Towards scalability improvements of HybriD-GM quantum simulator

João Vítor Venceslau Coelho[1], Anderson Braga de Ávila[2], Samuel Xavier de Souza[3], Renata Hax Sander Reiser[2]

[1]Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN)
Natal - RN - Brasil

[2]Centro de Desenvolvimento Tecnológico
Universidade Federal de Pelotas (UFPel)
Pelotas - RS - Brasil

[3]Departamento de Engenharia de Computação e Automação
Universidade Federal do Rio Grande do Norte (UFRN)
Natal - RN - Brasil

joao.vitor.venceslau.701@ufrn.edu.br, abdavila@inf.ufpel.edu.br,
samuel@dca.ufrn.br, reiser@inf.ufpel.edu.br

Abstract. *Currently quantum computers require specialized hardware to be used, until they become more feasibly the testing of quantum algorithms is done by simulation, which is a task of high time and space complexity. To address this issue, the HybriD-GM simulator, and many others around the world, are under research and improvement. This paper is an ongoing work on improvements for the multi-core support of the HybriD-GM simulator. The experiments are done with the Shor and Grover algorithms, while varying the number of cores and qubits for each. To improve the scalability the removal of critical regions by the usage of tasks and other OpenMP resources is exploited, and the study about the impact of others variables on the simulator is already planned, as well improvements for future works on the GPU, and GPU-CPU approaches and expansion to distributed computing for horizontal scalability of the system.*

1. Introduction

At the moment quantum computers are still in their early days, requiring specialized and expensive hardware to be used. This way, until quantum computers become more feasible, the development and testing of quantum algorithms without this specialized hardware can only be done by simulating quantum computing on classical computers. However the simulations of generic quantum algorithms on classical computers are tasks of high temporal and spatial complexity.

To address this issue, many simulators are under research and improvement, such as the Intel Quantum Simulator [Guerreschi et al. 2020], the staq quantum toolkit [Amy and Gheorghiu 2020], the QuEST quantum toolkit [Jones et al. 2019], our HybriD-GM simulator [Avila et al. 2023] and many others.

The HybriD-GM simulator explores high-performance computing (HPC) to improve performance by:

- optimizing resources and enabling hardware-independent scalability;
- exploiting composition and projection operators;
- including coalescing memory management;

These strategies act on quantum structures to obtain a hybrid structure that integrates CPU and/or GPU architectures.

The current work to improve the scalability of the simulator focus on multi-core support, the GPU and hybrid support will be addressed in future work. This way, the vertical scalability of the simulator on multi-core hardware is the target of improvement. The organization of this report is as follows: first, we have this introduction about the theme and focus of the research; next, the methodology, where the steps to achieve an increase in scalability are detailed; then, the partial results obtained until now are analyzed and discussed. At the end, some final considerations about the present work.

2. Methodology

The first step was to obtain a baseline to compare our future results with, keeping the consistency on hardware and software used. To achieve this pre-requisite we used the facilities of the Núcleo de Processamento de Alto Desempenho (NPAD) from UFRN, using the following hardware and software:

- 2 x CPU Intel Xeon Sixteen-Core E5-2698v3 with 2.3 GHz/40M cache/ 9.6 GT/s
- 128GB of RAM DDR4 2133
- Operational System: Rocky Linux 8.5 (Green Obsidian)
- Compilers: clang++ 13.0.1 and nvcc V11.7.64
- Library: OpenMP 4.5

The experiment consists of the simulation of Shor's and Grover's quantum algorithms. They were compiled using the flags '-fopenmp -Ofast' to utilize the OpenMP library and compiler optimizations. The tests were executed varying the number of cores (1, 2, 4, 8, 16, 32) and the number of qubits, this is, for each combination of a number of cores and a number of qubits the algorithms were executed. While the values for the 'cpu_region' and 'cpu_coalesc' were fixed at 14 and 11, these two arguments are related to memory management by coalescing the qubits to increase the cache hit-rate, in an effort to achieve better performance. The exact values for qubits used on the experiments were:

- Shor Quantum Algorithm (qubits, execType, seed):
  (15 1 0), (17 1 0), (19 1 0), (21 1 0), (23 1 0), (25 1 0), (27 1 0)
- Grover Quantum Algorithm (qubits, execType, seed):
  (22 1 0), (23 1 0), (24 1 0), (25 1 0), (26 1 0)

The simulator algorithm accountable for distributing the load of computation between threads has a strategy of tiling to group the data in batches to each thread acts upon, the tiling factor used on multi-core execution is determined by the 'qubits', 'cpu_region' and 'cpu_coalesc' parameters. The operators inside the coalesced region are grouped with the operators outside until reach the region limit to form a 'region_mask' that is used to identify the tiles. The number of tiles is given by: $(qubits - region)^2$.

