Quantum Programming Without Tears

Max Cutugno Josh Gordon Tino Tamon

June 8, 2018

Abstract

We benchmarked several platforms for quantum computing by running important quantum algorithms on them.

Contents

1 Platforms

We tested four platforms.

1. Liquid
2. PyQuil
3. QISkit
4. Quipper

These have been ordered in increasing difficulty to install. In the interest of reducing future suffering, we have documented our install process for each, as well as provided descriptions of them.

1.1 Descriptions

PyQuil and QISkit are very similar; they are both python based, their main purpose is to describe a quantum assembly language, and they come with an API to run your programs on a quantum computer (either at Rigetti Computing with PyQuil or IBM with QISkit).
Liquid is Microsoft's platform, and it uses the Q# language. It comes with a simulator to run circuits, with no option for running on an actual quantum computer.
Quipper is an embedded language in Haskell, and is developed by Peter Selinger and Benoit Valiron. It has three simulators, but no option to run programs on a quantum computer.

Both PyQuil, Liquid, and Quipper come with a library of quantum algorithms. Liquid has an incomplete version of QLSA, and we have not been able to run the Quipper version.

1.2 Installation

1.2.1 Liquid

1.2.2 PyQuil

To install PyQuil, we used pip. This worked on Windows, however, we found Anaconda to be easier on Mac and Linux.

```
pip install pyquil
```

But, we can also install it within a conda environment as follows:

```
$ source activate myquil
(myquil) $ pip install pyquil
(myquil) $ pyquil-config-setup
Welcome to PyQuil!
Enter the required information below for Forest connections.
If you haven't signed up yet you will need to do so first
  at https://forest.rigetti.com
Forest API Key: <ENTER API HERE>
User ID: <ENTER ID HERE>
Pyquil config file created at '/home/tino/.pyquil_config'
If you experience any problems see the guide
  at https://go.rigetti.com/getting-started
```

Now, we can run some python programs in a directory of examples:

```
(myquil) $ python run_quil.py hello_world.quil
Running Quil Program from:  hello_world.quil
---------------------------
Output:
[[1, 0, 0, 0, 0, 0, 0, 0]]
```

1.2.3 QISkit

Installing QISkit was similar; we used pip on windows and Anaconda on Mac and Linux.

```
pip install qiskit
```

It may be helpful to have Jupyter Notebook installed for running some of the examples in QISkit.

1.2.4 Quipper

Quipper was the hardest to install, as the package manager for Haskell, Cabal, did not work for some of the dependencies, namely easyrender and newsynth among others.

The key is to download the source for Quipper from the website at `https://www.mathstat.dal.ca/~selinger/quipper/`. Once you have the source, extract it to a folder of your choosing. Use cabal to install as many of the dependencies as you can. Some will fail, but this is no cause for alarm. Simply go to the Haskell package repository, hackage, and download the source for that dependency. Extract the folder with the module you need into your Quipper folder.

2 Helpful Tools

2.1 Anaconda

To install python in a self-contained environment (which may avoid subtle clashes between python2 and python3 as well as their respective libraries), one option is to use anaconda. We stress that this is optional as one can install pyquil directory without using anaconda.

First, we install anaconda into some directory, say `anaconda2`. Then, to make it visible globally, we update the bash file `.bashrc`:

```
#Place the following line in the file .bashrc (at the end)
export PATH="$USER/anaconda2/bin:$PATH"
```

Here the variable `$USER` specifies the path to the user's main directory. Make sure to source the file from the command line:

```
$ source .bashrc
```

Now conda should be visible from anywhere.

We can update conda first (to catch any latest upgrades):

```
$ conda update conda
```

Now, we create a conda *environment* for running our python based programs:

```
# create a new conda environment called 'myquil'
$ conda create --name myquil python=python3.6
$ conda env list
```

To enter the conda environment, we do

```
$ source activate myquil
(myquil) $
```

and to exit we do

```
(myquil) $ source deactivate
$
```

2.2 Jupyter Notebook

If one wishes to use Jupyter notebook, we need to the add the conda environment to the notebook. First, we enter the conda environment and then install a python kernel for jupyter notebook and connect it to the conda environment:

```
(myquil) $ pip install ipykernel
(myquil) $ python -m ipykernel install --user --name myquil
```

Now, we can run jupyter notebook (from outside conda) and choose the appropriate kernel (and hence conda environment).

3 Main algorithms

We examine several basic known quantum algorithms and verify if they are implemented within a certain quantum software platform.

NAME	Pyquil	QISkit	Quipper	Liquid
Bell circuit	✓	✓	✓	✓
teleportation	✓	✓	✓	✓
Deutsch	×	×	×	×
Deutsch-Jozsa	✓	×	×	×
Bernstein-Vazirani	✓	×	×	×
Simon	×	×	×	×
QFT	✓	×	✓	✓
Phase estimation	×	×	×	×
Shor	×	×	×	×
Grover	×	×	×	×
Singular value estimation	×	×	×	×
Hamiltonian simulation	×	×	×	×
HHL	×	×	×	×

4 Additional algorithms

1. (Dürr and Høyer) Finding the minimum element in an array.
2. (Ambainis) Determining if an array contains distinct items.
3. (Child and Goldstone) Continuous-time spatial search.

5 Cross Compilation

During testing, we developed several tools for interfacing these platforms. In the interest of not writing the same code many times, most of said tools are translators.
The focus of the translators is on a graphical tool for designing circuits we developed. This representation is called Circuit Diagrams.