Atlantis Studies in Uncertainty Modelling, volume 1

11th Conference of the European Society for Fuzzy Logic and Technology (EUSFLAT 2019)

Towards Memory Access Optimization in Quantum Computing

Mateus Nascimento, Anderson de Avila, Renata Reiser, and Maurcio Pilla
PPGC – CDTEC, UFPEL, Pelotas, Brazil
{mmsdnascimento, abdavila, reiser, pilla}@inf.ufpel.edu.br

Abstract

Due to the lack of quantum hardware, quantum simulation (QS) is still the most common method to study quantum computing. Due to the great number of operations and memory requirements, simulators should be properly built to efficiently explore modern classic computers. This work focuses on the memory access requirements that present a bottleneck not just for QS, but for computer science research as a whole. Our proposed methodology collaborates with the memory usage reduction, mitigating the effects of the memory wall. Our results show improvements in memory usage and the overall speed of circuit simulations. Our case test in the D-GM simulator with controlled operators, such as dense, primary and secondary diagonal patterns, showed an improvement for execution time in the order of 50 %. Our methodology can be applied not only in QS, but also for related applications with complex memory access patterns.

Keywords: Quantum computing, Quantum simulating, D-GM environment.

1 Introduction

One of the greatest challenges when simulating quantum computing (QC) is dealing with memory issues such as exponential growth of states [9] and complex patterns that impact the exploration of temporal and spatial locality.

Due to the state space of quantum computing, more complex quantum algorithms (QA) imply in the expansion of quantum states. Quantum gates are modeled and applied as the tensor product over the matrices representing the quantum states, thus the number of operations greatly increases with the size of the global state. However, even with these pitfalls, simulation of QC is still one of the best ways to study and develop QA [14].

The D-GM (Distributed Geometric Machine) environment [2] has been developed providing optimizations for QC simulation in order to allow the simulation of large QA with faster execution time. In addition, D-GM has the ability to use GPU to perform heavy calculations in parallel with the main CPU, thus making it a hybrid architecture.

1.1 Main contribution

This paper is an original contribution to confront the following research problem: How can quantum simulators improve the way memory is addressed in quantum data structures considering the analysis of previously recorded patterns?

Aligned with this problem statement, in this work a *methodology to analyze quantum transformations focusing on the possibility of memory access improvements* is developed. In addition, this paper describes the software prototype development and validation tests.

This work also considers the data analysis related to memory behavior of quantum systems, dealing with techniques of memory access in the application of unitary and controlled operations.

The analysis module developed in this work was used with the D-GM simulator as a study case, since D-GM provides a consolidated framework where quantum transformations (QT) are performed. After analysing the behavior of different controlled gates, modifications to the simulator were implemented, improving performance by about 50%.

1.2 Paper Outline

This paper is organized as follows. The Introduction describes quantum simulators, considering how they handle memory access. Section 3 introduces the D-GM simulator, which is used throughout this paper. Section 4 describes the new methodology focusing on reducing the quantity of memory access on quantum circuits simulations, also, it is explained and demonstrated how this methodology was idealized and applied, with the demonstration of its results. Finally, Section 5 presents our conclusions.

2 Related Works

In this Section, we briefly introduce how three simulators represent and address their global states.

2.0.1 QuIDDPro Quantum Simulator

The simulator QuIDDPro proposed in [15] uses structures called *QuIDDs* (*Quantum Information Decision Diagrams*) as an efficient way to represent multi-dimensional QT and states, represented by matrices which have replicated blocks for identical values. The patterns which are constant in many algorithms can be used more than once, allowing a controlled redundancy which results in significant reduction of memory usage and in data accessing time.

QuIDD provides a compressed matrix and vector representations, allowing computations to be performed directly over these optimized structures. In such simulator performance, the idea is that memory consumption increases greatly as the number of states of a quantum system grows, reducing the memory usage is a complex task, since it is an exponential problem. This approach has shown improvements when compared to other simulators, being faster and using significantly less memory (as pointed out by [1, 16]).

