Distributed Memory Techniques for Classical Simulation of Quantum Circuits

Ryan LaRose
Department of Computational Mathematics, Science, and Engineering
Department of Physics
Michigan State University

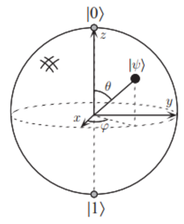
Introduction

Quantum computers use properties from quantum mechanics to rapidly process information. Recent research shows quadratic to exponential speedups in computational problems from quantum chemistry, optimization, and machine learning. Information is stored in quantum bits (*qubits*) and manipulated by quantum gates. Advantages over classical computers occur on quantum systems with around 50 qubits, but state of the art technology, while rapidly changing, sits well below this threshold. To study large scale quantum computing requires numerical simulation. We use (classical) HPC resources provided by ICER to simulate quantum computers with up to 33 qubits using hybrid OpenMP and MPI parallelization.

Memory Requirements of Qubits

A quantum bit (*qubit*), denoted $|\psi\rangle$, is a two level quantum system. It can be represented by a two dimensional vector on the unit sphere in the complex plane, known as the *Bloch sphere*. The state of multiple qubits is the tensor product of the states of individual qubits $|\psi_i\rangle, \mathrm{i} = 0, \ldots, \mathrm{n} - 1$:

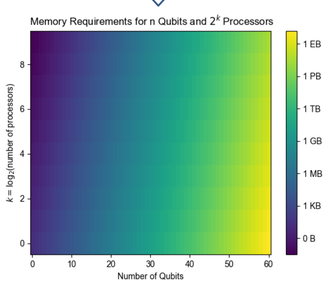
$$|\psi\rangle = |\psi_0\rangle \otimes |\psi_1\rangle \otimes \cdots \otimes |\psi_{n-1}\rangle$$



The Bloch Sphere: Graphical representation of a qubit in a two dimensional Hilbert space.

The dimension of the state vector $|\psi\rangle$ increases by two for every additional qubit. Storing $|\psi\rangle$ on a computer thus requires memory exponential in the number of qubits n. This memory "explosion" [1] is the primary difficulty in simulating quantum computers.

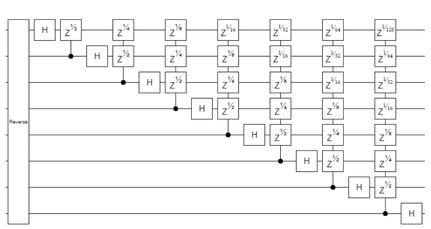
Qubit Memory Requirements



Methodology

To overcome this difficulty, we use distributed memory parallelism to partition the state vector across multiple processors. This technique allows us to simulate significantly larger quantum circuits and study algorithms such as the quantum Fourier transform (shown below), a subroutine in polynomial time factoring [2].

Quantum Fourier Transform Circuit: Lines represent qubits and boxes represent quantum gates.



Distributed Memory Implementation

We partition the state vector of 2^n amplitudes among 2^k processors and apply a single qubit gate via Algorithm 1. For qubits with positional index $i \leq k$, no communication is required between processors, and amplitudes can be updated simultaneously and independently via Algorithm 1. For qubits with positional index $i > k$, communication is required and processor pairs must be determined via Algorithm 2.

Let the state vector be represented by amplitudes α indexed with subscripts in binary notation. To apply a single qubit gate Q (2 x 2 matrix) with elements q_{jk} to the i^{th} qubit, one must apply the matrix product to each amplitude whose index differs in the i^{th} bit. This is implemented in Algorithm 1.

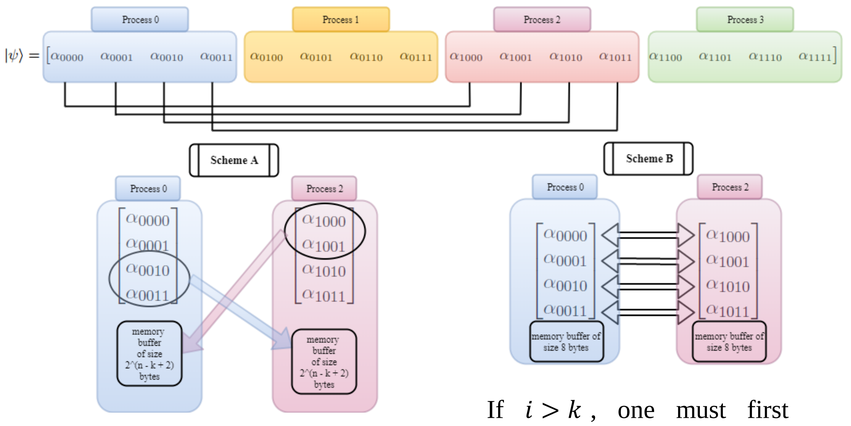
Algorithm 1 Single Qubit Gate Application

```
procedure APPLY(ψ, n, i, Q)
    for (j = 0; j < 2^n; j+=2^(i+1)) do
        for (k = j; k < j + 2^i; k++) do
            temp1 = ψ[k]
            temp2 = ψ[k + 2^i]
            ψ[k] = Q[0][0] * temp1 + Q[0][1] * temp2
            ψ[k + 2^i] = Q[1][0] * temp1 + Q[1][1] * temp2
```

