Supplementary Materials for QuanTrans

I. Quantum Circuit Simulation

This section introduces the mathematical model of quantum computing and reviews related work on distributed quantum circuit simulations.

A. Basics of Quantum Circuit Simulation

Quantum computing is a computational model based on the principles of quantum mechanics, and the full-state mathematical model allows classical computers to simulate quantum states and operations.

1) **Data structures:** In full-state simulations, the process of quantum computing is described by a quantum circuit consisting of quantum bits (qubits) and quantum gates.

Qubits and state vectors. The basic units for storing information in quantum computing are called *qubits*. Mathematically, the state of an n-qubit system can be represented by a vector $|\phi\rangle$ known as the *state vector*, which contains 2^n complex numbers called *amplitudes*. The state vector is represented as

$$|\phi\rangle = \alpha_{00\ldots0}\,|00\ldots0\rangle + \ldots + \alpha_{11\ldots1}\,|11\ldots1\rangle = \begin{bmatrix} \alpha_{00\ldots0} \\ \alpha_{00\ldots1} \\ \vdots \\ \alpha_{11\ldots1} \end{bmatrix},$$

where $\sum_{x\in\{0,1\}^n} |\alpha_x|^2 = 1$, as $|\alpha_x|^2$ represents the probability of the quantum system collapsing into the corresponding basis state $|x\rangle$ after measurement. The 0-th qubit represents the least significant qubit in this paper. For example, as shown in Fig. 1(a), if initially $q_2 = q_1 = |0\rangle$ and $q_0 = |1\rangle$, the initial state vector of this 3-qubit circuit is $|001\rangle$, with only one amplitude $\alpha_{001} = 1$.

Quantum gates and gate matrices. In a quantum circuit, the operations on qubits are described using *quantum gates*, which can change the state of a quantum system. Mathematically, each quantum gate can be represented by a complex unitary matrix. Fig. 1(a) shows a quantum circuit with two levels of quantum gates, containing a single-qubit gate G_A, a 2-qubit gate controlled-G_B (G_B is applied to q_1 if $q_2 = |1\rangle$), and a non-controlled 2-qubit gate G_C. In this paper, we mainly consider these three types of gates, as they are commonly used and can form a set of universal quantum gates. Given a quantum gate G_X, if it has i input qubits, mathematically speaking, G_X is a $2^i \times 2^i$ matrix. However, when G_X is applied to a quantum circuit, its input qubits may not be adjacent, thus covering more qubits. We use M_X to denote the complete gate matrix of G_X in the circuit. If G_X covers j qubits, M_X has a size of $2^j \times 2^j$. For example, G_A has only one input qubit q_1, so G_A and M_A are both 2×2 matrices. G_B itself has only one input qubit q_0, so G_B is a 2×2 matrix. Controlled-G_B has two adjacent inputs, so M_B is 4×4. Similarly, G_C has 2 adjacent inputs, so G_C and M_C are both 4×4 matrices.

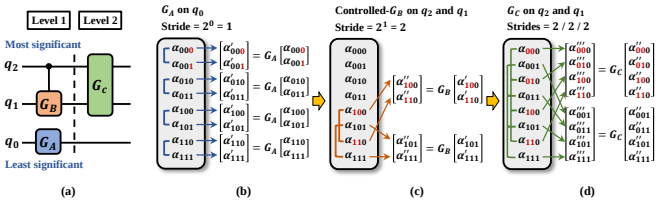

Fig. 1. (a) An example of a 3-qubit 2-level quantum circuit. The state vector simulation (SVSim) for (b) G_A, (c) G_B, and (d) G_C.

2) **Simulation methods:** Two basic methods involved in existing distributed simulation techniques will be introduced.