2.1 PVLIB Quantum Simulator

The simulator developed by Samoladas [13] is named PVLIB, which extends the *Multi-terminal Binary Decision Diagrams* (MTBDDs) [7] to efficiently represent QTs and quantum states, this approach is similar to the QuIDDPro (described in 2.0.1). Even though the PVLIB consists of a symbolical representation of QTs instead of matrices representation, either compressed (QuIDDPro) or not.

In this way, an MTDBB is used to represent the space of states of a quantum system, while its own structures are adopted to represent QTs. Additionally, evolution of the system does not require operations between two graphs with complex structures, but with recursive functions working through the space of states.

2.2 LIQUi$|\rangle$Quantum Simulator

LIQUi$|\rangle$is a research project from Microsoft [17] providing a software architecture for quantum computing which is hardware independent. It has its own language, projected with the sole purpose of quantum circuits development, having F# as its host language. Also, it allows data structure extraction from circuits, which can be used for optimization, generating a compact version, with better potential for simulation.

Two distinct simulation environments are available to users, allowing a trade-off between the number of qubits and classes of operations. The simulator is highly optimized, and is able to use a vast number of techniques: memory management, parallel execution, gate growing, and virtualization e.g., it can be simulated using cloud environment.

3 Distributed Geometric Machine

D-GM [5] allows simulation of quantum circuits through GPU, as an initial enforce to use hybrid architectures to simulations. In order to achieve better results with parallel simulations, the D-GM environment is structured using two main frameworks [4]:

(i) VPE-qGM (Visual Programming Environment for Quantum Geometric Machine Model) [10] as the development environment, a graphical interface providing graphical resources to quantum circuits mainly related to unitary and controlled QT; and

(ii) VirD-GM (Virtual Distributed Geometric Machine Model) [3] as the execution environment, handles tasks such as scheduling, communication, and synchronization when computations require a distributed simulation.

With all its features, the D-GM has a graphical interface, the possibility to simulate quantum circuits through parallel or sequential steps. Furthermore, new options are being added to the D-GM, such as using the GPU to simulate, and improvements [5]. Finally, Figure1 presents the D-GM framework organization with its distinct levels, each with its own purpose, with all of them working together to design and simulation of quantum circuits.

In order to handle memory management the D-GM simulator employs two vectors to work with operators: (i) the input vector, containing the actual state of the quantum system; and (ii) the output vector, receiving the new values resulting from amplitudes. While simulating quantum circuits, new amplitudes are generated through the most recent iteration, since new states have to be considered as operations continue.

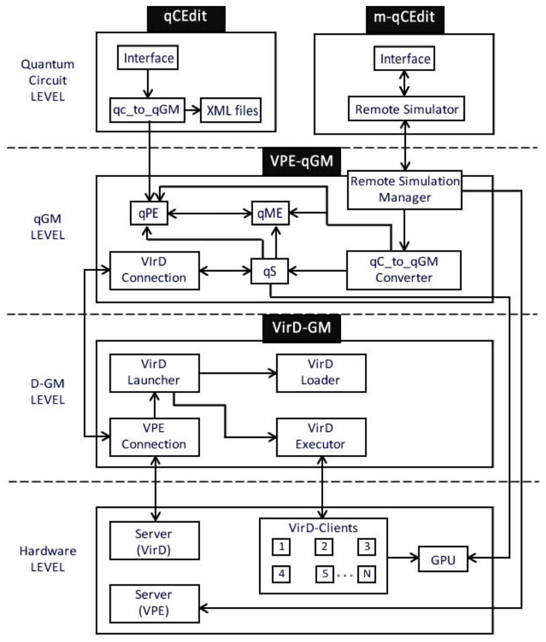


Figure 1: D-GM architecture.

