Science Bulletin 63 (2018) 964–971

Contents lists available at ScienceDirect

Science Bulletin

journal homepage: www.elsevier.com/locate/scib

Article

64-qubit quantum circuit simulation

Zhao-Yun Chen [a,b,1], Qi Zhou [a,b,1], Cheng Xue [a,b], Xia Yang [b], Guang-Can Guo [a], Guo-Ping Guo [a,*]

[a] CAS Key Laboratory of Quantum Information, University of Science and Technology of China, Hefei 230026, China
[b] Origin Quantum Computing Company Limited, Hefei 230026, China

ARTICLE INFO

Article history:
Received 22 March 2018
Received in revised form 11 May 2018
Accepted 8 June 2018
Available online 20 June 2018

Keywords:
Simulation of quantum circuits
Universal random circuit
Quantum supremacy
Partitioning
Parallel computing

ABSTRACT

Classical simulations of quantum circuits are limited in both space and time when the qubit count is above 50, the realm where quantum supremacy reigns. However, recently, for the low depth circuit with more than 50 qubits, there are several methods of simulation proposed by teams at Google and IBM. Here, we present a scheme of simulation which can extract a large amount of measurement outcomes within a short time, achieving a 64-qubit simulation of a universal random circuit of depth 22 using a 128-node cluster, and 56- and 42-qubit circuits on a single PC. We also estimate that a 72-qubit circuit of depth 23 can be simulated in about 16 h on a supercomputer identical to that used by the IBM team. Moreover, the simulation processes are exceedingly separable, hence parallelizable, involving just a few inter-process communications. Our work enables simulating more qubits with less hardware burden and provides a new perspective for classical simulations.

1. Introduction

The last few years have seen a series of significant advances in quantum computing, in particular regarding superconducting quantum chips with reports of devices of 20 and 50 qubits with good fidelity [1,2]. In the meantime, great progress has also been made with semiconductor quantum chips [3–5]. "Quantum supremacy" claims that the limit of classical computers would be transcended if a device of 50 qubits were made [6]. Direct simulations of 50 qubits take about 16-PB of RAM to store the full vectors. Google and IBM teams have proposed some efficient methods for simulating the low-depth circuit with more than 49 qubits (e.g., deferral of entanglement gates [7] and Feynman path method [8]). Here, we present a scheme to optimize the classical simulation of quantum circuits with low depth and large sampling number, with which we have performed a 64-qubit simulation with sampling number 2^{28} and depth 22. In particular, by transforming several control-Z (CZ) gates to measurement and single-qubit gates, the circuit is mapped onto an additional 2^n sub-circuits. These sub-circuits are formed by two blocks without any qubit entanglement between them, thereby converting an N qubit simulation problem into a group of $N/2$. Our method is similar to a small balanced cut in a two-dimensional grid [9], while the method developed by Aaronson is more general but more complicated as a compromise. For decomposing one CZ gate, their method splits the original circuit into eight sub-circuits, while in our case it splits into four. The results of all the sub-circuits are then added together to reconstruct the final state. In practice, we simulated the universal random circuit, which is used to characterize the quantum supremacy in the region of quantum chaos [10–17].

2. Methods

2.1. Partition scheme

A CZ gate can be transformed into two groups of measurement and single-qubit gates, specifically

$$\mathrm{CZ} = P_0 \times I + P_1 \times Z, \tag{1}$$

where $P_0 = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$, $P_1 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$. I denotes the unit matrix, and Z the Pauli-Z matrix. The transformation dismisses the entanglement gate between the two qubits and makes a copy to the circuit. We illustrate an 8-qubit circuit of depth 8 as an example (Fig. 1). By transforming CZ gates in the 7th and 8th layer (in the dashed boxes), the original circuit is converted to four copies. The final state is the addition of the four final states of each copy. In the four copies, qubits 0–3 are no longer entangled with qubits 4–7. Hence we can simulate them separately. At the beginning, 8 qubits can represent 2^8 states, and after the conversion, there are 8 circuits

* Corresponding author.
E-mail address: gpguo@ustc.edu.cn (G.-P. Guo).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.scib.2018.06.007
2095-9273/