The baseline implementation achieves the distribution of tasks by using critical regions in each thread to compute the 'region_mask' to the tile that must be computed. Our first modification was to remove these critical regions, changing the way the tiles were

given to the threads. Pre-computing the ‘region_mask‘s and storing them in an array, then looping by each ‘region_mask‘ on a ‘omp parallel for‘, this way the critical regions were removed and a finer control on the load balancing of the problem is possible, by changing the scheduling algorithm utilized, at the cost of this pre-computing of the ‘region_mask‘, our other approach was to each ‘region_mask‘ computed create a OpenMP task, this way we don’t need to store the ‘region_mask‘s and the threads can start working while these ‘region_mask‘s are computed.

The study of the impact of changing the scheduling algorithm and the usage of thread affinity on the simulator is another ongoing work, together with studies about the ideal values for tiling for better memory management and cache use.

3. Analysis and Discussion of Partial Results

The analysis of the results was done with the help of the PaScal Viewer [da Silva et al. 2019], a tool to generate a visualization of the efficiency trends of the simulator, helping with the identification of hot spots and opportunities for refactoring the code to achieve a greater performance. The tool offers visualizations about efficiency, scalability, load balancing, strong scalability and, weak scalability. On the y-axis of the figures, the number of cores is increased from 2 to 32 cores, and on the x-axis, the problem size (number of qubits) is increased from 21 to 25 qubits.