These new amplitudes are added to the output vector. As the operations proceed, the output vector becomes the input vector from the upcoming operations.

There are two relevant optimizations in the D-GM, the Reductions and Decompositions aiming at reducing the amount of memory necessary to perform QT.

Reductions focus on QT, mostly decreasing the stress of tensor products ($\otimes$) [6] by using the *Id*-operation. The *Id*-operation works replicating values of other operators, also they allow a better sparsity than other QT. When applying a tensor product with other QT, the main diagonal of the *Id*-operation keeps unitary values (1), generating data replication from other operators, while the secondary diagonal having null values (0) is used for the spatially in the resultant matrix.

Meanwhile, decomposition operators work by decomposing a QT, increasing the number of steps for computation, and allowing to control the amount of *Id*-operators in each step while preserving their behavior and properties. For instance, the operator $H \otimes H$ can be expressed as a decomposition in two steps, $H \otimes I$ and $I \otimes H$, keeping the same behaviour regardless their composition order.

Controlled QTs can also be decomposed keeping the controls associated with the operators in the computations. As the number of operations grows, the replication of previous states also grows. It is possible to replicate the operations, and even with larger matrices, most of it consists of copying the previous state. Furthermore, with the null values and the same operations, it is possible to take advantage of spatial and temporal locality. These two aspects are the main focuses, since read/store cannot be avoided.

The presented simulators showed distinct approaches towards the quantum simulation. These distinctions are important to choose the best options while working with QC, since better simulators are needed as the complexity of circuits grows. Further, the problem with memory was evident, which is still being a problem that cannot be fixed, only reduced.

For those interested in more comprehensive comparison of D-GM with QC simulators, see [11] for simulators which are being developed for some years, and [12] or [8] for more recent approaches.

4 Memory Access Optimization

In order to optimize the D-GM memory access during simulations, it was first necessary to study a memory pattern generated by three types of operators since they access the memory in distinct ways. To accomplish that, a module was developed to capture the D-GM memory access during an execution and convert it to a graphic visualization in order to facilitate the memory access analysis.

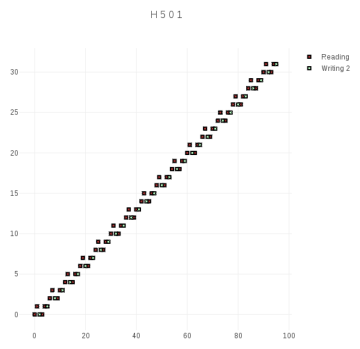
The memory access patterns in D-GM graphs for circuits with 5 qubits were produced, considering 3 types of operation (dense matrix, primary diagonal matrix, and secondary diagonal matrix). It had unitary and controlled operations, where the qubit with the operation and the one which has a control were changed.

In the following graphs (Figure 2), the horizontal axis (X-axis) corresponds to the execution time of quantum circuit simulation, meanwhile, the vertical axis (Y-axis) has positions from each vector (32 positions are available since 5 qubits are used). Last, the graphs have marks with distinct colors for reading and writing operations.

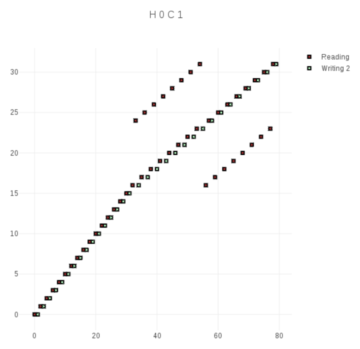
In this article, the graphical approach of unitary operations will be shown with operation on 0-qubit (having 5 qubits). While controlled gates will have just one control, the operation will be in the 0-qubit, and control on 1-qubit. Unfortunately, the high number of graphs will not allow to show them all.

See the three composing graphs in Figures 2(a), 2(b) and 2(c) one for each pattern, displaying the projected operation going on in qubit 0 and describing the allocated memory being accessed during execution time.

