Quantum Supremacy Circuit Simulation on Sunway TaihuLight

1st Riling Li
Department of Computer Science & Technology
Tsinghua University
Beijing, China
rl-li16@mails.tsinghua.edu.cn

2nd Bujiao Wu
CAS Key Lab of Network Data Science and Technology
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
wubujiao@ict.ac.cn

3rd Mingsheng Ying
Centre for Quantum Software and Information, University of Technology Sydney
Sydney, Australia
State Key Lab of Computer Science, Institute of Software, Chinese Academy of Sciences
Department of Computer Science & Technology, Tsinghua University
Beijing, China
Mingsheng.Ying@uts.edu.au

4th Xiaoming Sun
CAS Key Lab of Network Data Science and Technology
Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
sunxiaoming@ict.ac.cn

5th Guangwen Yang
Department of Computer Science & Technology
Tsinghua University
Beijing, China
ygw@tsinghua.edu.cn

***Abstract*— With the rapid progress made by industry and academia, quantum computers with dozens of qubits or even larger size are coming true. However, the fidelity of existing quantum computers often sharply decreases as the circuit depth increases. Thus, an ideal quantum circuit simulator on classical computers, especially on high-performance computers, is needed for benchmarking and validation. We design a novel simulator of universal random quantum circuits, often called "quantum supremacy circuits", and implement it on Sunway TaihuLight. The simulator can be used to accomplish the following two tasks: 1) Computing a complete output state-vector; 2) Calculating one or a few amplitudes. We target the simulation of 49-qubit circuits. For task 1), we successfully simulate such a circuit of depth 39, and for task 2) we reach the 55-depth level. To the best of our knowledge, both of the results are the state-of-the-art, which in return raises the bar of "quantum supremacy".**

***Index Terms*—quantum computing, quantum circuit simulation, Sunway TaihuLight, quantum supremacy**

I. Introduction

The concept of quantum computer was proposed almost four decades ago [7]. But for a long time people are not sure whether a quantum computer can indeed exceed the computing capability of its classical predecessors. Thanks to the process made by industry and academia in recent years, practical quantum computing might become reality soon. However, before a commercial quantum computer is launched on market, many tests and verification need to be done. One of the most important is to test the fidelity of quantum circuit. One way to accomplish this is to simulate quantum circuits by computing the ideal state amplitudes on a classical computer. A quantum simulator on classical computer could also be used to verify correctness of certain quantum algorithms and benchmark the notion "quantum supremacy".

Quite a few implementations of quantum circuit simulators have been developed in last few years [12, 15, 14, 9, 11, 3]. To test the limitation of quantum circuit simulation on classical computers, we design a simulator on Sunway TaihuLight, one of the most powerful high-performance computer at present in the world. We mainly aim at 49-qubit circuits, because such circuits are hard to directly simulate on other supercomputers due to the limited memory spaces, and it is widely believed that near-term quantum device will first achieve quantum supremacy at this scale. Our simulator can accomplish the following:

- ***Task 1***: Computing a complete output state-vector representing a quantum state;
- ***Task 2***: Sampling (i.e. calculating a small amount of) the amplitudes of a quantum state.

For Task 1, we are able to solve the lattice of 7×7 qubits with depth 39 [1] in several hours. As to Task 2, our method can calculate one amplitude for 49-qubit circuits of depth 55.

[1]The first layer of Hardmard gates are not counted as the circuit depth, we treat it as layer 0. So the circuit we simulate is from layer 0 to layer 39. This also holds for Task 2.

Moreover, our method for Task 1 can also be directly extended to the lattices of 7×8, 8×8, and 9×8 qubits, calculating a few amplitudes, though the reachable depth of the random circuit would be decreasing with the increasing of the qubits.

A. Comparison with other quantum simulators

For Task 1, the simulator described in [9] can simulate a quantum circuit with 5×9 qubits and depth 25 on the Cori II supercomputer in less than 10 minutes. Reference [11] reported the simulation of a 7×7 grid of qubits with depth 27, which costs more than one day on IBM Blue Gene/Q . In contrast, our method simulates a 7×7 grid of qubits with depth 39 in even less time. The task of simulating 7×8 grid of qubits with depth 23 was not finished in [11] because 2^{56} amplitudes are too many to calculate. For this task, they only calculated 2^{39} amplitudes. In contrast, our simulator can calculate $2^{37} \sim 2^{42}$ amplitudes in a short time for 7×8-qubit circuits of depth 35.

Task 2 of sampling amplitudes can be used to estimate the fidelity by computing the cross entropy, which usually needs $10^3 \sim 10^6$ samples [4]. The sampling target is to calculate an amplitude α_x, where $0 \leq x \leq 2^n - 1$ for an n-qubit circuit. The method of calculating the state-vector can also be directly applied to calculate a small number of amplitudes, though in this way the reachable depth is not very enough. However, we can get one amplitude for 7×7-qubit circuits of depth 55 by calculating the inner product of two 49-qubit state-vector, although this is very expensive. Our results of Task 1 and Task 2 shows that the bound of 49-qubit with depth 40 in [3] is no more out of reach. Thus, in a sense, our simulator raises the bar of "quantum supremacy".

B. Technical contributions of this paper

The first contribution of this paper is that we introduce a different partition scheme via a dynamic processing algorithm, by which we can save more time and space than [11]. At the same time, we propose several *global* optimizing techniques, including some new optimizations adaptive to the structure of 2D grid circuits and several other optimizations for reducing the network communication amount when implementing our method.

Our second contribution is some *local* optimizing techniques on a single node, which take the advantage of the many-core heterogeneous processor of Sunway. Our optimization greatly reduces the amount of memory access while it only increases a small quantity of calculation. We also apply some standard optimization such as vectorization. By all of these techniques, we can simulate a 28-qubit circuit on one core group very quickly, which is significant for improving the performance of 49-qubit circuit simulation.

Sunway TaihuLight adopts a popular many-core heterogeneous system architecture, of which in one CPU chip there are 4 core groups (CGs) each with 1 management processing elements (MPEs) and 64 computing processing elements (CPEs) [8]. Since in our implementation each core group corresponds to an unique MPI process, we regard a core group instead of a CPU chip as a single node in this paper to avoid some messy and confusing description. That is, *each node contains only 1 master core and 64 slave cores and corresponds to one MPI process.*

C. Organisation of the paper

Section 2 describe the simulation methodologies and optimization techniques. Section 3 presents the numerical results of our implementation as well as verification of the results. Section 4 draws a conclusion and discusses the problems to be solved in the future work.

