양자 소프트웨어 시뮬레이터 비교 및 분석

김제인, 조성민, 서승현
한양대학교 전자공학과
rean5123@hanyang.ac.kr, smcho3315@hanyang.ac.kr, seosh77@hanyang.ac.kr

Comparison and Analysis of Quantum Software Simulators

Jane Kim, Seong-Min Cho, Seung-Hyun Seo
Dept. of Electrical Engineering, Hanyang University

요 약

최근 IBM, Intel 과 같은 글로벌 ICT 기업들과 여러 스타트업들이 양자 컴퓨터 개발에 성공하였으며 그에 따라 양자 시뮬레이터와 컴파일러에 대한 관심이 높아졌다. 여러 개의 시뮬레이터가 존재하는 만큼 시뮬레이터마다 제공하는 기능과 성능 역시 제각각 다르다. 본 논문에서는 비교적 접근이 쉬운 파이썬과 Q# 기반의 대표적인 양자 시뮬레이터 3 가지(Qiskit, Project Q, Quantum Development Kit)에서 제공하는 기능들을 소개하고 시뮬레이션 실행시간을 비교한다. 10 뎁스의 20 큐비트 회로에서는 QDK 시뮬레이터가 0.227 초로 실행 시간이 가장 짧았고, 10 큐비트의 10 뎁스 회로의 경우 Project Q 가, 1000 뎁스의 경우 Qiskit 이 가장 짧은 실행시간으로 측정됐다.

1. 서론

최근 많은 기업들이 양자컴퓨터 개발에 뛰어들고 있다. Intel 은 최대 128 개의 큐비트를 제어할 수 있는 칩을 개발했으며, IBM 은 65 큐비트의 연산능력을 갖는 양자 컴퓨터를 시범 운영하고 있다. 이외에도 구글(Google), 마이크로소프트(MS) 등 글로벌 ICT 기업들과 이온큐(IonQ), 리게티(Rigetti) 등과 같은 스타트업들이 양자 컴퓨터 개발에 성공하였다. 양자컴퓨터의 하드웨어 개발은 이제 가속화 단계이지만 양자 컴퓨팅을 지원하는 프로그래밍 언어는 20 년 전부터 개발되어 왔다. 최근에는 양자 컴퓨터의 상용화가 가까워짐에 따라 양자 컴퓨팅 소프트웨어가 폭발적으로 증가하고 있다. 양자 컴퓨팅 소프트웨어에는 양자 풀-스택 라이브러리와 그 외에도 수십 가지의 양자 시뮬레이터들[1]이 있다. 하지만 많은 양자 시뮬레이터들이 생김으로써 연구자들이 자신의 연구에 어떤 시뮬레이터를 사용해야 할 것인지 알기 어렵다. 따라서 양자 소프트웨어의 연구가 잘 이루어지기 위해서는 현재의 양자 시뮬레이터들의 특징을 파악해 각 연구에 맞는 시뮬레이터의 사용이 필요하다.

본 논문에서는 연구자들이 양자 시뮬레이터를 처음 이용할 때 약간의 가이드라인이 될 수 있도록 대표적인 3 가지 양자 시뮬레이터가 제공하는 기능과 특징들에 대해 설명하고, 각각의 성능을 비교한다. 본 논문은 다음과 같이 구성된다; 2 장에서는 파이썬과 Q#기반의 대표적인 3 가지 양자 시뮬레이터(Qiskit, Project Q, Quantum Development Kit)에서 제공하는 기능 및 특징들을 소개하고, 3 장에서는 이러한 시뮬레이터들의 큐비트 및 뎁스 별 실행 시간을 비교한다.

"본 연구는 과학기술정보통신부 및 정보통신기획평가원의 대학 ICT 연구센터육성지원사업의 연구결과로 수행되었음" (IITP-2021-2018-0-01417)

2. 양자 소프트웨어 시뮬레이터

본 장절에서는 대표적인 양자 소프트웨어 시뮬레이터 3 가지 Qiskit, Project Q, QDK(Quantum Development Kit)의 주요 특징과 기능에 대해 소개한다.