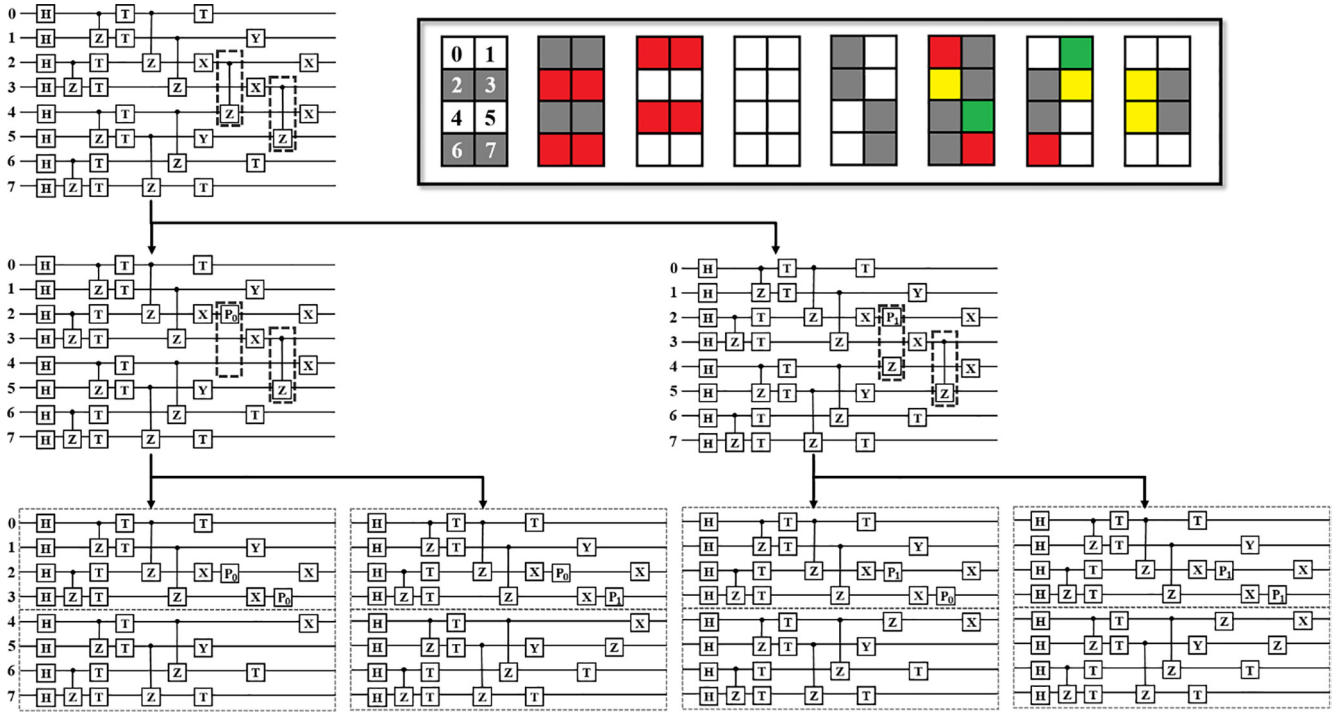

Fig. 1. An example for partitioning. The first row is the original circuit. The two CZ gates in the dashed boxes entangle the first and last four qubits. Next, the left CZ gate is transformed, the original circuit being equivalent to the addition of the circuits in the second row. Continuing, the right CZ gate is transformed, generating four circuits in the third row. The final state of the original circuit is equal to the addition of all transformed circuits. The dashed boxes in the third line divide each circuit into two parts, where they can be simulated independently. Inset: The 4×2 quantum circuit of depth 8 expressed in the form of a grid, where the coloured squares represent different gates: yellow–an X gate, green–a Y gate, red–a T gate, and grey–a CZ gate. Each graph represents a layer. The numbers denote the corresponding qubit in this example.

each with 4 qubits representing 8×2^4 states (see the grey dashed boxes in the bottom of Fig. 1); the space is reduced. Initially, there are 27 gates of 8 qubits, and after the conversion, there are 112 gates of 4 qubits — the time is therefore reduced.

In practice, we divided each of the 4 qubit into two half-circuits — an upper and a lower part as shown in the bottom of Fig. 1. In the first six layers, there is no CZ gate entangling the two half-circuits. In the 7th and 8th layers, the CZ gates entangling the two half-circuits should be transformed as mentioned above. The same holds for the transformation executed on the 15th and 16th layers. This generates 2^c circuits to be simulated, where c is the number of transformed CZ gates.

2.2. Methods to estimate the different qubit counts and depths

We provide in Table 1 a set of time estimates for various qubit topologies and circuit depths. The time estimation are obtained from

$$\text{Time} = \sum_{i=1}^{d(\text{depth})} n_i(\text{gates}) \times m(\text{circuits}) \times t(\text{time / gate})/s(\text{nodes}), \quad (2)$$

where n_i is the number of efficient gates in the i_{th} layer of each half circuit, d the depth, m corresponds to the number of the equivalent half-circuits, t is the average time per gate, and s is the number of nodes (or parallel units if more than one node are packed into a unit).

Under the rules described in Ref. [1], we estimate the number of X and Y gates in each layer of each half circuit (expect for the first three layers) to be 6, 8 and 10 for 56-, 64- and 72-qubit circuit, respectively. As we optimize the simulation of diagonal gates, all of the CZ gates and T gates in each half circuit can be combined into 2 gates. Therefore, for all scales of the circuits, we make $n_1 = 1$, $n_2 = 2$ and $n_3 = 2$ while for $i > 3$ we let n_i to be 8, 10 and 12 for the 56-, 64- and 72-qubit half circuit, respectively.

Table 1
Time estimations for different qubit counts and depths for the universal random circuit. The average time per gate of the 28-qubit circuit is estimated to be 0.25 s, and that of the 32-qubit and 36-qubit circuits (obtained from Ref. [18]) are 0.38 and 0.67 s, respectively.

Qubit	Depth	Time estimation
56	22	54.8 s
	23	7.68 min
	30	2.74 h
	31	22.7 h
	38	18.9 d
	39	155 d
64	22	6.90 min
	23	1.93 h
	30	1.73 d
	31	28.7 d
72	22	58 min
	23	16 h

2.3. Simulation schemes on different hardware circumstances

When the qubit count (simulated directly) increases from 28 to 32, and further to 36, different simulation strategies should be applied to adapt to the huge difference in the amount of storage required. Here, we propose three possible hardware configurations