Implementation of Grover's Quantum Search Algorithm using Rigetti Forest

Anthony, Arya Wicaksana

Abstract: ***Grover's quantum search algorithm allows quadratic speedup in unsorted search problem by utilizing amplitude amplification trick in quantum computing. In this paper, an approach to implement Grover's quantum search algorithm is proposed. The implementation is done using Rigetti Forest and Python. The testing and evaluation processes are carried on in two computers with different hardware specifications to derive more information from the result. The results are measured in user time and compared with implementation from Quantum Computing Playground. The user time of this implementation for 10 qubits and 1024 data is slower compared to Quantum Computing Playground's implementation. The proposed implementation can be improved by calculating the probability of Grover's quantum search algorithm in finding the appropriate search result.***

Index Terms: Grover, quantum algorithm, Rigetti forest, unsorted search.

I. INTRODUCTION

Moore's law said that the number of transistor in an integrated circuit would be doubled every 18 months and that would yield to faster processing speed [1]. When the size of a transistor sinks into the size of its atom, the electron no longer follows the principles of classical physics [2]. The electron may shoot-through from one transistor to another. Quantum computer was proposed to overcome that quantum phenomenon [3].

One application of quantum computer is to search a data in an unsorted database. This search problem can be done using Grover's quantum search algorithm in $O(N^{1/2})$ [4]. Grover's quantum search algorithm allows quadratic speedup [5] in unsorted search problem by utilizing amplitude amplification trick in quantum computing. Grover's quantum search algorithm has been successfully carried out in classic computer using Quantum Computer Language (QCL) [6].

Quantum Virtual Machine (QVM), such as Rigetti Forest can be used to do quantum programming and computational simulations in classic computer [7]. PyQuil library is needed to execute the Quil program on Rigetti Forest platform [7]. Quil is an instruction-based language to represent quantum computing with classical control and

feedback [8]. Quil can be used for quantum programming, as an intermediary format in classical programs, or target compilers for quantum programming languages [8]. In addition, Rigetti Quil Compiler can be used to compile and optimize Quil program [7].

Research related to the implementation of Grover's quantum search algorithm can be found on the Quantum Computing Playground (QCP), which is a web-based WebGL Chrome Experiment [9]. Quantum Computing Playground was created by a group of Google engineers in 2014 [9]. QScript is used in Quantum Computing Playground for the scripting language and compiler [9]. Quantum Computing Playground has one quantum register with size between 6 and 22 qubits [9].

This paper describes the implementation of Grover's quantum search algorithm using Rigetti Forest and Python. The testing uses five test scenarios and the evaluation of the performance is done using two computers with different hardware specifications. This is done to derive information on how the performance of the implementation would be differ between the two computers, since the execution of the implementation is carried out using a quantum simulator [10]. The performance results are measured by the execution time, specifically the user time [11], [12]. The result is compared with QCP's implementation.

II. METHODS

The problem that is solved by Grover's quantum search algorithm is that there is N data where only one data that meets the requirements, and that data must be obtained [13]. The input of this algorithm is $N=2^n$ states where n is the number of qubits [13]. These states are represented in n bitstring and there is a state that meets the requirements [13]. The result is a state that meets the requirements.

By using a unitary matrix called Oracle, data can be represented in qubits on quantum computers [14]. The Oracle function returns 1 for the appropriate data, otherwise, it returns 0. Oracle matrix U_f can be defined as:

$$U_f|x\rangle = (-1)^{f(x)}|x\rangle \quad (1)$$

This algorithm starts with computers in state $|0\rangle^{\otimes n}$ [5]. Hadamard transform is performed on each qubit to make the computer be in the state superposition as:

$$|\varphi\rangle = \frac{1}{\sqrt{N}}\sum_{x=0}^{N-1}|x\rangle \quad (2)$$

After that, each state has the same amplitude, that is $1/N^{1/2}$. This step is obtained in $O(\log N)$ [13].

The next step is the amplitude amplification process, where amplitude in

Revised Manuscript Received on September 22, 2019.

