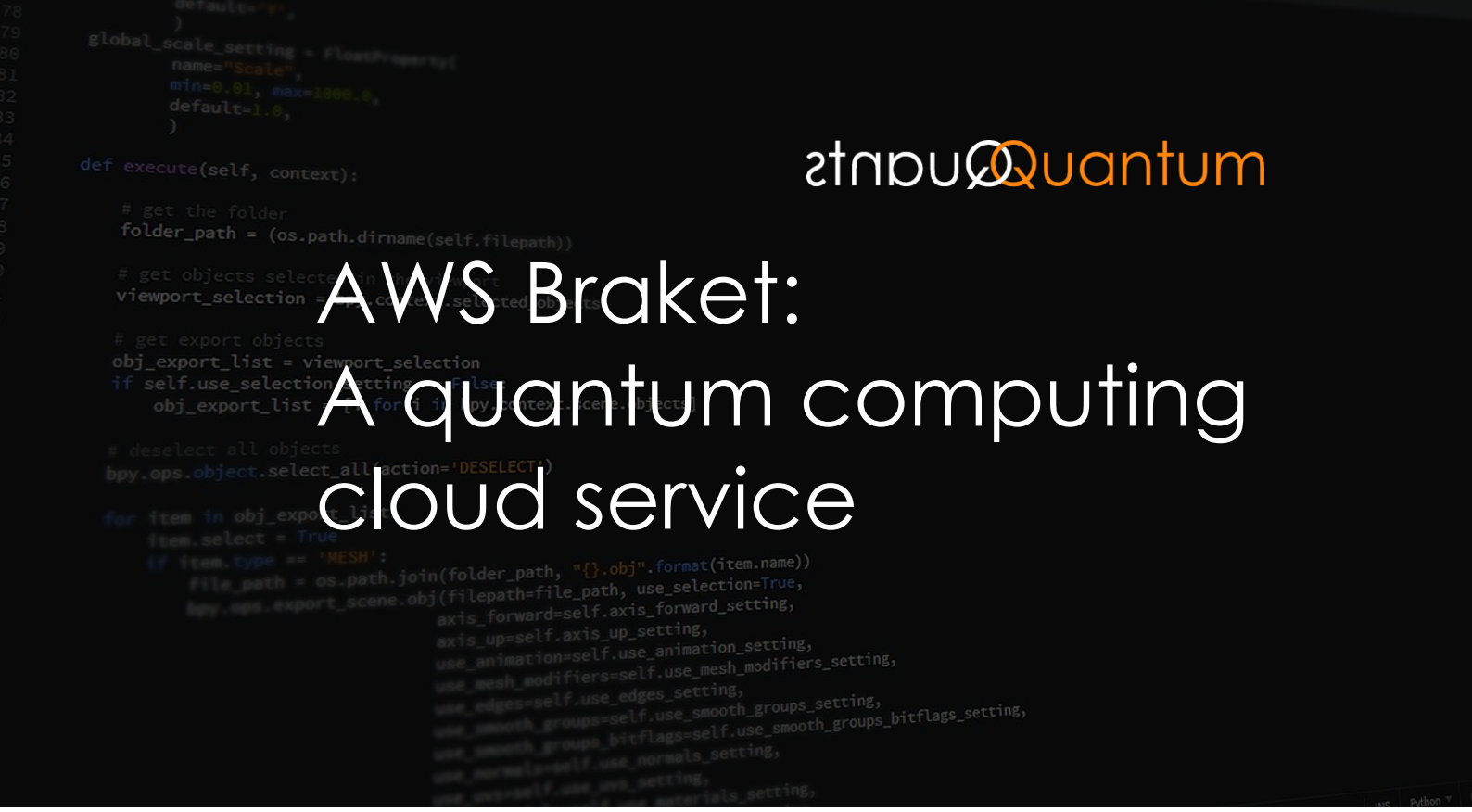

AWS Braket: A quantum computing cloud service

The use of quantum computers for practical purposes, from data science to logistic and finance applications, is becoming reality.

Several parties are building quantum computers and making them available to the businesses and organizations.

Together with the improvements on the quantum hardware, also the software to run codes on quantum chips is in constant development.

This article presents an overview of Amazon Web Services (AWS) Braket, a cloud service to run quantum computations on both quantum annealers and universal-gate quantum computer architectures.

Giuseppe Colucci

info.quantumquants@gmail.com

AWS Braket

Amazon Web Services (AWS) Braket[1] is a cloud service part of the AWS platform which allows the user to perform quantum computations on actual quantum devices.

This service is named after the bra-ket (or Dirac) notation of quantum mechanics to represent the state of quantum particles.

AWS Braket gives access to multiple quantum device architectures, from quantum annealers to universal-gate model Quantum Processing Units (QPUs).

AWS Braket includes a Jupyter Notebook development environment as well as a python API interface for the user to **build** quantum circuits and algorithms, **test** them on quantum circuit simulators, and **run** them on different quantum hardware or simulators.

[1] https://aws.amazon.com/braket/

To access the AWS Braket service, one must be signed in to the AWS platform. In the 'All Services' view one can select the Braket service under the section Quantum Technologies.

