Project Proposal: A classical simulator for quantum circuits dominated by Clifford gates

Patrick Rall and Iskren Vankov, advised by David Gosset
California Institute of Technology - February 14, 2016

Introduction

Simulation of quantum circuits on classical computers is a useful tool for the verification and validation of quantum devices. As we build systems with more and more qubits, we will enter a regime where simulation will become impractical. While this is a central reason for our interest in quantum devices, extending the reach of classical computers for verification purposes will have practical applications in experiments conducted in the near future.

A recent paper by Sergey Bravyi and David Gosset [1] describes an algorithm for quantum circuit simulation that is polynomial in the number of qubits. By utilizing the Gottesmann-Knill theorem to efficiently simulate Clifford gates, the procedure is only exponential in the number of T-gates used in the circuit. A preliminary Matlab implementation was capable of simulating a hidden shift quantum algorithm with 40 qubits and 50 T-gates on a laptop.

The goal of this project is to develop an open-source application that permits other physicists to use this algorithm. It will employ a massively parallel architecture, so it can be run on hundreds of CPUs in a computer cluster. It will also be able to make use of Graphics Processing Units (GPUs) present in many laptops and desktops, which could be useful for physicists with limited access to a cluster. Writing both parallel CPU and GPU implementations sounds like a bit of a stretch. I think we should state those two as the viable options and then, in the course of the project, decide which one (or with a small probability both) to do.

In addition to creating a useful tool, this project will be an excellent learning experience. We will learn more about the inner workings of the Gottesmann-Knill theorem, stabilizer states, and mathematical tools used in modern quantum information science. We will also gain experience in writing distributed software for several architectures, and how to automatically manage highly parallel code across many computers.

Algorithm sketch

The paper [1] describes two algorithms. The algorithm for computing the output probability of a string x which runs in

$$\text{poly}(n, m) + 2^{0.5t} t^3.$$

Another algorithm for sampling a string x from the output distribution runs in

$$\text{poly}(n, m) + 2^{0.23t} t^3 w^4,$$

where n is the number of qubits, m the total number of gates, t the number of T-gates and w the size of the string x.

- Both algorithms replace all occurrences of the T-gate with the gadget shown in Figure 1, so a Clifford-only circuit must be simulated.
- Each T-gate gadget consumes a non-stabilizer 'magic state':

$$A = \frac{1}{\sqrt{2}}(0 + e^{i\pi/4}1).$$

  Thus the circuit requires an additional $A^{\otimes t}$ state.
- The state $A^{\otimes t}$ is decomposed into a linear combination of $\chi \ll 2^n$ stabilizer states.
- The action of the circuit onto each stabilizer state in this linear combination can be simulated independently. This is why parallelism is so appropriate for this algorithm.
- The final measurements are also performed independently on each resulting state in the linear combination.
- Computing the norm of the final state can be achieved in $O(\chi t^3 \epsilon^{-2})$ for relative error ϵ. **Can this procedure be parallelized too?** Shouldn't be the case?

Software architecture

Scheduler, Network Manager (Python)
The heart of the application will perform all tasks that cannot be done in parallel. It will network together all computers that will perform computation and scan for computing resources. It will decompose $A^{\otimes t}$ state into χ stabilizer states and assign them to the available resources. Once parallel computation is complete it will normalize the resulting state and produce the final output.

CPU Simulator (C++)
A simple implementation of a stabilizer circuit simulator for a single state, aimed for high performance on a single core.

GPU Simulator (CUDA)
A highly parallel implementation of a stabilizer circuit simulator for several states. This component must be capable of adapting to a range of GPUs with different capabilities.

Command Line Interface (Python)
A simple tool capable of reading circuits from a file, starting the simulation, and presenting the result.

Web Interface (Python, Javascript)
A web server Shouldn't that be just a web interface, not a server? The computations will still happen locally for the user, we'll just be using their browser as a visual environment. Otherwise it would imply we're providing a whole server. providing a graphical user interface. It provides a tool for entering circuits **(maybe using LIQUID?)**, simulating them on demand, and gathering statistics for the results.

Time plan

Before Spring Term We will read papers, regularly meet with David Gosset, and solidify our understanding of the algorithms. We will study GPU programming and evaluate challenges and limitations in building a parallel stabilizer circuit simulator.

March A initial implementation of a stabilizer circuit simulator in C++ should be written during spring break. Unfortunately, I won't be around here during the break. Starting at the beginning of next term should still be fine. That way if there are any questions on how to proceed we can immediately dive into details when term begins and regular meetings resume. Spring term begins March 28. The last week of March should be dedicated to implementing the algorithms in [1] in python, utilizing the working stabilizer circuit simulator.

April April will be dedicated to building the GPU simulator, which will be unfamiliar territory for both participants. Iskren will be taking *CS179 GPU Programming* (textbook: [4]) and will be able to contribute with knowledge from the class. Both of us are experienced with the 'dive in at the deep end and figure out how to swim'-approach to coding.

May On May 2 David Gosset **will leave Caltech (how accurate is this?)**, so the physics of the algorithm must be working in all parts of the code by then. May will be spent writing the command line interface and web interface, as well as documenting and polishing the code. We will request time on Caltech's compute cluster to test the algorithm.

June The last weeks of term will be spent completing and polishing the documentation. Commencement is June 10.

I think we should avoid stating particular technologies (e.g. Python, JS, etc.) in the plan because we'll both be learning on the goal and might find out something else is more suitable.

References

[1] S. Bravyi, D. Gosset. Jan 29, 2016. "Improved classical simulation of quantum circuits dominated by Clifford gates". `http://arxiv.org/abs/1601.07601`

[2] S. Bravyi, G. Smith, J. Smolin. Jun 3, 2015. "Trading classical and quantum computational resources". `http://arxiv.org/abs/1506.01396`

[3] A. Aaronson, D. Gottesman. Jun 25 2004. "Improved Simulation of Stabilizer Circuits". `http://arxiv.org/abs/quant-ph/0406196`

[4] N. Wilt. Jun 12, 2013, Addison-Wesley Professional. “The CUDA Handbook: A Comprehensive Guide to GPU Programming”. http://www.cudahandbook.com/