Anthony, Department of Informatics, Universitas Multimedia Nusantara, Tangerang - 15810, Indonesia.
Arya Wicaksana, Department of Informatics, Universitas Multimedia Nusantara, Tangerang - 15810, Indonesia.

the appropriate state will be amplified, while the amplitude in the inappropriate state will be minimized [15]. This amplification process is obtained from repetitions of $\frac{\pi}{4}\sqrt{N}$ [16]. The repetition contains Oracle functions and Diffusion transform [13]. The Oracle function will reverse the appropriate amplitude state, while the inappropriate amplitude state does not change. The Diffusion transform will reverse amplitude around the average amplitude. Diffusion matrix *D* can be defined as:

$$D_{ij} = \frac{2}{N} if\ i \neq j\ and\ D_{ij} = -1 + \frac{2}{N} if\ i = j \quad (3)$$

After the looping process, there will be a unique state, where the Oracle function returns 1 to that state, which has a probability of more than 0.5 [13].

Implementation of Grover's search quantum algorithm is done using two computers with different hardware specifications. Table I shows the differences between computers. Five test scenarios are created and the details of each scenario are displayed in Table II.

Table- I: Computer specifications

Variable	Computer 1	Computer 2
Core Processor	I7-6700HQ (4 cores)	I7-8700k (6 cores)
Clock Speed	2.6 GHz	3.7 GHz
RAM	16 GB	32 GB

Table- II: Scenario for testing

Case	Possible Values	Dataset	Target Value
1	4	[3, 0, 2, 1]	1
2	8	[2 ,6, 1,0, 3, 7, 4, 5]	2
3	12	[4, 8, 1, 14, 2, 0, 3, 11, 15, 6, 7, 10]	14
4	16	[23, 17, 16, 30, 22, 4, 9, 29, 27, 18, 10, 1, 12, 25, 6, 20] ['0', '+', 9, 54, 20, '*', 44, 41, 36, 3, '=', '!', '?', 60, 57, '1', 5, 32, 17, 13]	25
5	20		3

III. RESULTS AND ANALYSIS

A. Testing

Table III shows the calculation results of maxValue, bitstring, nQubit, and nState for each scenario given in Table II. The maxValue variable is used to store the maximum value from the dataset. The bitstring variable is used to store the form of bitstring from maxValue. The nQubit and nState are variables used to store the number of qubits and the number of states used for searching.

Table- III: Calculation results for maxValue, bitstring, nQubit, and nState

Case	maxValue	bitstring	nQubit	nState
1	3	11	2	4
2	7	111	3	8
3	15	1111	4	16
4	30	11110	5	32
5	63	111111	6	64

Table IV shows the results of the Hadamard gate on each qubit. The number of qubits performed by Hadamard gate according to the value of nQubit variable in each scenario. This is the first step in Grover's quantum search algorithm.

Table- IV: Results of Hadamard gate on each qubit

Qubit	State
0	$\frac{1}{\sqrt{2}}\|0\rangle, \frac{1}{\sqrt{2}}\|1\rangle$
1	$\frac{1}{\sqrt{4}}\|00\rangle, \frac{1}{\sqrt{4}}\|01\rangle, \frac{1}{\sqrt{4}}\|10\rangle, \frac{1}{\sqrt{4}}\|11\rangle$
2	$\frac{1}{\sqrt{8}}\|000\rangle, \frac{1}{\sqrt{8}}\|001\rangle, \ldots, \frac{1}{\sqrt{8}}\|110\rangle, \frac{1}{\sqrt{8}}\|111\rangle$
3	$\frac{1}{\sqrt{16}}\|0000\rangle, \frac{1}{\sqrt{16}}\|0001\rangle, \ldots, \frac{1}{\sqrt{16}}\|1110\rangle, \frac{1}{\sqrt{16}}\|1111\rangle$
4	$\frac{1}{\sqrt{32}}\|00000\rangle, \frac{1}{\sqrt{32}}\|00001\rangle, \ldots, , \frac{1}{\sqrt{32}}\|11111\rangle$
5	$\frac{1}{\sqrt{64}}\|000000\rangle, \ldots, \frac{1}{\sqrt{64}}\|111111\rangle$

Table V, Table VI, Table VII, Table VIII, and Table IX show the calculation of the amplitude amplification process in Grover's quantum search algorithm for each scenario. After that process, the program measures every qubit to get the result.