The AWS Braket service has three main sections or tabs (Figure 1): devices, notebooks and tasks.

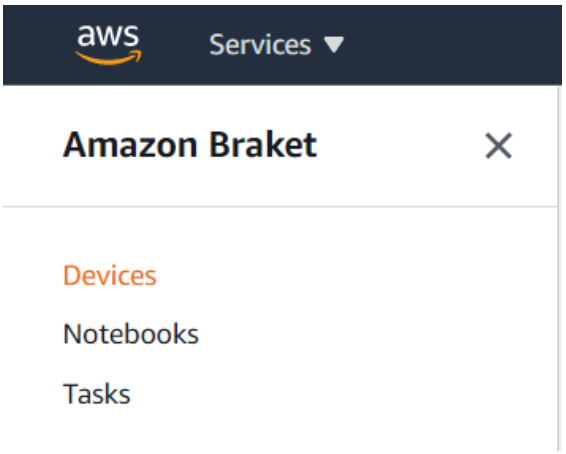

Figure 1: AWS Braket Service tabs.

The 'Devices' tab shows the available QPUs and quantum circuit simulators. At the time of writing, the following devices are included:

- D-Wave Advantage system 1.1: a quantum annealer based on superconducting qubits with 5000+ qubits.
- D-Wave DW 2000Q 6: same as the above, but with 2048 qubits.
- IonQ: Universal gate-model QPU based on trapped ions (11 qubits).
- Rigetti Aspen 8: Universal gate-model QPU based on superconducting qubits (31 qubits).
- Braket SV1 Simulator: a quantum computer simulator with 34 qubits.

The 'Notebooks' tab includes the notebook software development kit (SDK) environment, where the user can create a notebook instance which contains several examples of quantum circuits, algorithms and prototypes for real applications, including well-known algorithms (Grover, Quantum Fourier Transform, etc.), portfolio optimization, molecule simulation, signal noise analysis on a quantum device, etc. The notebook SDK allows for creating new notebooks, cloning the existing ones and running them on quantum devices or simulators.

Finally, the tasks from Notebook or API runs are visible in the 'Tasks' tab. The results of each run are stored (generally in JSON format) into a folder on the S3 storage of the cloud (Amazon Simple Storage Service or Amazon S3).

Example: using the most powerful quantum annealer in the world.

***Disclaimer**: the development and usage of AWS services is not free of charge and associated to costs as described at:*

https://aws.amazon.com/pricing/

Quantum Quants shall not be responsible for any cost or loss whatsoever sustained by any person who relies on the information presented in this article.

One of the most interesting applications of current and future quantum computers is the solution of **optimization problems**.

Several optimization algorithms have been developed for quantum computers, with particular success in solving quadratic optimization problems with binary variables, called

Quadratic Unconstrained Binary Optimization problems (**QUBO**)[2]. Examples of such problems are the "travelling salesman problem" and "the job-shop problem".

The most common quantum computers used for solving QUBO problems are **D-Wave** architectures[3] which are available in the AWS Braket service (both with 2048 and 5760 qubit architectures). This quantum computer uses the property of every physical system to move towards its lowest energy state (think about a mountain stone rolling down to the valley). **Quantum annealing** is a way of using this property in quantum systems to quickly converge to the optimal point of a problem.

In quantum annealing we map a cost function we want to optimize (e.g., a likelihood function) to a quantum computer function (called *potential* or *hamiltonian*). The qubits will then evolve towards the minimum of this function via thermal fluctuations (classical) and quantum fluctuations (tunneling). These two different qubit evolutions are sketched in Figure 2.

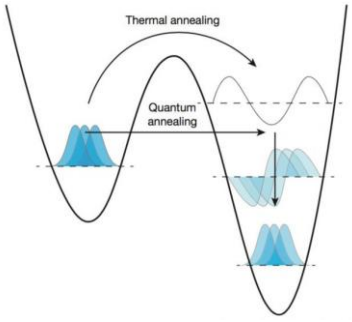

Figure 2: Quantum and Classical annealing evolutions. Source: Biamonte (2017).

[2] These types of problems can be mapped to the well-known Ising model used in physics for describing ferromagnets.

[3] https://www.dwavesys.com/

The additional quantum fluctuations available on a quantum computer improve the performance in spanning the solution space of the optimization problem (i.e. decreases the chance to end up in local minima).

Below, in Box 1, you can see an example of such optimization algorithm. In this case we use the API of AWS Braket to run the code both on a local simulator and on the D-Wave Advantage QPU, which is (at the time of writing) the most powerful quantum annealer with 5760 qubits.

For the sake of simplicity, we choose a very simple optimization problem without an actual practical application. In fact, we try to minimize a simple quadratic function of three variables. In general, when dealing with use cases in real applications, the number of variables involved increases. In these cases, the most difficult task is to define the mapping between the problem space (e.g. the minimization of the risk in a portfolio of financial assets) and the quantum device space (QUBO or Ising-model like).

