QCArchive

The QCArchive project enables running, organizing, storing, and sharing large numbers of quantum chemistry computations using a variety of different codes. Via a python and web API, users can programmatically submit, view, manage, and organize large numbers of calculations, and then analyze and share the resulting data.

Large-scale computation and data management

The goal of QCArchive is to make the acquisition and organization of quantum chemistry data easier for users who need large amounts of data.

When running enormous numbers of computations, several pain points arise that QCArchive aims to solve. With QCArchive, a user can submit computations, which are then supervised by the server. If spurious errors arise, computations can be automatically restarted without input required from the user. More serious errors are reported to the user, where they are able to take another action.manually restarting the computation or for errors and restarted In particular, the project sets out to solve the pain points above. In addition, submitting the same calculation twice does not lead to duplicate calculations, reducing clutter and waste from many identical calculations.

The results of the calculations are accessible via the web API and a python package (QCPortal) which the user can interface with common data analysis packages such as Pandas and Matplotlib.

Subprojects

The QCArchive project tackles the complexity and pain points in large quantum chemistry simulation campaigns by means of a few modular projects. Each project can be used independently of the others. In particular, QCSchema, QCElemental, and QCEngine do not require using QCFractal or QCPortal.

QCSchema/QCElemental

The QCSchema project defines a common input & output schema for quantum chemistry computations. This abstracts the details of preparing inputs and consuming outputs of a computation and defines common data types used for calculations such as molecules.

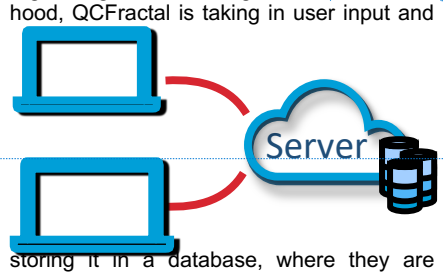
QCSchema is strictly a JSON schema, while QCElemental is a python implementation of the schema. QCElemental also contains useful features, such as parsing many different molecule formats and a utility for converting units.

QCEngine

QCEngine is responsible for actually running a computation. It takes in a QCSchema input, and generates a QCSchema output by running the selected quantum chemistry program. This module allows users to run different computational chemistry programs without requiring them to learn the details of preparing inputs and running computations with each individual code.

QCFractal/QCPortal

QCFractal is a database and web API for submitting and storing computations, and organizing, and sharing data. Under the hood, QCFractal is taking in user input and



storing it in a database, where they are placed in a queue. Distributed workers can

Commented [1]: in order to create datasets for use in ML, ...

Commented [2R1]: It might be better to give examples of use cases in the next part so that there is a bit more room for it. It is important to give them examples, but say that the uses of QCArchive are general.

Commented [3]: Some "cool" graphic here that shows the results of a lot of quantum calculations would be good. (Like the 3D PES type of picture that we discussed)

Commented [4]: All of the sentences in this section could be in one paragraph.

Commented [6]: What is the usefulness of this abstraction? Why should a user care?

Commented [7]: "many" or "a campaign of" computations?

Commented [8]: How much of this under the hood does the user need to know? Think relatively high level.

Commented [5]: It might be useful to also give potential use cases for each of these parts - especially if you want users to use the pieces separately. If this is not a big part of how you see users using QCArchive, then maybe it isn't so important to emphasize each of the parts?

then claim tasks from this queue, and return the results of the computation.

Users can track the progress of a large number of computations, restart errored calculations, and otherwise manage these computations.

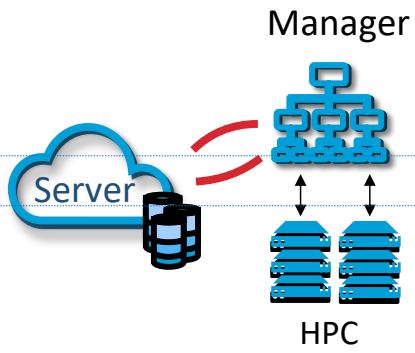
Once the computations are completed, users can then programmatically obtain and analyze the results of these computations.

Relatively homogeneous calculations with common methods, basis sets, or other parameters, can be organized into datasets, which allow submitting and managing these computations as a group rather than individually.

QCPortal is a python package for interacting with a QCFractal server. While it is possible to use a QCFractal server directly via the web API, it is often much easier for users to use this package.

Compute Managers

Managers are responsible for claiming tasks from a QCFractal server, running them, and uploading the results back to the server. Managers work with a pull-based model, where the QCFractal server keeps minimal tracking of the managers. These managers can be run on anything from a laptop to an HPC clusters.



QCArchive is intended for users who are familiar with quantum chemistry – and to some extent quantum chemistry software – and would like to automate calculations to a large degree, and/or to integrate these calculations into an existing workflow.

Deleted: for greater control and abstraction over large numbers of calculations (i.e.,

Deleted:)

Commented [9]: Can these be used independently of QCFractal? If not, then perhaps it should be in that section and should primarily emphasis the ability to run on multiple types of resources.

Commented [10]: Somewhere in here it would be useful to note how many different computational chemistry packages can be used - perhaps with some examples (although probably need to be careful with this). It is probably also useful to emphasize that QCArchive can be extended to work with other quantum packages.