Table- V: Calculation of amplitude state in amplitude amplification process for Scenario 1

i	Oracle State |01⟩	Oracle Other State	Diffusion State	|01⟩	Diffusion Other State
0	$-\frac{1}{2}$	$\frac{1}{2}$	1		0

Table- VI: Calculation of amplitude state in amplitude amplification process for Scenario 2

i	Oracle State |010⟩	Oracle Other State	Diffusion State	|010⟩	Diffusion Other State
0	$-\frac{1}{\sqrt{8}}$	$\frac{1}{\sqrt{8}}$	$\frac{5}{2\sqrt{8}}$		$\frac{1}{2\sqrt{8}}$
1	$-\frac{5}{2\sqrt{8}}$	$\frac{1}{2\sqrt{8}}$	$\frac{11}{4\sqrt{8}}$		$-\frac{1}{4\sqrt{8}}$

Table- VII: Calculation of amplitude state in amplitude amplification process for Scenario 3

i	Oracle State |1110⟩	Oracle Other State	Diffusion State	|1110⟩	Diffusion Other State
0	$-\frac{1}{4}$	$\frac{1}{4}$	$\frac{22}{32}$		$\frac{6}{32}$
1	$-\frac{22}{32}$	$\frac{6}{32}$	$\frac{122}{128}$		$\frac{10}{128}$
2	$-\frac{122}{128}$	$\frac{10}{128}$	$\frac{1004}{1024}$		$-\frac{52}{1024}$

Table- VIII: Calculation of amplitude state in amplitude amplification process for Scenario 4

i	Oracle State |11001⟩	Oracle Other State	Diffusion State	|11001⟩	Diffusion Other State
0	$-\frac{1}{\sqrt{32}}$	$\frac{1}{\sqrt{32}}$	$\frac{46}{16\sqrt{32}}$		$\frac{14}{16\sqrt{32}}$
1	$-\frac{46}{16\sqrt{32}}$	$\frac{14}{16\sqrt{32}}$	$\frac{562}{128\sqrt{32}}$		$\frac{82}{128\sqrt{32}}$
2	$-\frac{562}{128\sqrt{32}}$	$\frac{82}{128\sqrt{32}}$	$\frac{5486}{1024\sqrt{32}}$		$\frac{334}{1024\sqrt{32}}$
3	$-\frac{5486}{1024\sqrt{32}}$	$\frac{334}{1024\sqrt{32}}$	$\frac{46322}{8192\sqrt{32}}$		$-\frac{238}{8192\sqrt{32}}$

Table- IX: Calculation of amplitude state in amplitude amplification process for Scenario 5

i	Oracle State |000011⟩	Oracle Other State	Diffusion State	|000011⟩	Diffusion Other State
0	$-\frac{1}{8}$	$\frac{1}{8}$	$\frac{47}{128}$		$\frac{15}{128}$
1	$-\frac{47}{128}$	$\frac{15}{128}$	$\frac{2402}{4096}$		$\frac{418}{4096}$
2	$-\frac{2402}{4096}$	$\frac{418}{4096}$	$\frac{50398}{65536}$		$\frac{5278}{65536}$
3	$-\frac{50398}{65536}$	$\frac{5278}{65536}$	$\frac{947426}{1048576}$		$\frac{56610}{1048576}$
4	$-\frac{947426}{1048576}$	$\frac{56610}{1048576}$	$\frac{16468318}{16777216}$		$\frac{403742}{16777216}$
5	$-\frac{16468318}{16777216}$	$\frac{403742}{16777216}$	$\frac{267982802}{268435456}$		$-\frac{1976158}{268435456}$

qubits.