2.1 Qiskit

Qiskit[2]은 2017 년 IBM 에서 배포한 오픈소스 양자 풀-스택 라이브러리이다. Qiskit 은 Terra, Aer, Ignis, Aqua 네 가지로 구성되어있다.

- Terra: 나머지 Qiskit 하위 구조들이 놓아지는 기반을 형성한다. 또한 펄스와 회로 단위에서 양자 프로그램을 구성하는 기반을 제공한다.
- Aer: 양자 회로를 위한 고성능 시뮬레이터 프레임워크를 제공한다.
- Ignis: 잡음과 오류 관리에 있어 특화됐다. 게이트 검증을 통해 게이트를 향상시키며, 노이즈가 있을 때 교정 회로(calibration circuits)를 구동 한다. 양자 오류 정정을 설계하거나 오류 특성을 찾으려는

사람들이 주로 사용한다.

- Aqua: 양자 컴퓨팅을 위한 응용 프로그램을 만드는데 사용된다. 특정 연산 작업을 위한 가속화 장치로 양자 컴퓨터를 사용한다.

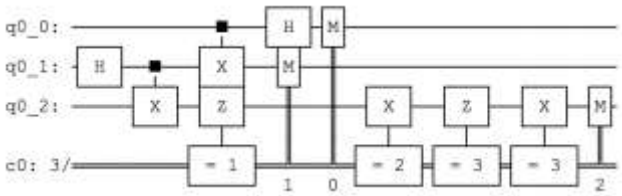
2.1.1 Aer 시뮬레이터 백엔드

Qiskit 에서 Aer 백엔드를 통해 아래의 3 가지 시뮬레이터를 사용할 수 있다.

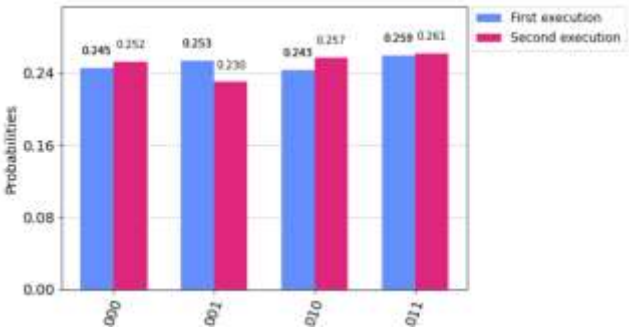
- QasmSimulator: 최대 32 큐비트까지 실행 가능하다. Qiskit 회로의 멀티-샷 실행이 가능하고 카운트와 메모리를 반환한다. 샷은 시뮬레이션시 설계한 회로의 실행 횟수를 의미한다.
- StatevectorSimulator: Qiskit 회로의 싱글-샷 실행이 가능하고 애플리케이션 실행 후 최종 상태 벡터를 반환한다.
- UnitarySimulator: Qiskit 회로의 싱글-샷 실행이 가능하고 회로 자체의 최종 유니타리 행렬을 반환한다.

2.1.2 시각화

위의 시뮬레이터들을 이용해 양자 회로의 데이터를 시각화 할 수 있다. 아래 그림 1 은 Qasm-Simulator 를 사용하여 3-큐비트 퀀텀 텔레포테이션 회로[5]를 설계한 후 qiskit 라이브러리의 'circuit_drawer'함수를 사용하여 시각화한 회로를 보여준다.



(그림 1) Qiskit 퀀텀 텔러포테이션 회로



(그림 2) Qiskit 히스토그램

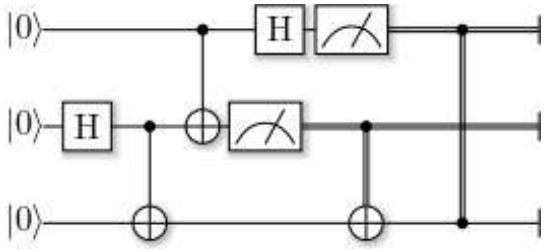
그림 2 는 그림 1 의 회로를 1000 샷씩 2 번 실행하여 측정한 결과값들의 비율을 히스토그램으로 표현한 그림이다. qiskit 라이브러리의 'plot_histogram' 함수를 사용하여 시각화할 수 있다. 회로를 여러 번 실행한 결과를 히스토그램에 나타낼 수 있고 레이블 이름 및 위치, 막대 색깔과 정렬 순서 등을 설정할 수 있다.