We can state the problem of Box 1 as: *Find the minimum of the function f of binary variables (x, y, z):*

$$f(x, y, z) = -3x - y + z + 2xy + 6xz + yz$$

Classically, the simplest approach to solve this problem is to evaluate the function in all possible triplets (x, y, z). However, with increasing number of variables, this procedure is not efficient and different algorithms must be used.

Box 1: Find the minimum of a function on D-Wave Advantage in AWS Braket.

```
# import relevant AWS and D'Wave modules
import boto3
from braket.ocean_plugin import BraketDWaveSampler
from dwave.system.composites import EmbeddingComposite
from dwave_qbsolv import QBSolv
import dimod

# The name of the bucket where to store the results.
# Make sure you created this bucket in S3
my_bucket = "YOUR_BUCKET_NAME"
# the name of the folder in the bucket
my_prefix = "YOUR_FOLDER_NAME"

s3_folder = (my_bucket, my_prefix)

# Set sampler to be used for the optimization using BraketSampler
sampler = BraketDWaveSampler(s3_folder, 'arn:aws:braket:::device/qpu/d-
wave/Advantage_system1')

# We now need to create a matrix Q to map the minimization problem of the
# function to the space of the QPU. The mapping procedure is called Embedding.
# The function EmbeddingComposite allows for automatically map the problem
# to the structure of the solver (QPU).
embedded_sampler = EmbeddingComposite(sampler)

# The problem of the function must be mapped to a QUBO functional form,
# i.e. a quadratic function in the variables X=(x,y,z), which in matrix notation
# can be expressed as:
#                 f(x,y,z) = f(X) = X^T.Q.X
#
# Since we deal with binary variables, the linear terms correspond to quadratic
# terms in the same variable, e.g. x = x^2, therefore the main diagonal of
# matrix Q will contain the linear terms, while the off-diagonal terms correspond
# to the mixed variable terms (or in physical terms the "interaction terms"):
# Q = {(x,x):-3, (x,y): 2, (x,z): 6,
#                (y,y):-1, (y,z): 1,
#                          (z,z): 1}
# The matrix Q is clearly symmetric, therefore only one side of the off-diagonal
# terms is shown.
linear = {'q0': -3, 'q1': -1, 'q2': 1} # linear coefficients
quadratic = {('q0', 'q1'): 2, ('q0', 'q2'): 6, ('q1', 'q2'): 1} # quadratic
coefficients

# Define variables for building the binary quadratic model (BQM) for the QUBO.
# - offset: constant energy offset associated with the binary quadratic model.
# - vartype: specify whether the variables are binary or spin.
offset = 0.0
vartype = dimod.BINARY

# Define the BQM
bqm = dimod.BQM(linear, quadratic, offset, vartype)

response = embedded_sampler.sample(bqm)
print('Quantum solution:')
print('samples   = '+str(list(response.samples())))
print('energies = '+str(list(response.data_vectors['energy'])))
print('counts = '+str(list(response.data_vectors['num_occurrences'])))
```

Snippet 1: This code calculates the minimum energy of the system associated to the function f(x,y,z).
The solution is {'x': 1, 'y': 0, 'z': 0} and the function value in this point is -3.

For testing and prototyping, it is recommended to use the local simulator. For more information, please refer to the Ocean documentation of QBSolv (https://docs.ocean.dwavesys.com/projects/qbsolv/en/latest/index.html).

Disclaimer: the development and usage of AWS services is not free of charge and associated to costs as described at: https://aws.amazon.com/pricing/
Quantum Quants shall not be responsible for any cost or loss whatsoever sustained by any person who relies on the information presented in this article.

Conclusions

The AWS Braket service allows for an easy build, test and run of algorithms and circuits on several quantum devices and simulators. The embedding in Python (via notebook SDK and API) allows for using all needed python modules on the side of the quantum computation.

AWS Braket also allows the user to run their code on the most recent quantum annealer, the D-Wave Advantage, which is the (currently) most powerful and connected quantum computer in the world[4].

At Quantum Quants we built several prototypes for practical solutions from finance (e.g. portfolio optimization) to data science and logistics (crane assignment and scheduling).

Currently, we also investigate algorithms and implementations for the calculation of molecules and compounds. Together with the investigation on the AWS Braket platform and on D-Wave architecture, at Quantum Quants we also investigate possibilities on the more versatile universal-gate model architecture and also alternatives to the AWS platform, including the IBM Quantum Experience framework[5] the Google framework of Tensorflow-quantum[6].

About Quantum Quants

Quantum Quants helps businesses to get insights on applications of quantum computing to the financial industry, data science and supply-chain.

Giuseppe Colucci is a PhD in theoretical physics. He is a senior ALM specialist at de Volksbank N.V. and owner of Quantum Quants. He is an expert on quantum theory, data science models and optimization. He developed interest for quantum computing since his Theoretical Physics studies and is currently active in publishing academic papers on applications of quantum computing.

Quantum Quants, Rotterdam (The Netherlands)

email: info.quantumquants@gmail.com

[4] https://www.dwavesys.com/d-wave-two%E2%84%A2-system
[5] https://quantum-computing.ibm.com/
[6] https://www.tensorflow.org/quantum