B. Evaluation

User Time Comparison Between Implementations

The user time between this implementation and Quantum Computing Playground (QCP) on Computer 1 is given in Fig. 1. When the possible values is less than or equal to 128 and the number of qubits is less than or equal 7, the user time of this implementation is faster than the Quantum Computing Playground. When the possible values is greater than 128 and the number of qubits is greater than 7, the user time of This implementation is longer than the Quantum Computing Playground's. The longest user time of This implementation is 100.81 second. The longest user time of QCP's implementation is 21.93 second with 1024 data and 10

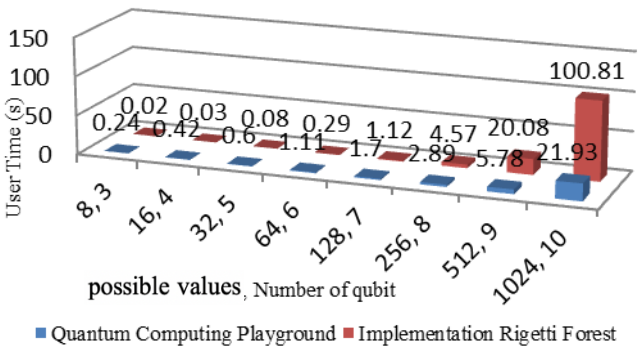

Fig. 1. User time comparison on Computer 1

The user time between this implementation and Quantum Computing Playground on Computer 2 is shown in Fig. 2. When the possible values is less than or equal to 128 and the number of qubits is less than or equal 7, the user time of this implementation is faster than QCP's implementation. When the possible values is greater than 128 and the number of qubits is greater than 7, the user time of this implementation is slower than the QCP's. The longest user time of this implementation with 1024 data and 10 qubits is 62.08 second. The longest user time of QCP's implementation with 1024 data and 10 qubits is 21.04 second.

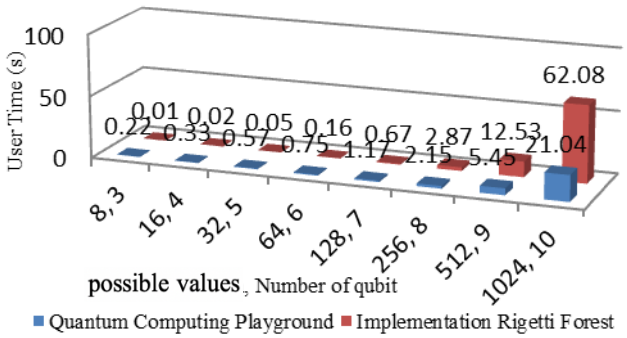

Fig. 2. User time comparison on Computer 2

With 1024 data and 10 qubits, the user time of this implementation is slower than Quantum Computing Playground on both computers. However, the user time difference in Computer 1 is larger than Computer 2. In Computer 1, the user time of QCP's implementation is around five times faster than this implementation. In Computer 2, the user time of QCP's implementation is around three times faster than this implementation.

User Time Comparison Between Computers

The user time of this implementation on two different computers is shown on Fig. 3. For every possible value and qubit, the user time of Computer 2 are faster than the user time of Computer 1. The correlation between user time, processor core, and clock speed in this implementation is a negative correlation. With 1024 data and 10 qubits, the user time of this implementation is 100.81 second on Computer 1 and 62.08 seconds on Computer 2. The user time difference between the two computers is around 62%.

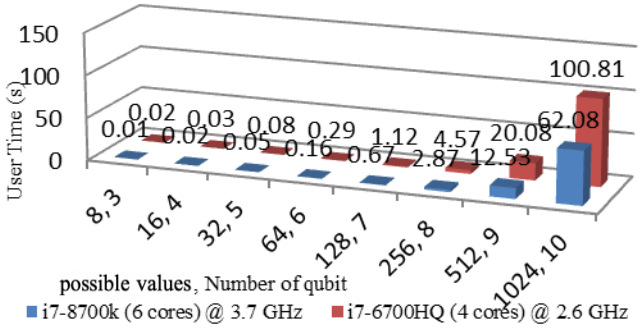

Fig. 3. User time comparison of this implementation

The user time of QCP's implementation on two different computers is shown in Fig. 4. For every possible value and qubit, the user time of Computer 2 are faster than Computer 1. The correlation between user time, processor core, and clock speed in QCP's implementation is a negative correlation. With 1024 data and 10 qubits, the user time of QCP's implementation is 21.93 second on Computer 1 and 21.04 second on Computer 2. The user time difference between the two computers is around 4%.