II. Methodologies and optimizations

Before describing our methods and optimizations, let us recall some basic notions and notations. A basic storage unit in a quantum computer is a quantum bit (qubit). Generally, we can use a 2^n-length complex vector to describe an n-qubit state, such as $|\psi\rangle = (\alpha_0, \alpha_1, ...\alpha_{2^n-1})^T$. A practical quantum circuit consists of single-qubit and 2-qubit gates. A quantum circuit performed on an n-qubit state can be regarded as a $2^n \times 2^n$-dimensional matrix multiplying the 2^n-dimensional complex vector. Without doubt, there is no need to really calculate and store the matrix in the simulation. One can perform the quantum gates one by one on the quantum state, making the calculation point-wise. For example, a single-qubit gate $U^k = \begin{pmatrix} a & b \\ c & d \end{pmatrix}$ on qubit k can be treated as an n-qubit gate which acts as identity on the other $n-1$ qubits, as denoted by $U = I^{\otimes n-k-1} \otimes U^k \otimes I^{\otimes k}$, where I is the identity operator on a single qubit. In this paper, a superscript indicates the qubit that the gate is performed on, and a subscript is used as a gate label or an index of amplitude. To perform U^k on a state $|\psi\rangle = (\alpha_0, \alpha_1, \cdots, \alpha_{2^n-1})^T$, we can pair $(\alpha_i, \alpha_{i+2^k})$ together and then apply U^k to each pair, where $i = i_{n-1}...i_1 i_0$ is the binary representation of i and $i_k = 0$. After the calculation, we get a new pair $(\alpha'_i, \alpha'_{i+2^k})^T = U^k(\alpha, \alpha_{i+2^k})^T$. The result is then $|\psi'\rangle = (\alpha'_0, \alpha'_1, ...\alpha'_{2^n-1})^T = U^k|\psi\rangle$. The 2-qubit gates used in this paper are mainly the two controlled-gate: CNOT and CZ. A controlled-gate $CU^{c,t}$ act on qubits c and t, with the first being the control bit and the second being the target bit. The performance of a 2-qubit gate is similar to that of a single-qubit gate with the only extra consideration of whether the control qubit is in state $|1\rangle$; for more details, we refer to [10].

For a quantum circuit simulation, the initial state is usually the product state:

$$|0\rangle^{\otimes n} = \underbrace{|0\rangle \otimes |0\rangle \otimes ... \otimes |0\rangle}_{n}.$$

If there is no 2-qubit gate in the circuit, the state will persistently remain a product state and only $O(n)$ space is needed to describe the state. But as the number of two-qubit gates increases, the quantum state may become highly entangled. In this case, the storage of the state will require $O(2^n)$ space. However, for a 2-qubit gate $CU^{c,t}$, we do not need to directly calculate the result like the single-qubit case. Instead we can

decompose it to $CU^{c,t} = P_0^c \otimes I^t + P_1^c \otimes U^t$, where $P_0 = |0\rangle\langle 0|$ and $P_1 = |1\rangle\langle 1|$, and we separate the quantum state $|\psi\rangle$ into two parts: $|\psi_{i_c=0}\rangle = (\alpha_0, \alpha_1, ..., \alpha_{2^n-1})^T$ and $|\psi_{i_c=1}\rangle = (\beta_0, \beta_1, ..., \beta_{2^n-1})^T$. For $i = i_{n-1}...i_1 i_0$, if $i_c = 0$ then $\beta_i = 0$ else $\alpha_i = 0$, and apparently $|\psi\rangle = |\psi_{i_c=0}\rangle + |\psi_{i_c=1}\rangle$. Let $|\psi'_{i_c=0}\rangle = I^t|\psi_{i_c=0}\rangle$ and $|\psi'_{i_c=1}\rangle = U^t|\psi_{i_c=1}\rangle$. From now on, the reminding simulation has two branches: one starts from $|\psi'_{i_c=0}\rangle$, and the other starts from $|\psi'_{i_c=1}\rangle$. We can immediately merge $|\psi'_{i_c=0}\rangle$ and $|\psi'_{i_c=1}\rangle$ into $|\psi'\rangle$, and it is exactly the same as the result of directly performing $CU^{c,t}$ on $|\psi\rangle$. But we can also do the merging later, and from the two paths it will generate more branches as some other subsequent 2-qubit gates are decomposed. This idea make it possible that the whole circuit can be partitioned into two subcircuits with the entanglement between them deferred. Until an appropriate stage we can combine these branching paths into a complete output state, because the memory cost and run-time increase exponentially in the number of 2-qubit gates decomposed. Essentially, this idea comes from the notion of "Feynman" path integral and a similar idea appeared in [2, 11, 3]. Figure 1 gives a simple example showing how circuit partition works.

A universal random quantum circuit has a 2D grid architecture [4]. The quantum gates used in this type of circuits are $H, X^{1/2}, Y^{1/2}, T$ and CZ. The 2-qubit gate CZ only appears between two adjacent qubits in the grid. Since in each layer of an $n \times m$-grid universal random circuit the positions of CZ gates are fixed, we can find a concrete partition scheme according to the scale of circuit to be simulated. Our techniques for finding this partition scheme are described in detail in the following subsection.

A. Method for computing the complete state-vector (Task 1)

Figure 1 describes the basic ideas of circuit partition. But as the circuit depth increases, the number of decomposed 2-qubit gates also increases. To increase the depth of circuits that can be simulated, we propose two optimizing techniques, which enable us to simulate 49-qubit circuits of depth 39, computing the complete output state-vector:

- ***Technique 1***: We analyze the structure of universal random circuits, exploit the diagonal properties of CZ gates, and propose a technique, called *implicit decomposition*, which can decompose extra 7 CZ gates without requiring too much extra memory space, so as to increasing 8 depths for the circuits simulated.
- ***Technique 2***: We propose a dynamic programming (DP) algorithm to find a good partition scheme for a given simulation task. In contrast to the general heuristic search method that usually takes a long time, this DP algorithm is efficient and can find an optimal partition scheme under certain constrains. It also makes the simulation easier to optimize and parallelize and thus improves the performance in time-to-solution.

1) Implicit decomposition: For a better understanding of Technique 1, let us first consider a circuit example shown in Fig. 2. The partition scheme for simulation is illustrated by the blue dotted line and yellow dotted line. The dotted lines cut two CZ gates, and partition the circuit into 3 parts: A, B, C. This procedure is similar to [11]. We use $|\phi\rangle$ and $|\xi\rangle$ to describe the initial states in the two subsystems. Because there are two CZ gates being decomposed (cut by dotted lines), four branching paths in total are generated. Let $|\phi^{out}_{l_1,l_2}\rangle$ be the resulting state after performing the gates in part A, l_1 and l_2 denote the indices of qubit 2 in two different time. State $|\xi^{out}_{l_1,l_2}\rangle$ is similar to $|\phi^{out}_{l_1,l_2}\rangle$. Then we have[2]:

$$|\phi^{out}_{l_1,l_2}\rangle = X^0 CZ^{0,1} Y^1 CZ^{2,1}_{l_2} T^1 CZ^{2,1}_{l_1} H^0 H^1 |\phi\rangle$$

where l_1 and l_2 denote the indices of qubit 2 when decomposing the cutting CZ gates, thus $CZ^{2,1}_0 = I^1$ and $CZ^{2,1}_1 = Z^1$. Furthermore, we have:

$$|\xi^{out}_{l_1,l_2}\rangle = X^3 P^2_{l_2} T^2 CZ^{2,3} P^2_{l_1} H^3 H^4 |\xi\rangle$$

where P is a projection operator: $P_i|i\rangle = |i\rangle$ and $P_i|1-i\rangle = 0$ for $i = 0, 1$. The starting state of part C is:

$$|\psi^{in}\rangle = \sum_{l_1,l_2} |\phi^{out}_{l_1,l_2}\rangle \otimes |\xi^{out}_{l_1,l_2}\rangle \tag{1}$$