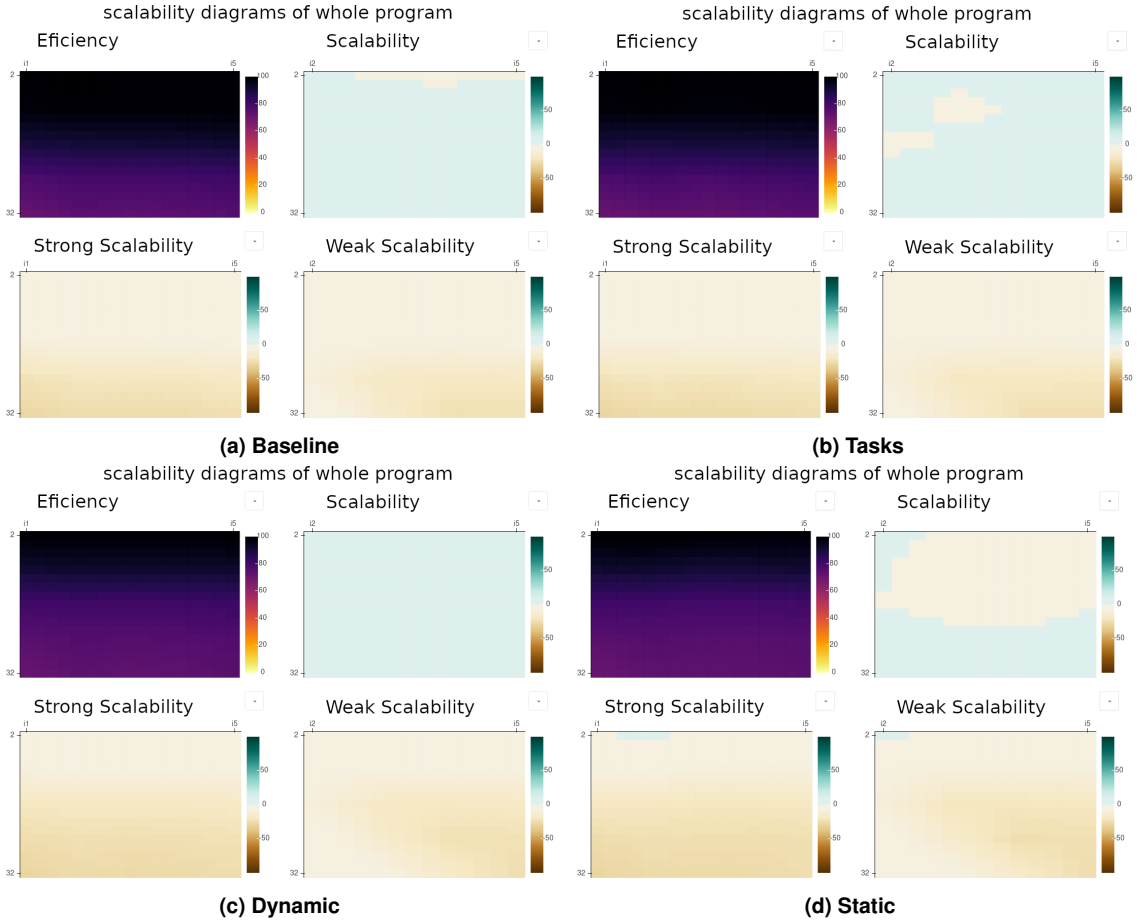


Figure 1. Efficiency and scalability of the Grover Algorithm (from PaScal Viewer)

The baseline already had a high efficiency, but the critical regions utilized are an undesirable approach because can become a bottleneck when many threads are utilized. The tasks approach retained a similar efficiency and scalability, but without the usage of the critical regions, which is a positive result. While the dynamic and static approaches, where an array of 'region_mask' are pre-computed, unfortunately, resulted in a decreased efficiency, the static approach had a bigger impact on the scalability too, as we can observe in the Figure: 1d, top right corner, on scalability, where a large yellowish hole is present in what in the other approaches is mostly bluish color.

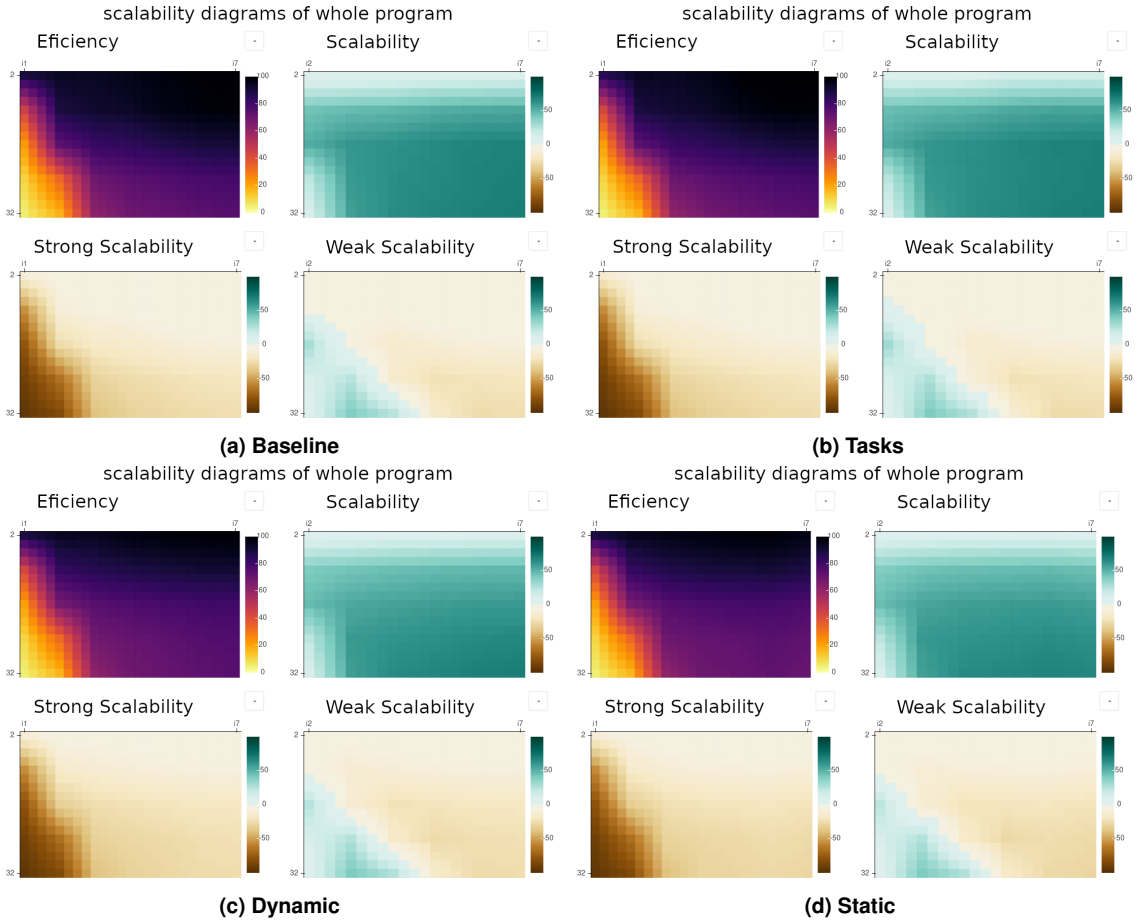
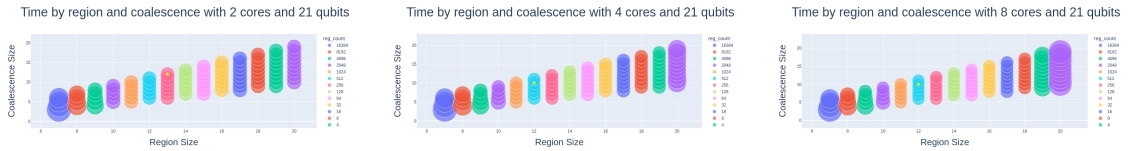