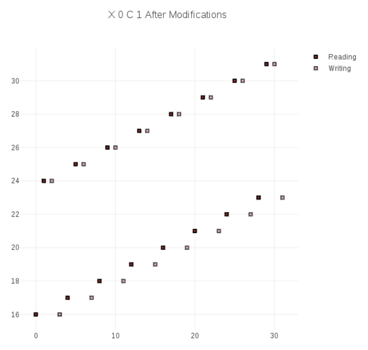
The unitary gates have a simple access pattern as they do not depend on external factors as controlled operations. For each graph, the operation qubit was dislocated to the next qubit (through all available posi-



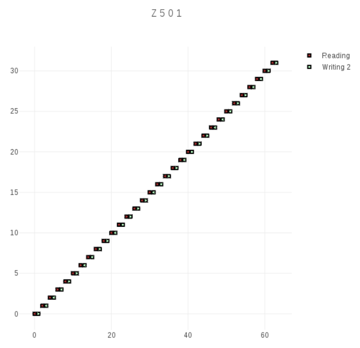
(a) Unitary dense



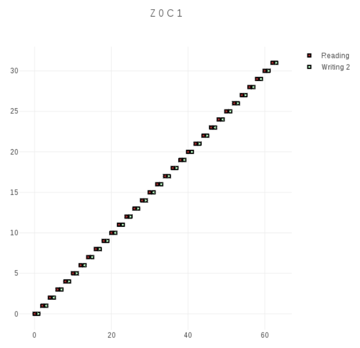
(d) Controlled dense



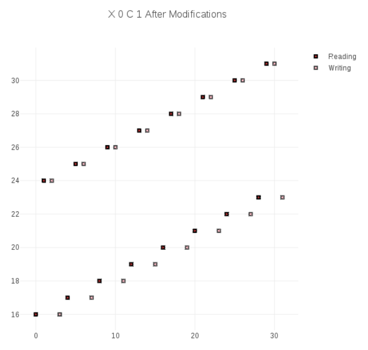
(g) Controlled dense after modifications



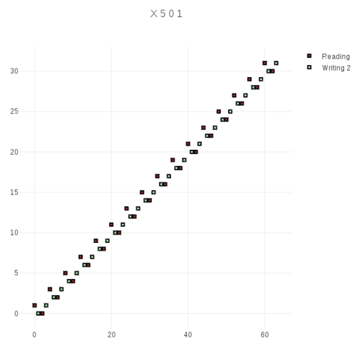
(b) Unitary primary diagonal



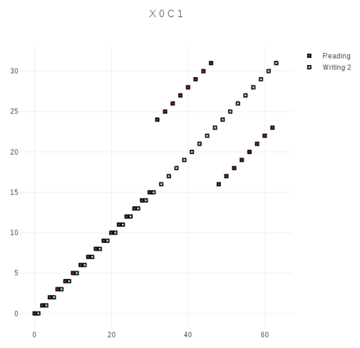
(e) Controlled primary diagonal



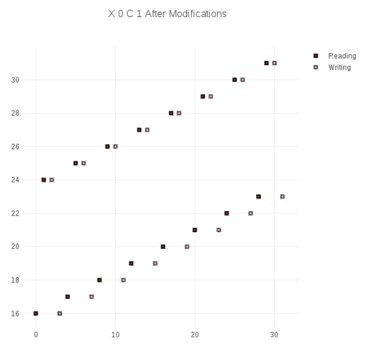
(h) Controlled primary diagonal after modifications



(c) Unitary secondary diagonal



(f) Controlled secondary diagonal



(i) Controlled secondary diagonal after modifications

Figure 2: Comparison between previous and modified dense operator simulations in D-GM

tions), producing 5 graphs for every kind of operation (a sum of 15 in the end).