We perform the remaining gates in part C, and the eventually state $|\psi^{out}\rangle$ is:

$$|\psi^{out}\rangle = Y^1 X^2 CZ^{1,2} X^2 |\psi^{in}\rangle \tag{2}$$

There are $2^4 = 16$ amplitudes to calculate for $|\psi^{in}\rangle$ (or $|\psi^{out}\rangle$). But we do not need memory space for 16 amplitudes(not counting memory space for $|\phi\rangle$ and $|\xi\rangle$) to accomplish the calculation. Note that in part C there are gates only performed on qubit 1 and qubit 2, and no gate on qubit 0 and qubit 3. So $|\psi^{in}\rangle$ can be divided into 4 blocks by enumerating the indices of qubit 0 and qubit 3. Let $|\psi^{q_1,q_2}_{i_0,i_3}\rangle$ denote the reduced state of qubit 1 and qubit 2 with qubit 0 at $|i_0\rangle$ and qubit 3 at $|i_3\rangle$. Then $|\psi^{in}\rangle = \bigoplus_{i_0,i_3} |\psi^{q_1,q_2}_{i_0,i_3}\rangle$. Equation (2) turns into:

$$\begin{aligned} |\psi^{out}\rangle &= \bigoplus_{i_0,i_3} |\psi^{out,q_1,q_2}_{i_0,i_3}\rangle \\ &= \bigoplus_{i_0,i_3} (Y^1 X^2 CZ^{1,2} X^2 |\psi^{q_1,q_2}_{i_0,i_3}\rangle) \end{aligned} \tag{3}$$

Now we know that if all the results of part A and part B are calculated and stored in memory, there only needs extra space for $2^2 = 4$ amplitudes in part C. From now on we set an amplitude as a basic storage unit ($8 + 8 = 16bytes$), and the total space consumption is $S_A + S_B + 4$ instead of $S_A + S_B + 16$, where S_A and S_B are the space consumption of part A and part B, respectively.

From the above analysis, we know $S_A = S_B = 2^{2+2} = 16$. Nevertheless, S_B can actually halve. Note that after the second cut CZ gate being decomposed, there is no other gate in part B performed on qubit 2, so for any amplitude of $|\xi^{out}_{l_1,l_2}\rangle$, let it be ξ_{i_2,i_3,l_1,l_2}, where i_2 and i_3 are the indices of qubit 2 and qubit 3. We have

$$\xi_{i_2,i_3,l_1,l_2 \neq i_2} = 0, \text{ for } l_2 = 0, 1$$

[2]The order of performing gates in circuit is from left to right, while the matrix-vector multiplication is from right to left.

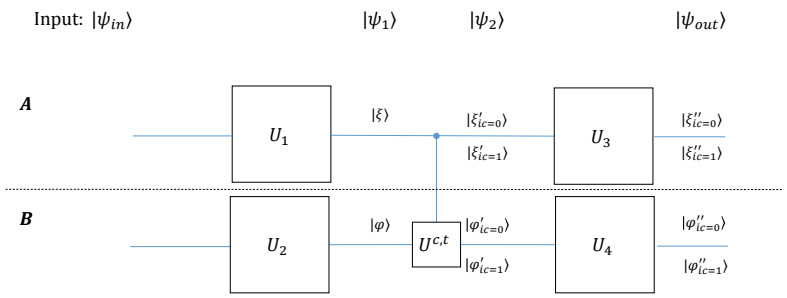

Fig. 1. An illustrative example of how gate decomposition works. The initial state is $|\psi_{in}\rangle = |0\rangle^{\otimes n}$ and the circuit is $\mathcal{U}_{circuit} = (U_3 \otimes U_4)CU^{c,t}(U_1 \otimes U_2)$. The whole system can be regarded as a composition of two subsytems **A** and **B**, while the gate $CU^{c,t}$ (qubit c in **A** and qubit t in **B**) causes their entanglement. Assume **A** has n_A qubits and **B** has n_B qubits, and the total qubit number is $n = n_A + n_B$. After performing U_1 and U_2, the whole state is $|\psi_1\rangle = |\xi\rangle \otimes |\varphi\rangle$. With the decompsing of $CU^{c,t}$ we get two branches: qubit c at $|0\rangle$ and qubit c at $|1\rangle$. After performing $CU^{c,t}$, we have $|\xi^{'}_{i_c}\rangle$ and $|\varphi^{'}_{i_c}\rangle$, $i_c = 0, 1$. Finally, we perform U_3, U_4 and get $|\xi^{''}_{i_c}\rangle = U_3|\xi^{'}_{i_c}\rangle$, $|\varphi^{''}_{i_c}\rangle = U_4|\varphi^{'}_{i_c}\rangle$. The result is $|\psi_{out}\rangle = \sum_{i_c=0,1} |\xi^{''}_{i_c}\rangle \otimes |\xi^{''}_{i_c}\rangle$. Note that after performing $CU^{c,t}$ we need to double the space to storage the two branches (qubit c at $|0\rangle$ or $|1\rangle$). So, the total space consumption is $2^{n_A+1} + 2^{n_B+1}$, in contrast to 2^n, the space consumption of directly calculating the resulting state of the composite system gate by gate.

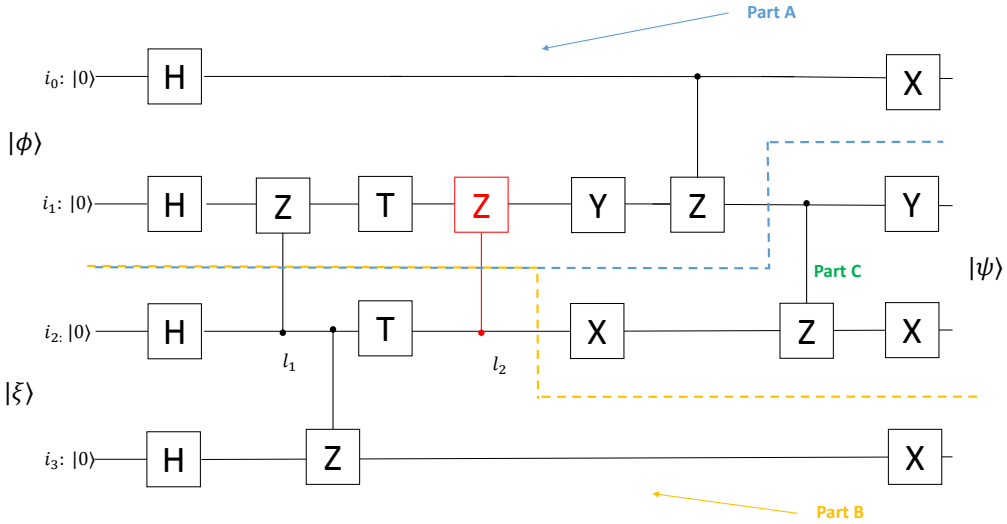

Fig. 2. Example of a 4-qubit circuit. In this example, there are 2 CZ gates being decomposed. Note that the second decomposed CZ gate only doubles the space consumption of part A, because after this CZ gate there is no gate in part B performed on qubit 2, thus the implicit decomposition.

Thus, the index l_2 could be absorbed into index i_2, and we only need to store $|\xi^{out}_{l_1}\rangle$. This means that when we finish the calculation of part B, only $2^{1+2} = 8$ amplitudes to be stored, while in part A there are still $2^{2+2} = 16$ amplitudes to be stored. Because the second cut CZ gate is decomposed but the space consumption need not be doubled for control part (part B in this example), we call this technique *implicit decomposition*.

The implicit decomposition is useful when the space consumption of part A and part B is not balanced, say there need space S_A to store part A and some fewer space S_B to store part B, we could apply implicit decomposition to part A, and only making the space of part A to S'_A while leaving the space of B unchanged, until S'_A and S_B are almost in the same magnitude.