Amplitude Update Equation

$$\alpha_{*\cdots *0_i *\cdots *} = q_{11}\alpha_{*\cdots *0_i *\cdots *} + q_{12}\alpha_{*\cdots *1_i *\cdots *}$$
$$\alpha_{*\cdots *1_i *\cdots *} = q_{21}\alpha_{*\cdots *0_i *\cdots *} + q_{22}\alpha_{*\cdots *1_i *\cdots *}$$

Communication Protocols



Algorithm 2 Determining Communication Pairs

```
procedure MY_PAIR(my_rank, n, k, i)
    m = n - k
    if (q >= m) then:
        return (my_rank * 2^m + 2^i) % 2^n // 2^m
procedure COMPUTE_PAIRS(n, k, i)
    pairs = hash table
    ranks = array(0, 1, 2, ..., 2^k - 1)
    while ranks is not empty do
        p = ranks[0]
        pairs[p] = MY_PAIR(p, n, k, i)
        remove p and my_pair from ranks
    return pairs
```

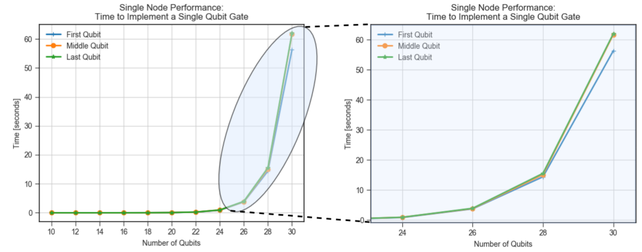
If $i > k$, one must first determine which processors need to communicate, then establish a protocol by which processors send and receive amplitudes of the state vector. Algorithm 2, shown on the left, efficiently determines processor pairs.

Communication Scheme A in the figure above has each processor exchange half their amplitudes. While requiring less communication latency, a larger memory buffer on each processor is needed. We implement scheme B to maximize the number of qubits that can be simulated by minimizing the extra memory required on each node.

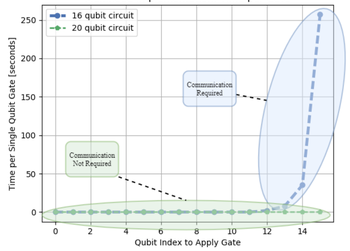
Results and Timing Benchmarks

One the intel16 cluster on the Laconia HPC system, our single node implementation can simulate up to 30 qubits, requiring 0.55 seconds for a single qubit gate on circuits of less than 24 qubits. In the distributed implementation, our communication protocol allows simulation of quantum circuits of up to 33 qubits with the state vector partitioned across 64 processors, the best single qubit gate (SQG) application time being 3.04 seconds.

Single Node Performance



Multinode Performance



Comparison of Performance

Simulator	Supercomputer	Bench-mark	Max Qubits	Time [s]
This work	Laconia	SQG	33	3.04
QX	[unknown]	ENT	29	63.02
LiQUi|⟩	Desktop	ENT	23	4.09
qH*i*PSTER	Stampede	SQG	40	1.22
Haner, Steiger	Cori II	SQG	45	0.97

- QX (N. Khammassi et. al)
- LiQUi|⟩ (arXiv:1402.4467)
- qH*i*PSTER (arXiv:1601.07195)
- Haner, Steiger (arXiv:1704.01127)

SQG = Single Qubit Gate ENT = Entanglement circuit

Discussion

Because our communication scheme is dominated by latency, the single qubit gate application time increases quickly for qubits that require communication, as can be seen in the plot on the left. To improve this, one can use a communication scheme that sends more amplitudes (at the cost of a larger memory buffer). The ideal communication protocol lies somewhere between scheme A and scheme B. Future work will explore these communication schemes as information compression in storing the state vector $|\psi\rangle$.

Acknowledgements

This work was supported in part by Michigan State University through computational resources provided by the Institute for Cyber-Enabled Research.

References

[1] Richard P. Feynman, *Simulating Physics with Computers*, International Journal of Theoretical Physics, Vol. 21, 1982.
[2] Peter W. Shor, *Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer*, Proceedings of the 35th Annual Symposium on Foundations of Computer Science, 1995.