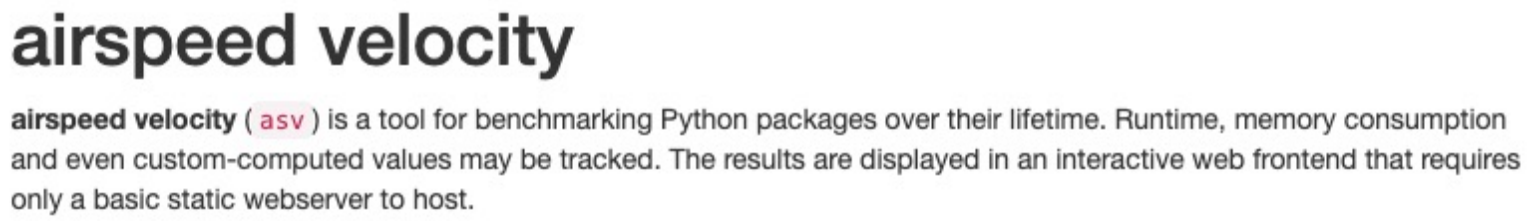
Performance benchmarking in Qiskit Applications Modules

Dariusz Lasecki

Airspeed velocity tool

https://asv.readthedocs.io/en/stable/



asv.conf.json – configuration file

```
    // List of branches to benchmark. If not provided, defaults to "master"
    // (for git) or "default" (for mercurial).
    "branches": ["main"], // for git
    // "branches": ["default"],    // for mercurial

    // The DVCS being used.  If not set, it will be automatically
    // determined from "repo" by looking at the protocol in the URL
    // (if remote), or by looking for special directories, such as
    // ".git" (if local).
    "dvcs": "git",

    // The tool to use to create environments.  May be "conda",
    // "virtualenv" or other value depending on the plugins in use.
    // If missing or the empty string, the tool will be automatically
    // determined by looking for tools on the PATH environment
    // variable.
    "environment_type": "virtualenv",

    // timeout in seconds for installing any dependencies in environment
    // defaults to 10 min
    //"install_timeout": 600,

    // the base URL to show a commit for the project.
    "show_commit_url": "http://github.com/Qiskit/qiskit-nature/commit/",

    // The Pythons you'd like to test against.  If not provided, defaults
    // to the current version of Python used to run `asv`.
    "pythons": ["3.8"],
```

```
{
    // The version of the config file format.  Do not change, unless
    // you know what you are doing.
    "version": 1,

    // The name of the project being benchmarked
    "project": "qiskit-nature",

    // The project's homepage
    "project_url": "https://qiskit.org/documentation/nature/",

    // The URL or local path of the source code repository for the
    // project being benchmarked
    "repo": "https://github.com/Qiskit/qiskit-nature.git",

    // The Python project's subdirectory in your repo.  If missing or
    // the empty string, the project is assumed to be located at the root
    // of the repository.
    // "repo_subdir": "",

    // Customizable commands for building, installing, and
    // uninstalling the project. See asv.conf.json documentation.
    //
    "install_command": [
        "return-code=any python -c \"import shutil; shutil.rmtree('{build_dir}/build')\"",
        "return-code=any python -c \"import shutil; shutil.rmtree('{build_dir}/qiskit_nature.egg-info')",
        "python -mpip install git+https://github.com/Qiskit/qiskit-terra",
        "python -mpip install git+https://github.com/Qiskit/qiskit-aer",
        "python -mpip install {wheel_file}",
    ],
    "uninstall_command": [
        "return-code=any python -mpip uninstall -y {project}",
        "return-code=any python -mpip uninstall -y qiskit-aer qiskit-terra",
    ],
```

```
    // The directory (relative to the current directory) that benchmarks are
    // stored in.  If not provided, defaults to "benchmarks"
    "benchmark_dir": "benchmarks",

    // The directory (relative to the current directory) to cache the Python
    // environments in.  If not provided, defaults to "env"
    "env_dir": ".asv/env",

    // The directory (relative to the current directory) that raw benchmark
    // results are stored in.  If not provided, defaults to "results".
    "results_dir": ".asv/results",

    // The directory (relative to the current directory) that the html tree
    // should be written to.  If not provided, defaults to "html".
    "html_dir": ".asv/html",
```

```python
class ProteinFoldingProblemBenchmarks:
    """Protein Folding Problem benchmarks."""

    version = 1
    params = [
        ["Neuropeptide", "NeuropeptideDummySide", "Angiotensin", "AngiotensinDummySide"],
        ["MiyazawaJerniganInteraction", "RandomInteraction", "MixedInteraction"],
    ]
    param_names = ["peptide", "interaction type"]

    def __init__(self):
        self.peptides = {
            "Neuropeptide": ("APRLRFY", [""] * 7),  #
            "NeuropeptideDummySide": ("APRLRFY", ["", "", "R", "", "T", "W", ""]),
            # Neuropeptide with dummy side chains
            "Angiotensin": ("DRVYIHPFHL", [""] * 10),  # Angiotensin I, human
            "AngiotensinDummySide": (
                "DRVYIHPFHL",
                ["", "", "P", "R", "L", "H", "Y", "", "I", ""],
            ),
        }  # Angiotensin I, human with dummy side chains

        self.interactions = {
            "MiyazawaJerniganInteraction": MiyazawaJerniganInteraction(),
            "RandomInteraction": RandomInteraction(),
            "MixedInteraction": MixedInteraction(),
        }

    def setup(self, peptide_id, interaction_id):
        """setup"""
        qasm_sim = Aer.get_backend("qasm_simulator")
        self._qins = QuantumInstance(backend=qasm_sim, shots=1)
        self.main_chain_residue_sequence = self.peptides[peptide_id][0]
        self.side_chain_residue_sequences = self.peptides[peptide_id][1]
        peptide = Peptide(self.main_chain_residue_sequence, self.side_chain_residue_sequences)
        interaction = self.interactions[interaction_id]
        self.protein_folding_problem = ProteinFoldingProblem(
            peptide, interaction, PenaltyParameters()
        )

    def time_generate_peptide(self, _, __):
        """Time generation of a peptide."""
        return Peptide(self.main_chain_residue_sequence, self.side_chain_residue_sequences)

    def time_generate_full_qubit_operator(self, _, __):
        """Time generation of full protein folding qubit operator."""
        return self.protein_folding_problem._qubit_op_full()

    def time_generate_compressed_qubit_operator(self, _, __):
        """Time generation of compressed protein folding qubit operator."""
        return self.protein_folding_problem.qubit_op()