For a universal random quantum circuit of $m \times n$ grid, the implicit decomposition works well when m or n is odd. Our main target is 7×7-qubit circuits. At first, the two cutting lines overlap and partition the circuit into two parts: part A with 21 qubits, part B with 28 qubits. Then we apply this technique to part B, as shown in Fig. 3.

2) Dynamic programming: To reduce the total space consumption and make the simulator easier to optimize, we avoid decomposing CZ-gates between part C and parts A, B. At first

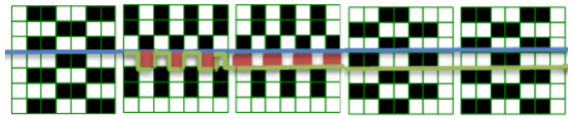

Fig. 3. Implicit decomposition applied to 49-qubit universal random circuits. In this and following figures, two adjacent blocks represent a CZ gate. And the blocks in red represent the CZ gate that could be implicit decomposed. Because part A has 21 qubits and part B has 28 qubits, without implicit decomposition $S_B = 2^{28+7}$,$S_A = 2^{21+7}$. After we use implicit decomposition on these seven cut CZ gates at the second and third layer, the ultimate space consumption of part B is $s'_B = S_B/2^7$, and now $S_A = S'_B$.

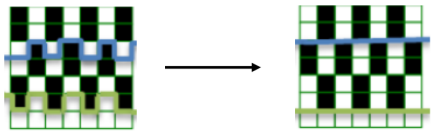

Fig. 4. A illustrative example of legal transition between two adjacent layers. From layer t to $t+1$, the position of upper splitting line (in blue) transforms from $(0,1,0,1,0,1,0)$ to $(1,1,...,1)$. Since $(1,1,...,1)$ does not cut any CZ gate at layer $t+1$, this is a legal transition. So $f(t+1,1,1,...,1) = min\{f(t+1,1,1,...,1), f(t,0,1,0,1,0,1,0)\}$.

two splitting line overlap. When the two splitting lines separate, they start to walk around the CZ gates and will not cut-off any one of them. A feasible partition scheme also requires the total space consumption less than the memory space. To find an optimal partition scheme under these constrains, we design a dynamic algorithm for state compression.

Let $f(t, i_1, i_2, i_3, i_4, i_5, i_6, i_7)$ denote the minimal space consumption (exponential in f) of part A when the partition scheme reaches layer t and the position of upper splitting line is $(i_1, i_2, ..., i_7)$. Here, $(i_1, i_2, ..., i_7)$ is used to indicate the distance between the current position and the initial position. Initially, the upper splitting line is beneath the third row of qubits, and $f(1, 0, ..., 0) = 21$. Note that $f(7, 0, 0, ..., 0) = 21 + 3 = 24$ and $f(8, 0, 0, ..., 0) = 21 + 3 + 4 = 28$. Since we have a restriction that no CZ gate between part A and part B could be decomposed, $f(t, i_1, i_2, ..., i_7)$ will be *illegal* when $(i_1, i_2, ..., i_7) \neq (0, 0, ..., 0)$ and $(i_1, i_2, ..., i_7)$ splits some CZ gate at layer t.

Similarly, let $g(t, i_1, ..., i_7)$ denote the target function of part B. Then $g(1, 0, 0, ..., 0) = 28$ and $g(8, 0, 0, ..., 0) = 35$. The recursion from $g(t-1)$ to $g(t)$ is similar to f, and the only difference is that when $(i_1, i_2, ..., i_7)$ first leaves the initial position it has a chance of applying implicit decomposition.

To find a good partition scheme for circuit of depth t, we can traverse $f(t)$ and $g(t)$ to find a optimal combination of $f(t, i_1, i_2, ..., i_7)$ and $g(t, j_1, j_2, ..., j_7)$ which minimizes $S_A + S_B + S_C$, where $S_A = 2^{f(t,i_1,i_2,...,i_7)}$, $S_B = 2^{g(t,j_1,j_2,...,j_7)}$ and $S_C = 2^{\sum_{1 \leq k \leq 7}(i_k + j_k)}$. Several partition schemes are illustrated in Fig. 5 and Fig. 8. There need space $S_A = S_B = 2^{35}$, $S_C = 2^{28}$ to execute the simulation for depth 27, while $S_A = S_B = 2^{42}$ for depth 35 and 39. The maximal number of qubits that could be simulated on one node is 28, this indicates that from depth 35 to depth 39, there will be a drop

Algorithm 1: Pseudocode of the Dynamic Processing

Input: $f(t)$
Output: $f(t+1)$
for $0 \leq i_1, i_2, ..., i_7 \leq 3$ **do**
  $f(t+1, i_1, i_2, ..., i_7) = \infty$;
  if *position* $(i_1, i_2, ..., i_7)$ *is not illegal* **then**
    continue;
  end
  $cutnum =$ *number of CZ gates splitting by* $(i_1, ..., i_7)$;
  for $0 \leq j_1 \leq i_1, 0 \leq j_2 \leq i_2, ..., 0 \leq j_7 \leq i_7$ **do**
    $f(t+1, i_1, i_2, ..., i_7) = min\{f(t, j_1, j_2, ..., j_7) + cutnum, f(t+1, i_1, i_2, ..., i_7)\}$;
  end
end

Depth	27	35	39
S_A	2^{35}	2^{42}	2^{42}
S_B	2^{35}	2^{42}	2^{42}
S_C	2^{28}	2^{28}	2^{42}

TABLE I
THE SPACE CONSUMPTION OF PARTS A,B,C FOR 49-QUBIT CIRCUIT OF DIFFERENT DEPTH.

in performance.

3) Summary: The two techniques we proposed above not only work for universal random circuit. For circuit of an arbitrary 2-D grid structure, our method can find a proper partition scheme to reduce the time and space complexity of simulation.

Obviously, 7×7-qubit circuits of depth 39 and of depth 40 have different difficulties in physical preparation and operation. We provide an evidence showing that our method might reach depth 40 if the target circuit is a "lucky circuit".

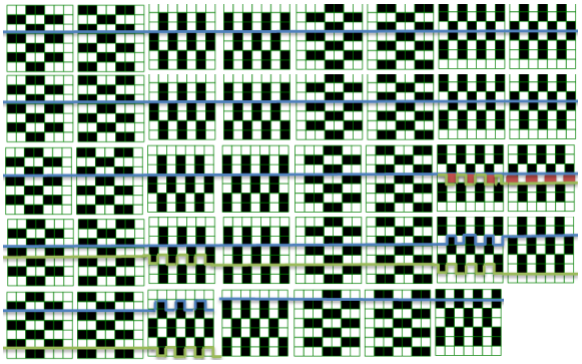

Fig. 5. Partition scheme for 49-qubit circuit of depth 39. The scheme for depth 35 is exactly the front 35 layers of this figure, except for that at layer 35 two splitting lines are still straight as in layer 34 and cut 6 extra CZ gates (See the last layer of partition scheme for 49-circuit of depth 27 in Fig. 8). Note that the CZ gates at the last layer would not impact the probabilities and could be removed because they diagonal. Thus the number of cut CZ gates being cut (the implicit decomposed CZ gates and CZ gates at the last layer not included) is 14 for both depth 35 and 39, but in the case of depth 35 $S_C = 2^{28}$ and in the case of depth 39 $S_C = 2^{42}$.