Operation matrix simulation. The state vector of an n-qubit circuit has a size of 2^n. To change its state, the *operation matrix simulation* (OMSim) constructs an operation matrix of $2^n \times 2^n$ at each level of the circuit. For the j-th level, use M_i^j to represent the i-th quantum gate matrix from high-order to low-order. The operation matrix O_j of level j can be constructed by

$$O_j = M_1^j \otimes M_2^j \otimes M_3^j \otimes \ldots,$$

where $\otimes$ refers to the tensor product of matrices. Note that if no gate is applied to a qubit, it can be supplemented with a 2×2 identity matrix I. For example, in Fig. 1(a), the operation matrices of level 1 and 2 are $M_B \otimes M_A$ and $M_C \otimes I$, respectively. Once the operation matrix is computed, we can use matrix-vector multiplication to update the state vector, transforming the state of a quantum system. The state vector after T levels can be described as

$$|\phi_T\rangle = O_T \cdot O_{T-1} \cdot \ldots \cdot O_1 \cdot |\phi_0\rangle\,,$$

where $|\phi_0\rangle$ is the initial state vector. The matrix-vector mathematical model serves as a universal representation of quantum computing, but the overhead is extremely high. For an n-qubit circuit, the time complexity of the multiplication of the $2^n \times 2^n$ operation matrix and 2^n state vector every level is $O(2^{2n})$, which severely limits the scale of quantum circuits that can be simulated.

State vector simulation. Instead of constructing the entire operation matrix, the *state vector simulation* (SVSim) directly applies the transformations specified by quantum gates to the state vector to reduce the overhead. For a single-qubit gate G_A, applying it to the k-th qubit is equivalent to applying its gate matrix to every pair of amplitudes whose binary indices differ only in the k-th bits ($k = 0, \ldots, n-1$):

$$\begin{bmatrix} \alpha'_{*\ldots*0_k*\ldots*} \\ \alpha'_{*\ldots*1_k*\ldots*} \end{bmatrix} = G_A \cdot \begin{bmatrix} \alpha_{*\ldots*0_k*\ldots*} \\ \alpha_{*\ldots*1_k*\ldots*} \end{bmatrix}. \tag{1}$$

The stride between $\alpha_{*\ldots*0_k*\ldots*}$ and $\alpha_{*\ldots*1_k*\ldots*}$ is 2^k, i.e., $(*\ldots*1_k*\ldots*)_2 - (*\ldots*0_k*\ldots*)_2 = 2^k$. For example, Fig. 1(b) shows how to do SVSim for G_A on q_0. Four pairs of amplitudes are 00$\underline{0}$-00$\underline{1}$, 01$\underline{0}$-01$\underline{1}$, 10$\underline{0}$-10$\underline{1}$, and 11$\underline{0}$-11$\underline{1}$.

Their binary indices differ only in the 0-th position. The stride within each pair is $2^0 = 1$. As shown in Fig. 1(c), for a 2-qubit gate controlled-G_B on a control qubit q_c and a target qubit q_t, assuming $c > t$, G_B of size 2×2 is applied to q_t if $q_c = |1\rangle$:

$$\begin{bmatrix} \alpha'_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix} = G_B \cdot \begin{bmatrix} \alpha_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix}. \tag{2}$$

As shown in Fig. 1(d), for a non-controlled 2-qubit gate G_C of size 4×4 on q_c and q_t, SVSim can be described as:

$$\begin{bmatrix} \alpha'_{*\ldots*0_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*0_c*\ldots*1_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha'_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix} = G_C \cdot \begin{bmatrix} \alpha_{*\ldots*0_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*0_c*\ldots*1_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*0_t*\ldots*} \\ \alpha_{*\ldots*1_c*\ldots*1_t*\ldots*} \end{bmatrix}, \tag{3}$$

where the strides between these four amplitudes are 2^t, $2^c - 2^t$, and 2^t. Since each gate on the circuit requires performing SVSim on the entire state vector, for an n-qubit quantum circuit with T levels, the time complexity of SVSim is $O(n \cdot 2^n \cdot T)$, which is still relatively high.