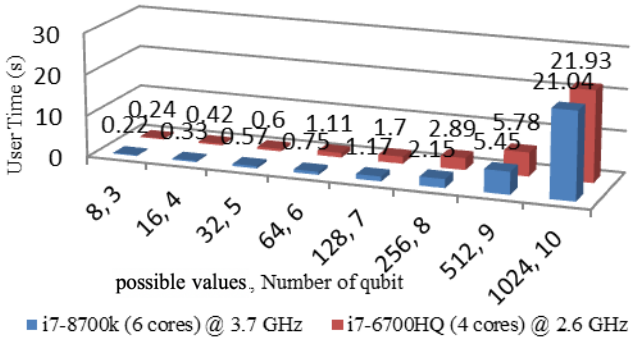

Fig. 4. User time comparison of QCP's implementation

The user time of this implementation on two different computers with various possible values is shown in Fig. 5. When the possible values increases, the user time also increases. The correlation between user time and the possible values in this implementation is a positive correlation. With 1024 data, the user time of this implementation is 100.81 second on Computer 1 and 62.08 second on Computer 2. The user time difference between the two computers is around 62%.

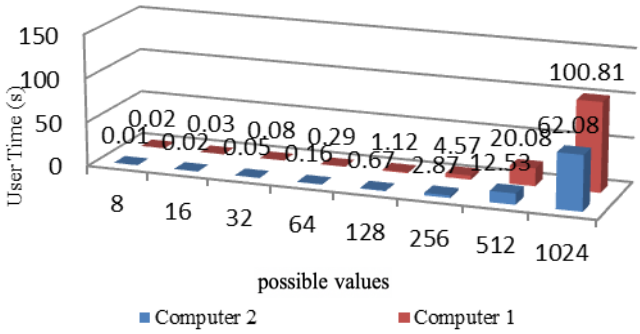

Fig. 5. User time comparison of this implementation with various possible values

The user time of QCP's implementation on two different computers with various possible values is shown in Fig. 6. When the possible values increases, the user time also increases. The correlation between user time and the possible values in QCP's implementation is a positive correlation. With 1024 data, the user time of this implementation is 21.93 second on Computer 1 and 21.04 second on Computer 2. The user time difference between Computer 1 and Computer 2 is around 4%.

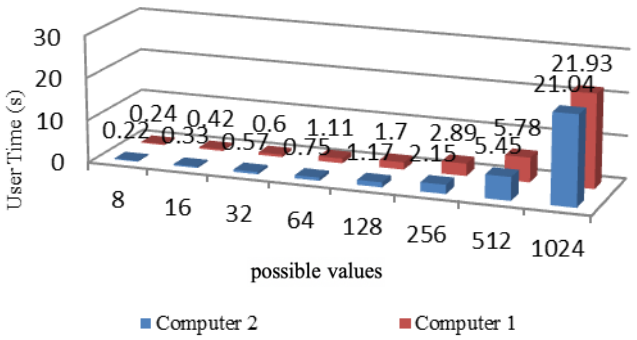

Fig. 6. User time comparison of QCP's implementation with various possible values

IV. Conclusion

In this research, the implementation of Grover's quantum search algorithm could be accomplished using Rigetti Forest and Python. The user time of this implementation is slower than the Quantum Computing Playground with 1024 data and 10 qubits in both Computer 1 and Computer 2. The user time difference with 1024 data and 10 qubits in Computer 1 (around five times difference) is larger than Computer 2 (around three times difference).

The number of processor cores and clock speed has a negative correlation with user time in both implementations. The user time difference with 1024 data and 10 qubits in this implementation (62%) is larger than the Quantum Computing Playground (4%). The possible values has a positive correlation with user time in both implementations. The user time difference with 1024 data in this implementation (62%) is larger than the Quantum Computing Playground (4%).