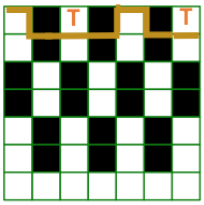

Fig. 6. Tensor slicing technique in [11]. This is a example that two T gates appears in the four positions, thus 5 qubits in total could be sliced.

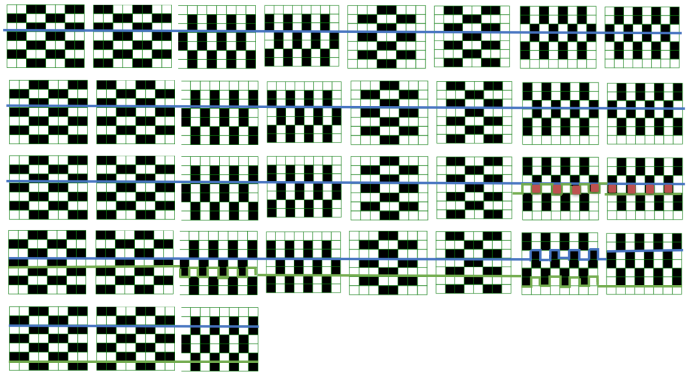

Fig. 7. Partition scheme for 56-qubit circuits of depth 35. The method for solving a part of amplitudes from this circuit is exactly the same as computing a complete state-vector for 49-qubit circuits. Note that here $S_A = S_B = 2^{48}$, which already exceed the memory limit of Sunway. Thus a little space-time tradeoff [2] is needed.

For example, using the tensor slice technique in [11], if the there is at least one T gate in the four single-qubit gates on qubit $0, 2, 4, 6$ at layer 40, the size of a slice is at most 2^{45}, which could be directly simulated on Sunway, though the performance will further drop from depth 39 to depth 40. Because $X^{1/2}, Y^{1/2}, T$ appears randomly at the positions for single-qubit gates, the probability of a 49-qubit, a 40-depth universal random circuit that could be simulated on Sunway is:

$$p = 1 - (2/3)^4 = 65/81$$

B. Calculating one or a few amplitudes (Task 2)

The method in Section II-A can be used to compute the complete state-vector for 49-qubit circuits. For circuits of 56 qubits or larger size, it is difficult to calculate all the amplitudes due to limited time and space. However, to test the fidelity of a real quantum circuit, one only needs to sample (i.e. calculate a small number of) amplitudes, usually ranging from 10^3 to 10^6[4]. Our method can finish this task very efficiently because all the amplitudes of eventually state of the part A and part B are stored in memory. For example, a circuit of 7×8 qubits with depth 35, or 8×8 qubits with depth 29 is easy to sample a large amount (e.g. $\geq 2^{32}$) of amplitudes.

When focusing on a 7×7-qubit circuit, we will introduce a novel method to calculate an amplitude of the final state. The target is to calculate

$$\alpha_x = \langle x|\mathcal{U}_{circuit} H^{\otimes 49}|00...0\rangle$$

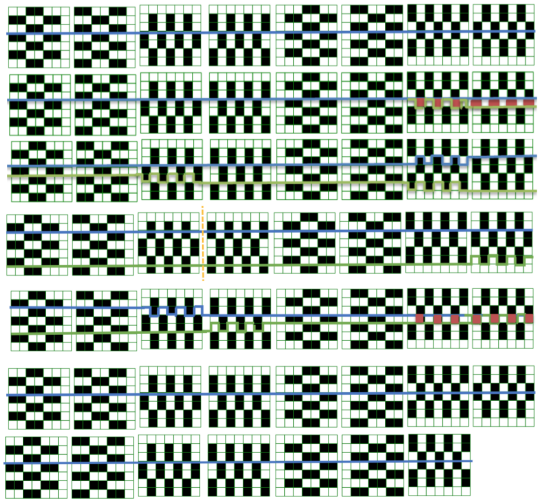

Fig. 8. Partition scheme for calculating one amplitude for 49-qubit circuit of depth 55. Note that the first 27 layers represents exactly the partition scheme for computing the complete state-vector for 49-qubit circuit of depth 27. Let $\mathcal{U}_{circuit} = \mathcal{U}_1\mathcal{U}_2$, where $\mathcal{U}_1$ represents the circuit for first 27 layers, and $\mathcal{U}_2$ represents the circuit for subsequent 28 layers. They are divided by the orange dotted line in the figure. Layer 55 is the first layer when computing $|\varphi\rangle$, the 3 cut CZ gates in this layer do not increase the space consumption because $|x\rangle$ is a classical bit, and performing these 3 CZ gates on $|x\rangle$ will not change the state, only probably multiplying it by a factor -1.

for $0 \leq x \leq 2^{49} - 1$. Since we could calculate the complete state-vector for a 7×7-qubit circuit of depth 27, we can also sample one amplitude for a circuit of depth 55. Let $U_{circuit} = \mathcal{U}_2\mathcal{U}_1$ in which $\mathcal{U}_1$ has 27 layers and $\mathcal{U}_2$ has 28 layers, $|\psi\rangle = \mathcal{U}_1 H^{\otimes 49}|00...0\rangle$ and $|\varphi\rangle = \mathcal{U}_2^\dagger|x\rangle$, we have:

$$\alpha_x = \langle\varphi|\psi\rangle$$

Thus, we calculate $|\varphi\rangle$ and $|\psi\rangle$ simultaneously in memory, during the calculation we computing the inner product of every two corresponding blocks of 2^{28} amplitudes of $|\varphi\rangle$ and $|\psi\rangle$. Sum all 2^{21} inner products and we get α_x. Because the least space consumption of simulating a circuit of depth 27 is O(2^{35}), this method can be parallelized to calculate more amplitudes.

C. Optimization for reducing communication amount

The method presented above concerns mainly the memory space limitation of Sunway. But if the network communication amount is too large in implementing this method, it will be impractical since the network bandwidth of Sunway is limited. In this subsection we describe a method to reduce the communications of a key step of our method.

Recall that equation (3) is the final step to compute the complete state-vector, and this step can be divided into $2^n/S_C$ subtasks which could be paralellized, where n is the number of qubits in the whole circuit and S_C is the space consumption of part C. In Fig. 2, $S_C = 2^2$ and $n = 4$. For 49-qubit circuits

of depth 27 and depth 35, (3) can be rewritten as:

$$|\psi^{out}\rangle = \bigoplus_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}} |\psi^{out,q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle$$
$$= \bigoplus_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}} (\mathcal{U}_C|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle) \quad (4)$$

where $\mathcal{U}_C$ denotes the unitary operation represented by part C. Note that part C is a 28-qubit circuit, and $|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle$ can be treated as a 28-qubit state. So, computing $|\psi^{out,q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle$ is essentially the same as simulating a 28-qubit circuit, which could be finished in a single node. Now the remaining problem is to efficiently prepare $|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle$ for each set of possible values of $(i_0, i_1, ..., i_{13}, i_{42}, i_{43}, ..., i_{48})$, where $i_k \in \{0, 1\}$. Note that

$$|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle = \sum_{l_1,l_2,...,l_t,i_{21},i_{22},...,i_{27}} (|\phi^{out,q_{14},q_{15},...,q_{20}}_{i_0,i_1,...,i_6,l_1,...,i_{27}}\rangle \otimes |\xi^{out,q_{21},q_{22},...,q_{42}}_{i_{42},i_{43},...,i_{48},l_1,...,i_{27}}\rangle) \quad (5)$$