Figure 2. Efficiency and scalability of the Shor Algorithm (from PaScal Viewer)

The impacts of the changes for the usage of tasks or the use of dynamic and static scheduling on the Shor algorithm were very similar to the impacts on the Grover algorithm previously analyzed. The use of tasks maintained a similar efficiency with the benefit of removing the critical region, while the dynamic and static approaches had a decrease in efficiency. However, the static approach this time did not have a visible impact on the scalability, maintaining a similar overall performance. The next step in the research is to analyze the impact of the number of regions and coalescence to achieve better results in the simulations. Some preliminary studies indicate that as the number of cores increases, the larger the region, the longer the simulation run time.



(a) Shor 2 cores and 21 qubits **(b) Shor 4 cores and 21 qubits** **(c) Shor 8 cores and 21 qubits**

Figure 3. Time of Shor on variation of regions and coalescence with 21 qubits

4. Final considerations

The current work presented an brief introduction about the quantum computing challenges with simulations, proposal and scope of the research on increasing the scalability of the Hybrid-GM, the methodology utilized, describing the hardware where the simulations were done and the inputs for each algorithm tested. The changes on the simulation approach for parallelization were discussed, the problem of the presence of critical regions on the computation of the 'region mask's and the possible solutions proposed. Then the analysis of the partial results obtained, with graphical visualizations about the efficiency and scalability of the simulator on the two algorithms, at the end the study about the impact caused by the values of region and coalescence size, where the relation about the increase of cores with the region and run time.

5. Funding

This research was supported by the High-Performance Computing Center at UFRN (NPAD/UFRN) and by the CNPq through a technological development scholarship to JVVC (300094/2023-0). The funders played no role in the study design, data collection and analysis, or the decision to publish or prepare the manuscript.

References

Amy, M. and Gheorghiu, V. (2020). staq: A full-stack quantum processing toolkit. *Quantum Science and Technology*, 5(3).

Avila, A., Santos, H., Cruz, A., de Souza, S. X., Lucca, G., Moura, B., Yamin, A., and Reiser, R. (2023). HybriD-GM: A framework for quantum computing simulation targeted to hybrid parallel architectures. *Entropy*, 25(3).

da Silva, A. B. N., Cunha, D. A. M., Silva, V. R. G., de A. Furtunato, A. F., and de Souza, S. X. (2019). PaScal viewer: A tool for the visualization of parallel scalability trends. In *Programming and Performance Visualization Tools*, pages 250–264. Springer International Publishing.

Guerreschi, G. G., Hogaboam, J., Baruffa, F., and Sawaya, N. P. D. (2020). Intel quantum simulator: a cloud-ready high-performance simulator of quantum circuits. *Quantum Science and Technology*, 5(3).

Jones, T., Brown, A., Bush, I., and Benjamin, S. C. (2019). QuEST and high performance simulation of quantum computers. *Scientific Reports*, 9(1).