B. Distributed state vector simulation

Due to the high complexity of quantum circuit simulation methods, the memory and computing power of a single computational node are insufficient. Therefore, some work has explored the distributed architecture for simulating quantum circuits, leveraging the computational resources of a distributed cluster. To utilize multiple nodes, it is necessary to partition the original simulation task and allocate it to multiple machines. A common method of partitioning is to distribute the elements of the state vector evenly across different nodes and then use SVSim to update the state vector. After partitioning the state vector, some SVSim operations require data communication between nodes, which becomes a performance bottleneck for distributed simulations.

As shown in (1), (2), and (3), an SVSim operation involves multiple amplitudes from the state vector. If the amplitudes involved are stored on different nodes, communication is required. Given an n-qubit circuit, the state vector has $N = 2^n$ amplitudes. Assume that there are H identical computational nodes, where H is an exponential power of 2, i.e., $H = 2^h$. The amplitudes are evenly distributed across these H nodes, where each node stores $L = N/H = 2^{n-h}$ amplitudes. Let $l = n - h$. Since there are 2^l amplitudes stored per node, for gates applied to low-order l qubits, SVSim can be performed locally, as the strides between the amplitudes involved are less than 2^l, called local SVSim. However, for gates applied to the remaining high-order h qubits, the strides exceed 2^l, which means that amplitudes stored on the other nodes are involved, called distributed SVSim. Communication among nodes is required.

In the next section, we will review the applicability conditions and basics of QuanPath in detail.

II. The Foundation of QuanPath

QuanPath is a distributed quantum circuit simulation framework. As described in Sect. I-B, given an n-qubit T-level circuit, it divides the state vector of size 2^n into H segments. Each computational node in the distributed cluster processes

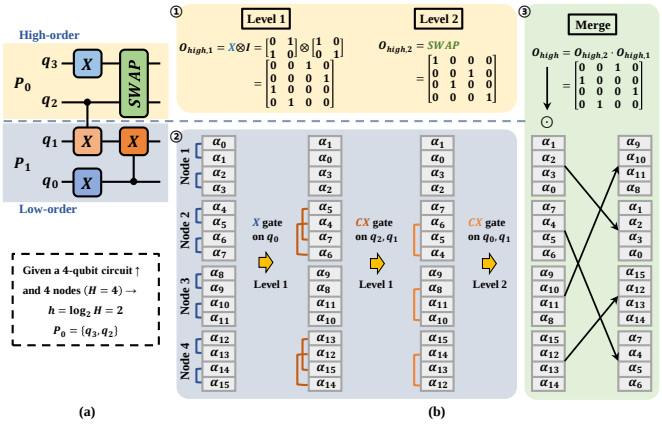

Fig. 2. (a) An example of a 4-qubit 2-level quantum circuit. (b) The simulation process of QuanPath for a separable quantum circuit.

one out of H segments. In this way, we can parallelize the computation and reduce the memory requirements on each node. From the perspective of qubits, the division of state vectors is equivalent to partitioning n qubits into 2 sets: high-order qubit set P_0 of size h and low-order qubit set P_1 of size l, where $h = \log_2(H)$, and $l = n - h$. The number of nodes determines the size of a high-order qubit set. For example, Fig. 2(a) shows a 4-qubit 2-level quantum circuit ($n = 4$, $T = 2$). Given $H = 4$ nodes, the state vector of size $N = 16$ is partitioned into 4 segments, each node stores $L = 4$ amplitudes. The number of high-order qubits $h = 2$.

A. Applicability Conditions

QuanPath proposes a low-communication distributed simulation technique to fuse intermediate multi-level distributed computation to only one level. It is important to note that QuanPath is specifically designed for *separable circuits*, whose definition is as follows [1].