where t is the number of decomposed CZ gates, $t = 7$ for depth 27, and $t = 14$ for depth 35. $|\phi^{out,q_{14},q_{15},...,q_{20}}_{i_0,i_1,...,i_6,l_1,l_2,...,l_t,i_{21},i_{22},...,i_{27}}\rangle$ is a 2^7-length state-vector, where the indices of qubit 0-6 are $i_0, i_1, ..., i_6$, the indices of control qubits of t decomposed CZ gates are $l_1, l_2, ..., l_t$, and the indices of control qubits of 7 implicit decomposed CZ gate are $i_{22}, ..., i_{27}$. Similarly, $|\xi^{out,q_{21},q_{22},...,q_{41}}_{i_{42},i_{43},...,i_{48},l_1,l_2,...,l_t,i_{21},i_{22},...,i_{27}}\rangle$ is a 2^{21}-length state-vector, where $i_{42}, i_{43}, ..., i_{48}$ are the indices of qubit 42-48, the definition of $l_1, l_2, ..., l_t$ and $i_{21}, i_{22}, ..., i_{27}$ are the same as in $|\phi\rangle$. Because of implicit decomposition, for each value set of $i_{21}, i_{22}, ..., i_{27}$, $|\xi^{out,q_{21},q_{22},...,q_{42}}_{i_{42},i_{43},...,i_{48},l_1,l_2,...,l_t,i_{21},i_{22},...,i_{27}}\rangle$ has only 2^{14} non-zero amplitudes. Thus, (5) can be rewritten as:

$$|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle = \sum_{l_1,l_2,...,l_t} (\bigoplus_{i_{21},i_{22},...,i_{27}} |\phi^{out,q_{14},q_{15},...,q_{20}}_{i_0,i_1,...,i_7,l_1,...,i_{27}}\rangle \otimes |\xi^{out,q_{28},q_{29},...,q_{42}}_{i_{42},...,i_{48},l_1,...,l_t,q_{21}=i_{21},...,q_{27}=i_{27}}\rangle) \quad (6)$$

where $|\phi^{out,q_{14},q_{15},...,q_{20}}_{i_0,i_1,...,i_6,l_1,...,i_{27}}\rangle (|\phi\rangle$ for short) is still of 2^7-length but $|\xi^{out,q_{28},q_{29},...,q_{42}}_{i_{42},...,i_{48},l_1,...,l_t,q_{21}=i_{21},...,q_{27}=i_{27}}\rangle (|\xi\rangle$ for short) is of 2^{14}-length.

Now we consider how to realize equation (6). The data from part A have 2^{t+7} complex numbers (amplitudes), and the data from part B have 2^{t+14} complex numbers. Thus, for depth 27, $t = 7$ and (6) can be finished in a single node since it has enough memory space to store the data both from parts A and B. However, for depth 35, this would not work since the data from part B have $2^{14+21} = 2^{35}$ complex numbers. Directly calculating (6) needs 2^7 nodes to communicate (e.g. an *MPI_Gather* is feasible). This is very inefficient, because there are $2^{49-28} = 2,097,152$ entities of (6) to calculate, and each entity needs an *MPI_Gather* in 128 nodes. Assume that we have $2^{15} = 32,768$ free nodes for this work. One round can calculate $2^{15-7} = 256$ entities. Then we need $\frac{2,097,152}{256} = 8192$ rounds to finish all the jobs, which would cost at least several days. However, we can slightly change the form of (6) to:

$$|\psi^{q_{14},q_{15},...,q_{41}}_{i_0,i_1,...,i_{13},i_{42},i_{43},...,i_{48}}\rangle = \bigoplus_{i_{21},i_{22},...,i_{27}} (\sum_{l_1,l_2,...,l_t} |\phi^{out,q_{14},q_{15},...,q_{20}}_{i_0,i_1,...,i_7,l_1,...,i_{27}}\rangle \otimes |\xi^{out,q_{28},q_{29},...,q_{42}}_{i_{42},...,i_{48},l_1,...,l_t,q_{21}=i_{21},...,q_{27}=i_{27}}\rangle) \quad (7)$$

Note that for each element *in the bracket*, the data from part A and part B have 2^{21} and 2^{28} complex numbers, respectively. This can be finished in a single node and the length of result is 2^{21}. We further append $|\phi\rangle$ with qubit 8-14 so that $|\phi\rangle$ also becomes a 2^{14}-length vector:

$$|\phi\rangle = |\phi^{out,q_8,q_9,...,q_{20}}_{i_0,i_1,...,i_6,l_1,...,i_{27}}\rangle$$

Now $|\phi\rangle \otimes |\xi\rangle$ is of 2^{28}-length and can still be calculated in a single node. Actually, it can be regarded as a 28-qubit state:$|\psi^{q_8,...,q_{20},q_{28},...,q_{42}}\rangle$. Our target is $|\psi^{q_{14},...,q_{27},q_{28},...,q_{42}}\rangle$. Note that for every group of 128 nodes, they store $|\psi^{q_8,...,q_{20},q_{28},...,q_{42}}\rangle$ for $(i_{21}, i_{22}, ..., i_{28}) = (0, 0, ..., 0)$ to $(1, 1, ..., 1)$. Thus, by performing an all-to-all on these 128 nodes, we get 128 entities of (6). Assume again that we have 2^{15} free nodes to work for part C. Then we can get 2^{15} entities in a single round. Furthermore, in only $2^{21-15} = 64$ rounds, the whole task can be finished. Though the communication time for one round here is slightly longer than an *MPI_Gather* on 128 nodes, the total communication time greatly decreases, which makes the task for depth 35 feasible.

Solving (6) for 49-qubit circuit of depth 39 is essentially the same as the case of depth 35.

D. Single node optimizations

In this subsection, we introduce our optimizations for the quantum circuit simulation at the single node (single core group) scale. The optimizations for every single node is very important for improving the performance of our method.

As agreed in Section I, one node represents a core group in Sunway, and it has 1 master core and 64 slave cores. Each node has 8GB DDR memory, shared by both master and slave cores. The maximum qubit number that could be simulated on one node is 28 if we use two doubles to represent an amplitude, since $2^{28} \times 16B = 4GB$. To obtain full power of Sunway TaihuLight, one must distribute most of the computing tasks to slave cores. Each slave core has a private and separate memory unit called *local data memory (LDM)* (also known as *scratch pad memory*) of 64kB size. To execute high-speed calculations a slave core must fetch data from the main memory to its own LDM and keeps it in LDM for calculation as long as possible. This fetching-data behavior is usually called *direct memory access* (DMA). To get high DMA bandwidth, it usually requires the data fetched consecutive.

