optimise to obtain 80% of this speed-up.

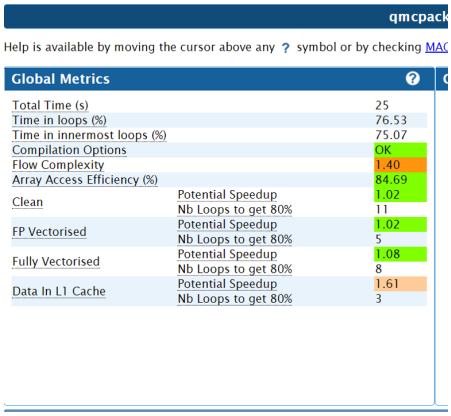

Fig. 1. Example of ONE View global metrics. Satisfactory values are highlighted in green, unsatisfactory ones in orange or red.

Figure 1 presents an example of global metrics. In this case, the expected speedups if cleaning or vectorising the code are low. Conversely, the speedup expected for improving data caching is significantly higher, and would require only 3 loops to be reached at 80%. The average number of paths by loop is 1.4, meaning some improvements could also be expected by simplifying the control flow.

Figure 2 presents an example of a speed-up prediction in the case of perfect data blocking (all data in L1 cache), which shows that a significant improvement can be expected by optimising the data accesses of only 3 loops.

B. Functions and Loop Summary

These views focus on the results gathered from the LProf profiling module. A first view includes a categorization display of the time spent in the application or its external dependencies (MPI, OpenMP, memory management, I/O, ...) and a breakdown of the relative coverage of each loop and function of the application. A second, more detailed view, displays the coverage of each function and of the loops they contain, along with their load distribution across the threads, and the call chains leading to their invocation.

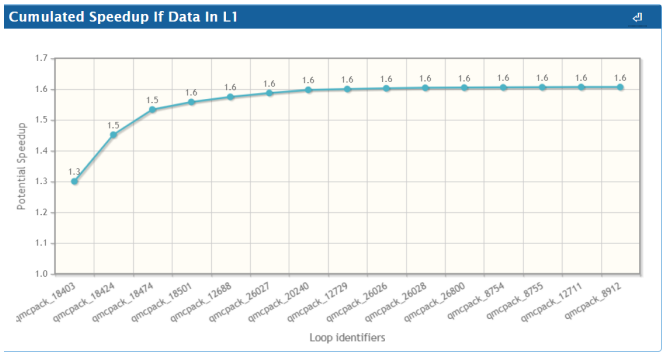

Fig. 2. Example of a projected speed-up. The vertical y-axis displays the cumulative speedup on the whole application which could be obtained by perfect blocking. The horizontal x-axis lists the loops by their decreasing impact in terms of performance gains.

C. Loop Reports

The loop reports describe the performance issues encountered in a given loop, along with hints on how to address them. These hints may advise to modify the compilation chain through flags or directives, or the code to avoid using some instructions, interchange loops, or use specialised functions. These reports also include all metrics gathered by MAQAO along with an estimation of their reliability.

D. Scalability Reports

The scalability views display for each parallel run the speed-up of the application compared to the reference run, and its efficiency, which is the ratio between the speed-up and the ideal speed-up defined as the speed-up expected given the number of threads compared to the reference run. This information is also available at the function and loop levels.

IV. Conclusion and Future Works

MAQAO aggregates a series of metrics into reports presenting a global view of the application performance, along with hints on how to improve it and an estimation of the corresponding gain. Future works will focus on following the evolution of architectures to provide up-to-date information, expanding the analysis capabilities of MAQAO by adding new modules focusing on other aspects of the performance analysis process and further increasing its usability for non performance optimisation experts.

References

[1] S. Koliaï, Z. Bendifallah, M. Tribalat, C. Valensi, J. Acquaviva and W. Jalby,: Quantifying Performance Bottleneck Cost Through Differential Analysis. 27th International ACM Conference on International Conference on Supercomputing, ICS 2013, Eugene, Oregon, USA.

[2] Z. Bendifallah, W. Jalby, J. Noudohouenou, E. Oseret, V. Palomares and A. Charif-Rubial: PAMDA: Performance Assessment Using MAQAO Toolset and Differential Analysis. 7th International Workshop on Parallel Tools for High Performance Computing, September 2013, ZIH, Dresden, Germany.

[3] E. Oseret, A Charif-Rubial, J. Noudohouenou, W. Jalby, G. Lartigue: CQA: A code quality analyzer tool at binary level. 21st International Conference on High Performance Computing, HiPC 2014, Goa, India, December 17-20, 2014.