The extensive study of controlled gates present more than one pattern according to whether the operation happens or not. Furthermore, when working with controlled gate two qubits have to be considered: the qubit where the operations are happening, and the controlled qubit (responsible for determining whether the operation happens or not). Then, our objective was to simulate all possibilities with 28 qubits. It was necessary to place the operation in one qubit, and analyze it while changing where the control was. In the end, for each gate, a sum of 20 graphs was produced.

See Figures 2(d), 2(e) and 2(f) showing samples of graphs obtained from controlled gates, each subfigure representing one of the patterns, whenever operations happen in 0-qubit while the control is on 1-qubit.

Information extracted from these graphs interpretation helps to improve the D-GM performance through memory access optimization.

By comparing the graphs for non-controlled and controlled gates, one can observe that for controlled gates only certain blocks of amplitudes have to be computed while other blocks are only copied from the input vector to the output vector once their amplitudes do not satisfy the control value.

As a result, removing one vector established a new pattern: only one vector for input and output, performing the computations in-place and thus avoiding unnecessary memory access, once the blocks which had been copied before will be in their place already.

Another optimization was to avoid verifying if an amplitude satisfies the control value (as the previous code), only then to perform the computation, by working directly with the blocks of amplitudes which need to be computed. As a result, information related to the block size, and stride are used to implement the new functions for each operator:
1. Primary diagonal: implemented a loop to go through the blocks, and an internal loop to compute each amplitude in a block.
2. Dense matrix and secondary diagonal: for these operators the computation of a amplitude must access other amplitudes besides itself.

Once the computation has to happen in-place, the amplitudes must be computed in pairs (according to the index of the target qubit) and use a temporary variable to resolve their dependencies.

Besides, a pair of amplitudes belongs (does not belong) to the same block if the index of the target qubit is previous to the index of the controlled qubit.

For the posterior case, the blocks are organized in pairs with correspondent positions forming a pair of amplitudes.

Putting all together, the implemented functions are split in two parts:

(i) “Previous” - consisting of a loop to go through the blocks and an internal loop to go through each pair of amplitudes in a block (determined by the target qubit) and computing them; and

(ii) “Posterior” - with a loop going through pairs of blocks (determined by the target qubit) and an internal loop computing a pairs of amplitudes formed by correspondents positions in the block pair.

Additional features improving memory usage introduces a new set of graphs (making a comparison before and after making changes).

From comparisons between Figures 2(d), 2(g) and Figures 2(f) to 2(i), and last Figures 2(e) and 2(h), one can realize a change in the number of memory accesses. This change was occasioned by the modifications made, mostly because of the second vector removal, and reusing the same in further loops interactions. Thus, the reduced number of access means an improvement over the memory allocation for the D-GM data structures.

In order to test this assumption, graphs of speedup are developed. Tests comprise 28 cases with 28 qubits, with each graph changing where the operation is happening, except when it happens in the control qubit, meaning in 0, 7, 13, 20 and 27-qubit.

Finally, for each case 30 simulations were made, taking the averages and the standard deviation as displayed Figure 3. Table 1 sums up the overall speed improvement.

Operation	Dense Op.	Prim. Diagonal	Sec. Diagonal
Standard	7.3s	4.5s	4.9s
Optimized	4.0s	2.2s	2.0s

Table 1: Comparison between execution times from D-GM with 28 qubits before and after modifications

Therefore, an overall improvement in execution times around 50 % faster towards the selected operations was achieved.