Further work could be done in improving the implementation accuracy of Grover's quantum search algorithm in finding the correct result. In addition, the implementation of Grover's quantum search algorithm can be done using other simulators with larger number of qubits. It is also an important issue in this research to bring the implementation into production, including the use of real quantum computers.

ACKNOWLEDGMENT

I would like to thank my supervisor Arya Wicaksana for the support, guidance, and advice in this research. I also want to thank Julio Christian Young for his suggestion on the subject and assistance in the laboratory.

REFERENCES

1. M. A. Kinnunen, "Examining the limits of Moore's Law: Possible influence of technological convergence on redefining the curriculum in ICT institutions," Helsinki Metropolia University, 2015.
2. N. Barde, D. Thakur, P. Bardapurkar, and S. Dalvi, "Consequences and Limitations of Conventional Computers and their Solutions through Quantum Computers," *Leonardo Electronic Journal of Practices and Technologies*, vol. 10, no. 19, 2011, pp. 161-171.
3. Y. Kanamori, S. M. Yoo, W. D. Pan, and F. T. Sheldon, "A short survey on quantum computers," I*nternational Journal of Computers and Applications*, vol. 28, no. 3, 2006, pp. 227-233.
4. M. Ying, "Quantum computation, quantum theory and AI," *Artificial Intelligence*, vol. 174, no. 2, 2010, pp. 162-176.
5. M. A. Nielsen and I. L. Chuang, "Quantum Computation and Quantum Information". New York: Cambridge University Press, 2010.
6. A. B. Mutiara, and R. Refianti, "Simulation of Grover's Algorithm Quantum Search in a Classical Computer," *International Journal of Computer Science and Information Security*, vol. 8, no. 9, 2010.
7. Rigetti Computing (2019). Installation and Getting Started [Online]. Available: http://docs.rigetti.com/en/stable/start.html.
8. R. S. Smith, M. J. Curtis, and W. J. Zeng, "A Practical Quantum Instruction Set Architecture," *arXiv preprint*, 2017.
9. G. Wroblewski, and L. Culp (2014). Help [Online]. Available: http://www.quantumplayground.net/#/about.
10. E. Hermann, B. Raffin, F. Faure, T. Gautier, and J. Allard, "Multi-GPU and Multi-CPU Parallelization for Interactive Physics Simulations," *Euro-Par 2010*, 2010, pp. 235-246.
11. A. Wicaksana, and C. Tang, "Virtual Prototyping Platform for Multiprocessor System-on-Chip Hardware/Software Co-design and Co-verification," in *Computer and Information Science*, R. Lee, Ed. Cham: Springer, 2017, pp. 93-108.
12. A. Wicaksana, "Multiprocessor System-on-Chip (MPSoC) virtual platform for hardware and software co-design and co-verification," Universiti Tunku Abdul Rahman, 2019.
13. L. K. Grover, "A fast quantum mechanical algorithm for database search," *Symposium on Theory of Computing*, 1996, pp. 212-219.
14. IBM Research and the IBM QX team (2017). Grover's Algorithm [Online]. Available: https://quantumexperience.ng.bluemix.net/proxy/tutorial/full-user-guide/004-Quantum_Algorithms/070-Grover's_Algorithm.html.
15. J. Hui (2018). QC-Grover's Algorithm [Online]. Available: https://medium.com/@jonathan_hui/qc-grovers-algorithm-cd81e61cf248.
16. M. Boyer, G. Brassard, P. Høyer, and A. Tapp, "Tight bounds on quantum searching," *PhysComp96*, 1996, pp. 1-8.

Authors Profile

Anthony, undergraduate student in Informatics at Universitas Multimedia Nusantara.

Arya Wicaksana, graduated from Universiti Tunku Abdul Rahman in VLSI Engineering (M.Eng.Sc.) and Universitas Multimedia Nusantara in Computer Science (S.Kom.). Research interests and works are: computational intelligence and quantum computing. E-mail: arya.wicaksana@umn.ac.id.