If LDMs fetch data from the main memory for every gate performed, and put the data back to main memory when the calculation is finished, the simulation will be inefficient due to

Depth	25	30	34	38
Swaps	6	7	8	9

TABLE II
FREQUENCY OF SWAPS FOR 28-QUBIT UNIVERSAL RANDOM CIRCUITS WITH DIFFERENT DEPTH, IN THE CASE THAT THE NUMBER OF LOCAL QUBITS IS 10.

low flop-to-Byte ratio[3]. In our experiments, the average execution time is 0.32s per gate in such way, thus the performance is bounded by DMA speed[4]. However, we can take advantage of the data locality in a better way to reduce the DMA amount. For example, if each slave core has fetched from the main memory 16KB data in its LDM, that is, $2^{14-4} = 2^{10}$ amplitudes. For any gate performed on those qubits with their ranks lower than 10 (0 is the lowest rank), the calculation can be executed in this 16 KB data, i.e. α_i and α_{i+2^k} are in the same LDM. Thus we can perform a bunch of gates acting on low-rank qubits once instead of performing the circuit gate by gate. We call these 10 qubits with lowest ranks *local qubits*, and other 18 qubits are *global qubits*[5]. This idea is similar to the gate fusion techniques in [12], which deals with the gates on low-rank qubits in cache. However, gate fusion is one-off, which can only be applied at the start of the circuit.

To make the above procedure repeatable, we also adopt the qubit reordering method. Qubit reordering are used in [6] and [9] to reduce the network communications. Here, our aim is to maintain the data locality and reduce the amount of memory access. That is, only diagonal gates, or non-diagonal gates on local qubits are calculated. To accomplish this, we need to execute two types of qubit rank swaps:

- Swap the qubits of rank 0-9 and qubits of rank 11-20
- Swap the qubits of rank 14-20 and qubits of rank 21-27

With a gate scheduling preprocessing program, which also utilizes the diagonal properties of gate T and CZ, we get the amount of swaps for 28-qubit quantum supremacy circuit: We achieve fast swaps of qubit rank with the help of slave cores. Swapping the qubits of rank 0-9 and qubits of rank 10-19 is essentially a transpose of a complex matrix. The dimension of this matrix is $2^{10} \times 2^{10}$, and $2^{28-20} = 256$ matrices in total need to be transposed. Swapping the qubits of rank 14-20 and qubits of rank 20-27 is similar, which is equivalent to a transpose of a $2^8 \times 2^8$-dimensional matrix, but each element of this matrix contains 2^{12} amplitudes.

Register communication is a unique function in Sunway CPU, designed for fast data transmission between LDMs. The qubit reordering can be further optimized using this feature. Because the slave cores in a row can send/receive messages in a communication bus, we can treat a row of 8 separate LDMs as a composite LDM, while the data exchange between these

[3]A 2×2 complex matrix multiplying a 1×2 complex vector needs 14 float-point operations. For a 28-qubit circuit, each non-diagonal gate requires $2^{28-1} \times 14$ float-point operations and 4G DMA *get* and *put*

[4] The DMA *get* bandwidth and *put* bandwidth are both less than 25GB/s in our experiments

[5]this is different to [6, 9]. Their distinction of 'global' and 'local' is about the multi-node and single-node level. While our 'global' corresponds to main memory, and 'local' corresponds to LDM

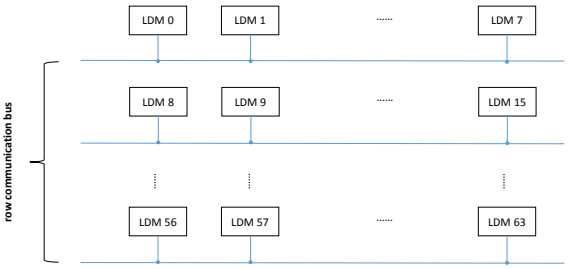

Fig. 9. Register communication. There are 8 row communication buses and 8 column communication buses in a core group. In this figure only row communication buses are plotted, since in our experiments only row communication is used. In fact, the row communication is used to achieve the swap between qubits of rank 8-10 and qubits of rank 11-13. Register communication has very high bandwidth, more than 200GB/s in total for 8 row communication buses. So the cost of this step is very small, and we can treat 8 LDMs in a row as new a composite LDM. Thus the number of local qubits turns into 14.

Depth	18	26	34	42	50
Gates	258	375	492	609	726
Swaps	3	4	5	6	7
Time	15.4s	22.6s	29.6s	36.7s	44.3s
Speedup	7.04	6.95	6.96	6.95	6.88

TABLE III
PERFORMANCE OF SIMULATING 28-QUBIT CIRCUITS WITH DIFFERENT DEPTH ON A SINGLE NODE. THE NUMBER OF GLOBAL QUBITS IS 14. SPEEDUP IS THE SPEED-UP RATIO TO METHOD OF PERFORMING GATE BY GATE WITHOUT ANY OPTIMIZATION BUT USING THE SLAVE CORES TO ACCELERATE.

8 LDMs is fulfilled by register communication. If each slave core fetches 32kB consecutive data from the main memory to its LDM, there will be 11 local qubits. But when considering a composite LDM formed by a row of LDMs, the number of local qubits turns into 14. Thus we only need to execute one type of qubit swap:

- Swap the qubits of rank 0-13 and qubits of rank 14-27

This is simply a transpose of a $2^{14} \times 2^{14}$-dimensional complex matrix, which can be quickly accomplished by slave cores. Because of the high bandwidth of register communication, the time consumed on register communication is very small, so as to improving the overall performance of single node case.

E. Other standard optimizations

In this subsection we briefly introduce some other standard optimizations provided by Sunway TaihuLight, which can be exploited for our simulation of quantum circuits.

1) Vectorization: Sunway TaihuLight provides many 256-bit data types. In our simulation the type $doublev4$ is adopted for vectorization. The data stored in LDM is a complex number array with one double as the real part of a complex number and another double as its imaginary part. To vectorize the double-precise float-point calculation, we put four amplitudes into two $doublev4$ registers v_r, v_i once. However, the real parts of these amplitudes are not consecutive, and the imaginary parts neither. For example, $\alpha_i = a[2 * i] + j * a[2 * i + 1]$ for

$0 \leq i \leq 3$, and v_r = simd_set_doublev4$(a[0], a[2], a[4], a[6])$, v_i = simd_set_doublev4$(a[1], a[3], a[5], a[7])$. These two *simd_set_doublev4* operations consumes 8 CPU cycles. But if let v_r = simd_set_doublev4$(a[0], a[1], a[2], a[3])$ and v_i = simd_set_doublev4$(a[4], a[5], a[6], a[7])$, and use two *vshuffle* operations to make v_r and v_i correct for calculation, the whole process only consumes 4 cycles. This optimization is also used in writing data from $doublev4$ registers back to LDM, improving the performance and FLOPs.

2) Instruction Reordering: Another optional optimization is instruction pipeline. The put and $store$ operations, together with the multiply-add operations, can form a pipeline to further reduce the calculation time, especially when the data dependency between adjacent instructions is little. This optimization also improves the computational efficiency.

III. Numerical Experiments and Results

Sunway TaihuLight is one of the most powerful supercomputer with over 100 Pflops computing capacity [1]. To test the limitation of our simulator on Sunway, we used 131072 nodes, which is around 80% of computing resource of the whole machine with nearly 1PB main memory in total. We implemented our simulator in C++ for master core managing programs and C for slave core computing programs. We use MPI for internode communications. To facilitate the most of computing capacity of Sunway we have used the *athread* library [8].

For the task of simulating a 49-qubit circuit of depth 35 and computing the complete state-vector, it takes around 3.7 hours. The bottleneck is (6), because solving 2^{21} entities of (6) needs $2^{21+7+35} = 2^{63}$ times of complex number multiplication. This step occupied around 90% of the run-time in the simulation of 35-depth circuit. For the task of simulating a 49-qubit circuit of depth 39, it takes around 4.2 hours. The reason for causing this drop in performance is that part C has 42 qubits in the case of depth 39, while part C only has 28 qubits in the case of depth 35. Thus in the case of depth 35, the calculation in part C are all within single nodes. While calculating a block of part C in the case of depth 39 is essentially simulating a 42-qubit circuit of depth 15, which needs one all-to-all communication on 2^{14} nodes [9]. Because there are $2^{49-42} = 128$ blocks to calculate, the amount of communication increases a lot.

