Chapter 10: Concluding Remarks

Yu-Chia Liu
5/25/2021

Outline

- Quantum Computers can be Digital
- Effective Error-Mitigation Techniques
- Achieving Greater Efficiency by Breaking Abstractions
- In Need for a New Systems Stack
    - Noise Mitigation and Error Correction
    - Quantum Programming Languages
    - Automated Compilation
    - Quantum-Classical Co-Processing
    - Scalable Software & Hardware Verification
- Last But not Least… Simulations

Quantum Computers can be Digital

- Quantum computing can be viewed as an analog enterprise, with its exponentially complex superposition and probabilistic outcomes of measurement
- The digital discipline is accomplished through quantum error correction codes and the measurement of error syndromes through ancilla qubits (scratch qubits)
- Operations on error-corrected qubits can be viewed as digital rather than analog, and only a small number of universal operations are needed for universal quantum computation
  - Hadamard gate (H gate)
  - Pi/8-phase gate (T gate)
  - Controlled-Not gate (CNOT gate)
- Quantum error correction codes have historically required enormous overhead

Effective Error-Mitigation Techniques

- High physical error rates in quantum devices can lead to high error-correction overhead
- Physical error-mitigation techniques promise to make devices more reliable and make low-overhead error-correction code possible
- Those techniques rely on examining the physical basis of the error
    - Combining noisy qubits to generate less noisy qubits at the physical level
    - Using qubits to control the noise source
- Possible error correction on applications' side
    - For example, Generalized Superfast [1], an encoding of quantum chemistry problems, can correct for a single qubit error

ESCAL Extreme Storage & Computer Architecture Lab

UC RIVERSIDE UNIVERSITY OF CALIFORNIA

Breaking Abstractions

- Compilation tools will play a critical role for practical quantum computation
- But those tools have to break traditional abstractions and be customized for machine device characteristics in a manner never before seen in classical computing
    - Compilers can target not only a specific program input and machine size, but the condition of each qubit and link between qubits
    - Instead of compiling to an instruction set, compilers can directly target a set of analog control pulses
    - Instead of using binary logic to target two-level qubits, compilers can target an n-ary logic composed of qudits

In Need for a New Systems Stack

- Quantum computing is at a similar stage of development as classical computing in the 1950s
- The systems stack for practical quantum computation
  1. An architecture with error-correction, error-mitigation and application-level fault-tolerance
  2. An expressive programming language
  3. An automated compilation and memory management framework
  4. An Integrated quantum-classical co-processing scheme
  5. Scalable software and hardware verification

Noise Mitigation and Error Correction

- Two classes of strategies for information protection and error reduction in quantum system
- Characterizing Realistic Noises
  - State and process tomography
  - Randomized benchmarking
- Noise Mitigation Strategies
  - Randomized compiling
  - Noise-Aware Mapping
  - Crosstalk-Aware Scheduling
- Quantum Error Correction
  - A way to systemically detect and correct a quantum error
    - Redundant Encoding
    - Digitizing Quantum Error

Quantum Programming Languages

- Low-Level Machine Languages
  - Quantum Assembly Language (QASM)
  - A direct translation from a quantum circuit to a sequential description instructions for executing a quantum program
  - Sequential QASM language suffers from its limitation on modeling complex classical control
- High-Level Programming Languages
  - Represent complex classical and quantum information processing in quantum algorithms
  - Designing a language that enables programmers to exploits these quantum properties on real hardware while maintaining usability remains challenging
  - Balancing between abstraction and detail is key
    - Exposing device specifics helps programmers write more efficient code
    - But it dramatically increases the complexity of the language
  - Because of the hybrid nature of classical and quantum information processing, most existing quantum programming languages are Domain-Specific Languages (DSLs)

Automated Compilation

- A quantum compiler aims to translate, transform and optimize a high-level quantum program into native instructions that a quantum machine recognizes and natively supports, balancing practical architectural constraints
- For a program to be realizable on a given hardware, a number of architectural constraints must be satisfied:
  - Instruction set
  - Qubit communication
  - Hardware noise
  - Available parallel control
- Circuit Synthesis and Compilation
  - Unitary synthesis focuses on exactly or approximately expressing arbitrary unitary transformations in a sequence of elementary gates.
  - The goal of gate scheduling is to utilize commutation relations to determine the ordering of the operations and to use circuit equivalence to simplify quantum programs
  - Qubit mapping is to strategically assign the variables in a quantum program to the qubits available in the system.
- Pulse Compilation
  - In the NISQ era and beyond, we will need to orchestrate the simultaneous quantum operations on hundreds or thousands of qubits
  - Classical and quantum control of qubits
  - Pulse generation and optimization
  - Calibration and Verification

Quantum-Classical Co-Processing

- Quantum computing hardware is currently envisioned to be a hardware accelerator for classical computers
- Classical processing and classical control play vital roles in quantum computing
    - A quantum algorithm generally involves classical pre- or post-processing
    - Efficient classical controls are needed for running the algorithm on hardware
- Advantages:
    - It sidesteps the “innovator’s dilemma” by leveraging an initial guess derived from classical technology, rather than directly competing with that technology.
    - Hybrid algorithms break a long program into multiple iterations of short programs, which allows us to effectively utilized the limited number of instructions a quantum machine can reliably execute.
    - It allows us to pick small but classically challenging problems that can be represented in a small number of quantum bits.
    - We have a clear measure of success, as we know that classically-computed ground state energy can be significantly higher than experimentally-observed values.

Scalable Software & Hardware Verification

- Hardware Verification
  - Refers to the problem of verifying that hardware is capable of performing quantum logic operations as intended by a program
  - At a basic level, the behavior of quantum devices can be characterized through quantum tomography
  - As machine becomes larger, we will need a system-level approach
- Software Verification
  - Refers to the problem where we want to verify that a quantum program is bug-free and implements the desired transformation
  - The purpose of software verification is typically two-fold:
    - High-level programs are bug-free
    - Compiler transformations preserve logical equivalence
  - Three useful verification approaches
    - Tracing via classical simulation
    - Assertion via quantum property testing
    - Proof via formal logic

Last But not Least… Simulations

- Once we have functional quantum computers, we may be able to use quantum algorithms to implement theorem provers and constraint solvers
- However, always be bootstrapping from simulator to hardware, from hardware to larger hardware.
    - Leading Techniques:
        - Density matrices
        - Stabilizer formalism
        - Graphical models
    - It allows us to execute a quantum program and verify its correctness even when no quantum hardware is available
    - It also sheds light on the not-so-well-understood computational power boundary between classical computers and quantum computers

Extreme Storage & Computer Architecture Lab

References

- [1] Setia, Kanav, et al. "Superfast encodings for fermionic quantum simulation." *Physical Review Research* 1.3 (2019): 033033.