2.2 Project Q

Project Q[3]는 ETH Zurich 에서 개발된 Python 언어 기반의 오픈소스 양자 소프트웨어 플랫폼이다. Project Q 는 다음 6 가지의 백엔드 기능을 제공한다.

- Simulator: 빠른 시뮬레이션이 가능한 고성능 시뮬레이터
- Emulator: 양자 알고리즘의 정확성 검증
- IBM Quantum Experience: 실제 IBM 의 5 큐비트 양자 컴퓨터 상에서의 양자 회로 동작
- Resource Counter: 양자 회로의 게이트 수와 뎁스 카운팅
- Drawing Engine: 양자 회로의 시각화
- Command Printer: 명령어 출력

Simulator 백엔드는 30 큐비트까지 지원하며, Emulator 백엔드는 50 큐비트까지 지원한다. Drawing Engine 시뮬레이터는 양자 회로도를 그리는 tex 파일의 코드를 제공하며, 이 tex 파일을 통해 양자 회로도를 pdf 파일로 출력할 수 있다.



(그림 3) Project Q 퀀텀 텔레포테이션 회로

2.3 QDK (Quantum Development Kit)

QDK[4]는 마이크로소프트(Microsoft)에서 제공하는 Q#기반의 프로그래밍 개발 키트이다. Q#은 오픈소스 양자 프로그래밍언어이다. Q# 프로그램은 주피터 노트북을 통해 콘솔 애플리케이션으로 실행되거나 파이썬 또는 닷넷(.NET) 프로그램에서 사용될 수 있다. QDK 는 다음 3 가지의 기능을 제공한다.

- Full state simulator: 최대 30 큐비트까지의 풀-스테이트 시뮬레이션을 지원한다.
- Trace simulator: 실제로 양자 프로그램을 실행하지 않고 양자 컴퓨터의 상태만을 시뮬레이션한다. 특정 인스턴스를 실행하는데 필요한 리소스를 추정할 수 있다.
- Toffoli simulator: NOT, CNOT, CCNOT 양자 연산자들만 지원한다. 풀-스테이트 시뮬레이터에 비해

기능은 떨어지지만 수백만 큐비트를 시뮬레이션 할 수 있다.

3. 시뮬레이터별 성능

본 장에서는 동일한 환경에서 각 시뮬레이터들의 큐비트 및 뎁스 별 실행 시간을 비교한다. Qiskit 은 Terra 를 기반으로 'Aer' 백엔드의 'qasm simulator'를 통해 양자 회로에 대한 시뮬레이션을 진행하며, Project Q 에서는 'Simulator' 백엔드를 통해 시뮬레이션을 진행한다. QDK 에서는 'Full state simulator'를 사용하였다.

3.1 시뮬레이터별 특징 비교

[표 4] 시뮬레이터 특징 비교

시뮬레이터	Qiskit (qasm_simulator)	Project Q (Simulator)	QDK (Full-state simulator)
큐비트 수	~ 32 qubits	~30 qubits	~30 qubits
사용 요금	X	X	X
프로그래밍 언어	Python	Python	Q#
양자 회로 시각화	O	O	X

표 4 는 2 장에서 설명한 시뮬레이터들의 특징을 정리하여 나타낸 것이다. Qiskit 의 'qasm_simulator'의 경우 최대 32 큐비트까지 사용이 가능하고, Project Q 의 'simulator'와 QDK 의 'full-state simulator'의 경우 최대 30 큐비트까지 시뮬레이션이 가능하다. 세가지 시뮬레이터 모두 오픈소스로 사용 요금이 없다. Qiskit 과 Project Q 는 파이썬 기반의 프로그래밍 언어를 사용하고 양자 회로의 시각화가 가능하다. QDK 의 경우 Q# 기반의 프로그래밍 언어를 사용하고 양자 회로의 시각화를 지원하지 않는다.