To calculate one amplitude from a 49-qubit circuit of depth 55, the main work is simulating two 49-qubit circuits of depth 27. Since the least space consumption of simulating a circuit of depth 27 is O(2^{35}), this task could be parallelized when we have enough computing nodes. We give the strong scaling behaviour in Figure 10. In this task, the bottleneck becomes (4) as the complexity of this step is $O(2^{21} n_C 2^{28}) = O(n_C 2^{49})$, where n_C is the number of circuits in part C. Specially, calculating one amplitude for a 49-qubit circuit of depth 55 takes 95.2 minutes using 32768 nodes.

Though we have not tried simulation of a 49-qubit circuit of depth 40 yet, the existing numerical results shows that 49-qubit circuit of depth 40 is no more out of reach, according to previous analysis. But the cost will be more expensive than the case of depth 39.

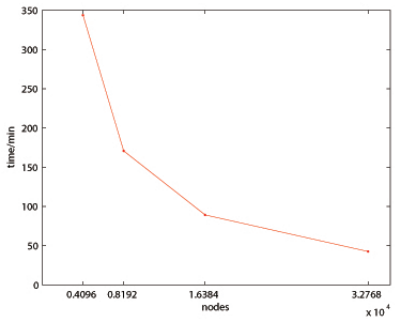

Fig. 10. Strong scaling behaviour of simulation 49-qubit circuit of depth 27, solving the complete state-vector

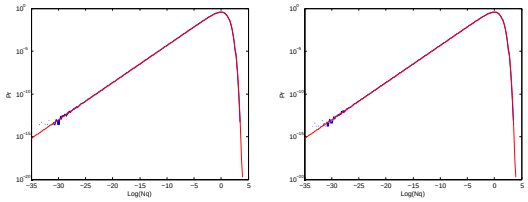

Fig. 11. Histograms of log-transformed outcome probabilities for 49-qubit circuits, compared to theoretical Porter-Thomas distribution [4]. The left is the result of simulating a circuit of depth 35. The right is the result of simulating a circuit of depth 39.

IV. Conclusion and Future Works

This paper describes our method and implementation of quantum circuit simulator on Sunway TaihuLight. The results raise the bar of "quantum supremacy", as simulating a 49-qubit circuit of depth 40 on a classical computer used to be considered difficult or even impossible. However, classical computers have their limits on simulating quantum circuits. Ultimately, there will be one day that quantum computer can solve certain problems which classical computers cannot. Before that day comes, simulating quantum circuits on classical computers is crucial to understand the power and limit of quantum computers. Even after that day, a simulator of quantum circuits on a classical computer will still be helpful for design, synthesis, testing and verification of quantum circuits.

Follow-up work of this paper includes further optimizations of our simulator and adding some new functions to it, e.g. (1) quantum circuit testing and verification; (2) simulation of real circuits with quantum noise, and (3) simulation, debugging and verification of more sophisticated quantum algorithms and quantum programs (with control flows) [16]. We hope that our simulator can be extended to serve as a useful tool in the design, testing and validation of future quantum computer hardware and software.

ACKNOWLEDGEMENT

We are very grateful to Gan Lin, Yu Haining, Zhang Wei, Shi Shupeng, Meng Hongsong, Yu Hongkun, Zhao Wenlai and the whole team at the National Super Computing Center in Wuxi for their kind helps. Special thanks go to Liu zhao, who has given us a lot of useful suggestions and assistance. This work is partially supported by the National Natural Science Foundation of China and the National Supercomputing Center in Wuxi.

REFERENCES

[1] "top500 list june 2017". https://www.top500.org/lists/2017/06/.

[2] Scott Aaronson and Lijie Chen. Complexity-theoretic foundations of quantum supremacy experiments. *arXiv preprint arXiv:1612.05903*, 2016.

[3] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, and H. Neven. Simulation of low-depth quantum circuits as complex undirected graphical models. *ArXiv e-prints*, December 2017.

[4] Sergio Boixo, Sergei V Isakov, V N Smelyanskiy, Ryan Babbush, Nan Ding, Zhang Jiang, John M Martinis, and Hartmut Neven. Characterizing quantum supremacy in near-term devices. *arXiv: Quantum Physics*, 2016.

[5] Davide Castelvecchi et al. Quantum cloud goes commercial, 2017.

[6] K De Raedt, K Michielsen, De Hans Raedt, B Trieu, G Arnold, M Richter, Th Lippert, Hiroshi Watanabe, and Nobuyasu Ito. Massively parallel quantum computer simulator. *Computer Physics Communications*, 176(2):121–136, 2007.

[7] Richard P Feynman. Simulating physics with computers. *International Journal of Theoretical Physics*, 21:467–488, 1982.

[8] Haohuan Fu, Junfeng Liao, Jinzhe Yang, Lanning Wang, Zhenya Song, Xiaomeng Huang, Chao Yang, Wei Xue, Fangfang Liu, Fangli Qiao, Wei Zhao, Xunqiang Yin, Chaofeng Hou, Chenglong Zhang, Wei Ge, Jian Zhang, Yangang Wang, Chunbo Zhou, and Guangwen Yang. The sunway taihulight supercomputer: system and applications. *Science China Information Sciences*, 59(7):072001, Jun 2016.

[9] Thomas Haner and Damian S Steiger. 0.5 petabyte simulation of a 45-qubit quantum circuit. *arXiv preprint arXiv:1704.01127*, 2017.

[10] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2000.

[11] Edwin Pednault, John Gunnels, Giacomo Nannicini, Lior Horesh, Thomas Magerlein, Edgar Solomonik, and Robert Wisnieff. Breaking the 49-qubit barrier in the simulation of quantum circuits. 10 2017.

[12] Mikhail Smelyanskiy, Nicolas Sawaya, and Alan Aspuruguzik. qhipster: The quantum high performance software testing environment. *arXiv: Quantum Physics*, 2016.

[13] Chao Song, Kai Xu, Wuxin Liu, Chui-ping Yang, Shi-Biao Zheng, Hui Deng, Qiwei Xie, Keqiang Huang, Qiujiang Guo, Libo Zhang, Pengfei Zhang, Da Xu, Dongning Zheng, Xiaobo Zhu, H. Wang, Y.-A. Chen, C.-Y. Lu, Siyuan Han, and Jian-Wei Pan. 10-qubit entanglement and parallel logic operations with a superconducting circuit. *Phys. Rev. Lett.*, 119:180511, Nov 2017.

[14] Krysta M Svore, A Geller, M Troyer, J Azariah, C Granade, B Heim, V Kliuchnikov, M Mykhailova, A Paz, and Dave and Roetteler, Martin Wecker. Q#: Enabling scalable quantum computing and development with a high-level dsl. *Proceedings of the Real World Domain Specific Languages Workshop*, 2018.

[15] Dave Wecker and Krysta M Svore. Liqui$|\rangle$: A software design architecture and domain-specific language for quantum computing. *arXiv: Quantum Physics*, 2014.

[16] Mingsheng Ying. Foundations of quantum programming, 2016.