Definition 1. *Let $Qub(C) = \{q_{n-1}, \ldots, q_0\}$ be the set of input qubits of circuit C and $Qub(G)$ be the set of input qubits of gate G. Let $Qub(C)$ be partitioned into high-order qubit set P_0 and low-order qubit set P_1. For exmaple, given $Qub(C) = \{q_5, q_4, q_3, q_2, q_1, q_0\}$, P_0 can be $\{q_5, q_4\}$ and P_1 can be $\{q_3, q_2, q_1, q_0\}$. Gate G is called a "multi-qubit gate" if $|Qub(G)| > 1$. Then, a quantum circuit C is "separable" by P_0 and P_1 if the following condition is satisfied:*

> *For each multi-qubit gate G in C such that $Qub(G) \cap P_0 \neq \emptyset$ and $Qub(G) \cap P_1 \neq \emptyset$, all qubits in $Qub(G) \cap P_0$ must be control qubits of G.*

For example, given two circuits in Fig. 3, let $P_0 = \{q_3, q_2\}$ and $P_1 = \{q_1, q_0\}$. The circuit in Fig. 3(a) is separable, but the one in Fig. 3(b) is inseparable by P_0 and P_1, as three multi-qubit gates do not satisfy the above condition.

B. The Process of QuanPath

Given the circuit in Fig. 2(a) separable by $P_0 = \{q_3, q_2\}$ and $P_1 = \{q_1, q_0\}$, the simulation process using QuanPath is shown in Fig. 2(b). For high-order qubits, OMSim is used.

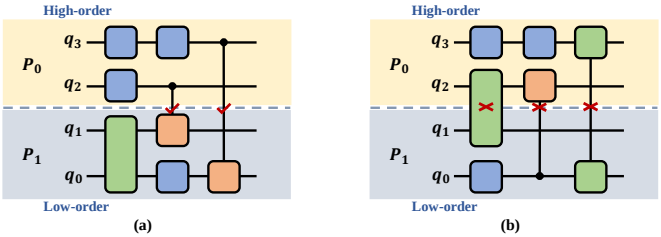

Fig. 3. Examples of two 4-qubit 3-level quantum circuits that are respectively (a) separable and (b) inseparable by P_0 and P_1.

For low-order qubits, local SVSim is performed on each node parallelly. A final distributed *merge* operation is conducted to combine the results. Proof of correctness can be seen in [1].

① **OMSim for high-order h qubits.** Simulating high-order gates by distributed SVSim can result in huge communication overhead in traditional methods. Therefore, QuanPath uses OMSim instead to construct an operation matrix $O_{high,j}$ of size $H \times H$ for gates on high-order h qubits at level j of the circuit $(1 \leq j \leq T)$.

② **SVSim for low-order l qubits.** For gates applied to low-order l qubits, SVSim can be done locally, as the strides of amplitudes involved are less than L. QuanPath uses local SVSim to update the partitioned state vector of size L on each node parallelly. The updated partial state vector stored on the i-th node is denoted as V_i $(1 \leq i \leq H)$.

③ **Merge operation.** Before the final merge operation, ① and ② are completely independent and can be performed in parallel. After that, the merge operation combines them to obtain the computation result of the state vector V', which is partitioned into H parts, denoted as $V'_1, \ldots, V'_H$. The calculation formula for the merge operation is

$$V'_{N\times 1} = \begin{bmatrix} V'_1 \\ V'_2 \\ \vdots \\ V'_H \end{bmatrix} = (O_{high,T} \cdot \ldots \cdot O_{high,1})_{H\times H} \odot \begin{bmatrix} V_1 \\ V_2 \\ \vdots \\ V_H \end{bmatrix}_{N\times 1},$$

where $\odot$ is partition-wise vector scalar multiplication. It is defined as

$$O_{H\times H} \odot V_{N\times 1} = V'_{N\times 1}, \text{ where } V' \text{ consists of}$$
$$V'_i = \sum_{j=1}^{H} o_{ij} \cdot V_j,\ i = 1, \ldots, H.$$

o_{ij} is the element located in the i-th row and j-th column of matrix O.

REFERENCES

[1] Y. Song, E. H.-M. Sha, Q. Zhuge, W. Xiao, Q. Dai, and L. Xu, "QuanPath: achieving one-step communication for distributed quantum circuit simulation," Quantum Information Processing, vol. 23, no. 1, p. 1, 2023. [Online]. Available: https://doi.org/10.1007/s11128-023-04192-x