3.2 시뮬레이터 실행시간 비교

본 논문에서는 각 양자 시뮬레이터에서의 큐비트 및 뎁스 별 실행 시간을 비교한다. 큐비트 별 실행 시간은 아다마르(Hadamard) 게이트가 각 큐비트마다 10 개씩 있는 10 뎁스의 회로에서 큐비트를 늘려가며 측정하였으며, 뎁스 별 실행 시간은 10 큐비트에서 아다마르 게이트의 뎁스를 늘려가며 측정하였다. 실행시간은 파이썬 time 라이브러리를 사용하여 측정한다. 양자 시뮬레이터는 윈도우 상의 주피터 노트북 환경에서 실행하였으며, 비교에 사용된 컴퓨터의 사양은 다음과 같다.

- CPU: Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz
- RAM: 16 GB

[표 2] 큐비트 증가에 따른 실행시간 비교 (10 뎁스)

Simulator \ Qubit	1Q	10Q	20Q
Qiskit	0.127	0.136	0.656
Project Q	0.001	0.090	178.787
QDK	0.057	0.085	0.227

[표 3] 뎁스 증가에 따른 실행시간 비교 (10 큐비트)

Simulator \ Depth	Depth 1	Depth 10	Depth 100	Depth 1000
Qiskit	0.130	0.136	0.180	0.672
Project Q	0.025	0.090	0.850	8.040
QDK	0.798	0.085	0.150	0.719

표 2 는 아다마르(Hadamard) 게이트가 각 큐비트마다 10 개씩 있는 10 뎁스의 회로에서 큐비트 증가에 따른 실행시간을 비교한 것이다. 1 큐비트를~~빗을~~ 사용했을 경우에는 Project Q, 10 큐비트와 20 큐비트를~~빗을~~ 사용했을 경우 QDK 시뮬레이터의 실행시간이 가장 짧았다.

표 3 은 10 큐비트 회로에서~~의~~ 아다마르 게이트의 뎁스 증가에 따른 실행시간을 비교한 결과이다. 뎁스가 1 일 경우 Project Q, 뎁스가 10, 100 일 경우 QDK, 뎁스가 1000 일 경우 Qiskit 시뮬레이터의 실행시간이 가장 짧았다.

4. 결론

무료로 양자 회로를 설계하고 시뮬레이션을 할 수 있는 많은 오픈 소스 시뮬레이터들이 존재한다. 그 중 본 논문에서 소개한 Qiskit, Project Q, QDK 는 모두 무료로 오픈되어 있고, 주피터 노트북등을 통해 쉽게 사용할 수 있다. 이 시뮬레이터들을 통해 본인이 설계한 양자 회로를 실제 양자 컴퓨터에서 실행해 보지 않아도 회로의 성능 및 상태 확인이 가능하다. 따라서 본 논문에서는 접근성이 쉬운 세가지 시뮬레이터를 선별하여 각 시뮬레이터의 특징 및 기능을 서술한 뒤, 뎁스와 큐비트 수에 따른 양자 회로의 시뮬레이션 시간을 비교해 보았다.

참고문헌

[1] Mark Fingerhuth, Open-Source Quantum Software Projects, accessed May 12, 2018.

[2] Qiskit 0.24.1 documentation , "2021/03/31"modified,

http://qiskit.org/documentation

[3] Damian S. Steiger, ProjectQ: An Open Source Software Framework for Quantum Computing, Institute for Theoretical Physics, Switzerland, 2018.01.31, 13p

[4] Microsoft documentation QDK , “2021/01/1”modified, https://docs.microsoft.com/en-us/azure/quantum/overview-qdk

[5] Ryan LaRose, Overview and Comparison of Gate Level Quantum Software Platforms, March 22, 2019