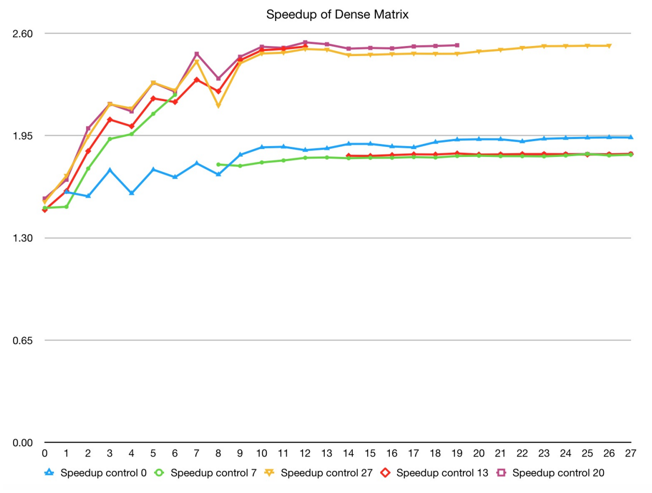
Furthermore, the module of analysis developed to assist this research can be easily modified to accommodate other types and program patterns, as a new resource helping to better understand the access memory working with data structures.

So, the developed methodology provides a coherent way to improve QC, based on assisting memory through a program execution.

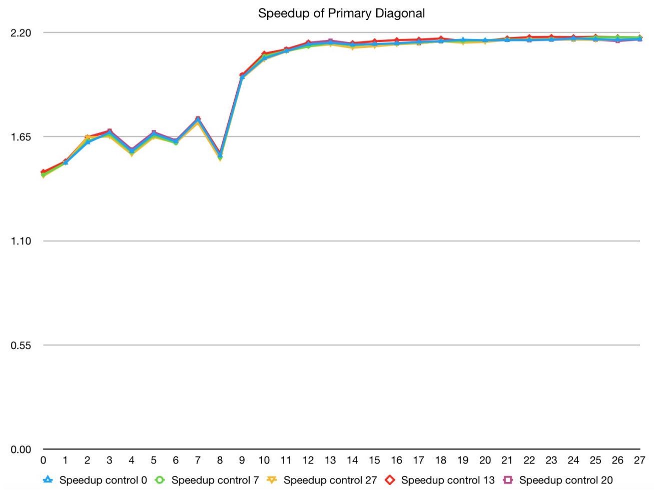
5 Conclusion

The QS in classic computers is a huge task requiring a considerable amount of efforts, since neither the necessary quantum hardware is available, nor the full potential of the machines. However, this is one of the most viable ways to study QA and also encourage more studies about the subject.

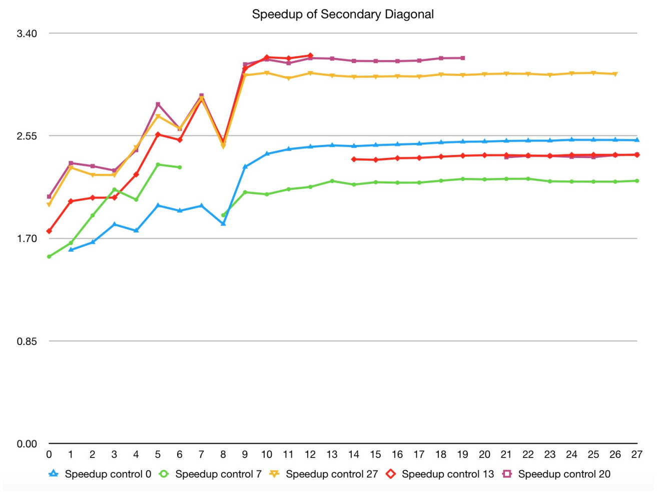
One constant problem for simulators is the inefficient use of the available resources. For this reason, the D-GM environment is trying to approach strategies, enabling the parallelism and the efficient usage of distinct peripheral and also allowing a better application of resources and improvement in performance as well.



(a) Speedup dense matrices.



(b) Speedup primary diagonals.



(c) Speedup secondary diagonals.

Figure 3: Speedup graphs of QT.

In this article, a methodology to improve performance in QC simulations is applied to the D-GM environment, a general-purpose quantum simulator. The goal was to improve memory bandwidth and latency through changing access patterns for operations. In order to achieve this goal, an analysis module was developed, allowing a perception of how quantum operations take place in the memory during a simulation.