```

https://github.com/Qiskit/qiskit-app-benchmarks

In order to run benchmarks, run:

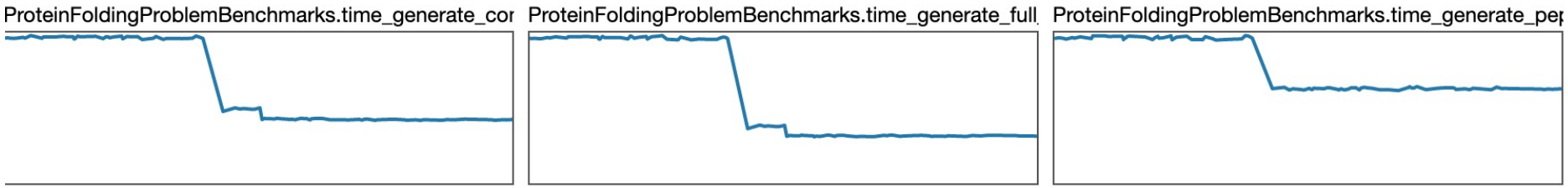
- Finance: `make asv TARGET=finance ASVCMD=run`
- Machine Learning: `make asv TARGET=machine_learning ASVCMD=run`
- Optimization: `make asv TARGET=optimization ASVCMD=run`
- Nature: `make asv TARGET=nature ASVCMD=run`

- https://qiskit.github.io/qiskit-app-benchmarks

Qiskit Application Benchmarks

- Finance
- Machine Learning
- Nature
- Optimization

protein_folding_problem_benchmark



Benchmark graph example