Collected information used only the operations that could help in the process of improving the performance of the D-GM (the set of data contained distinct types of operation for each pattern). The focus was on controlled operations with a single control.

Changes in the D-GM source code resulted from this step. As the main result, it was cut the amount of memory used by the D-GM environment. As mentioned before, the results of this change is summarized by table 1: a time of execution gain in the order of 50 % for each kind of operation.

Furthermore, the methodology developed here worked in conjunction with the D-GM, it also may be able to work with other programs with small modifications, creating a way to optimize memory usage.

Acknowledgement

This work was supported by CAPES/Brasil, Brazilian Funding Agency CAPES, MCTI/CNPQ Universal (448766/ 2014-0) and PQ(310106/ 2016-8) and PqG/FAPERGS 02/2017(17/2551-0001207-0).

References

[1] S. Aaronson, D. Gottesman, Improved simulation of stabilizer circuits, Physical Review A 70 (5) (2004) 052328.

[2] A. Avila, A. Maron, R. Reiser, M. Pilla, A. Yamin, Gpu-aware distributed quantum simulation, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, SAC '14, 2014, pp. 860–865.

[3] A. Avila, A. Maron, R. Reiser, M. Pilla, A. Yamin, Gpu-aware distributed quantum simulation, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 860–865.

[4] A. B. Avila, M. F. Shmalfuss, R. H. S. Reiser, M. L. Pilla, A. K. Maron, Simulação distribuída de algoritmos quânticos via gpus, in: Anais do XV Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD), SBC, São José dos Campos, 2014, pp. 1–12.

[5] A. B. de Avila, R. H. Reiser, M. L. Pilla, Quantum computing simulation through reduction and decomposition optimizations with a case study of shor's algorithm, Concurrency and Computation: Practice and Experience 29 (22) (2017) e3961.

[6] A. B. de Avila, R. H. S. Reiser, A. C. Yamin, M. L. Pilla, Scalable quantum simulation by reductions and decompositions through the id-operator, in: Proceedings of the 31st Annual ACM Symposium on Applied Computing, ACM, 2016, pp. 1255–1257.

[7] M. Fujita, P. C. McGeer, J.-Y. Yang, Multi-terminal binary decision diagrams: An efficient data structure for matrix representation, Formal methods in system design 10 (2-3) (1997) 149–169.

[8] I. Georgescu, S. Ashhab, F. Nori, Quantum simulation, Reviews of Modern Physics 86 (1) (2014) 153.

[9] R. LaRose, Distributed memory techniques for classical simulation of quantum circuits, arXiv preprint arXiv:1801.01037.

[10] A. Maron, A. Pinheiro, R. Reiser, A. Yamin, M. Pilla, Ambiente vpe-qgm: Em direçao a uma nova abordagem para simulaçoes quânticas, Revista do CCEI 14 (2010) 29–46.

[11] A. K. Maron, R. H. Reiser, M. L. Pilla, Um estudo das possibilidades de otimizaçao para simulaçao quântica.

[12] J. Preskill, Quantum computing in the nisq era and beyond, arXiv preprint arXiv:1801.00862.

[13] V. Samoladas, Improved BDD algorithms for the simulation of quantum circuits, in: European Symposium on Algorithms, Springer, 2008, pp. 720–731.

[14] T. Schaetz, C. R. Monroe, T. Esslinger, Focus on quantum simulation, New Journal of Physics 15 (8) (2013) 085009.

[15] G. F. Viamontes, Efficient quantum circuit simulation, Ph.D. thesis, The University of Michigan (2007).

[16] G. Vidal, Efficient classical simulation of slightly entangled quantum computations, Physical Review Letters 91 (14) (2003) 147902.

[17] D. Wecker, K. M. Svore, LIQ$Ui|\rangle$: A Software Design Architecture and Domain-Specific Language for Quantum Computing, Available from: <`http://arxiv.org/abs/1402.4467`>. Accessed: April, 2018. (2014).