High-Throughput Noncovalent Interactions with Psi4, QCSchema, & QCArchive

LORI A. BURNS

SHERRILL GROUP, GEORGIA TECH

SERMACS 2024, ATLANTA, GA
24 OCTOBER 2024

MolSSI

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

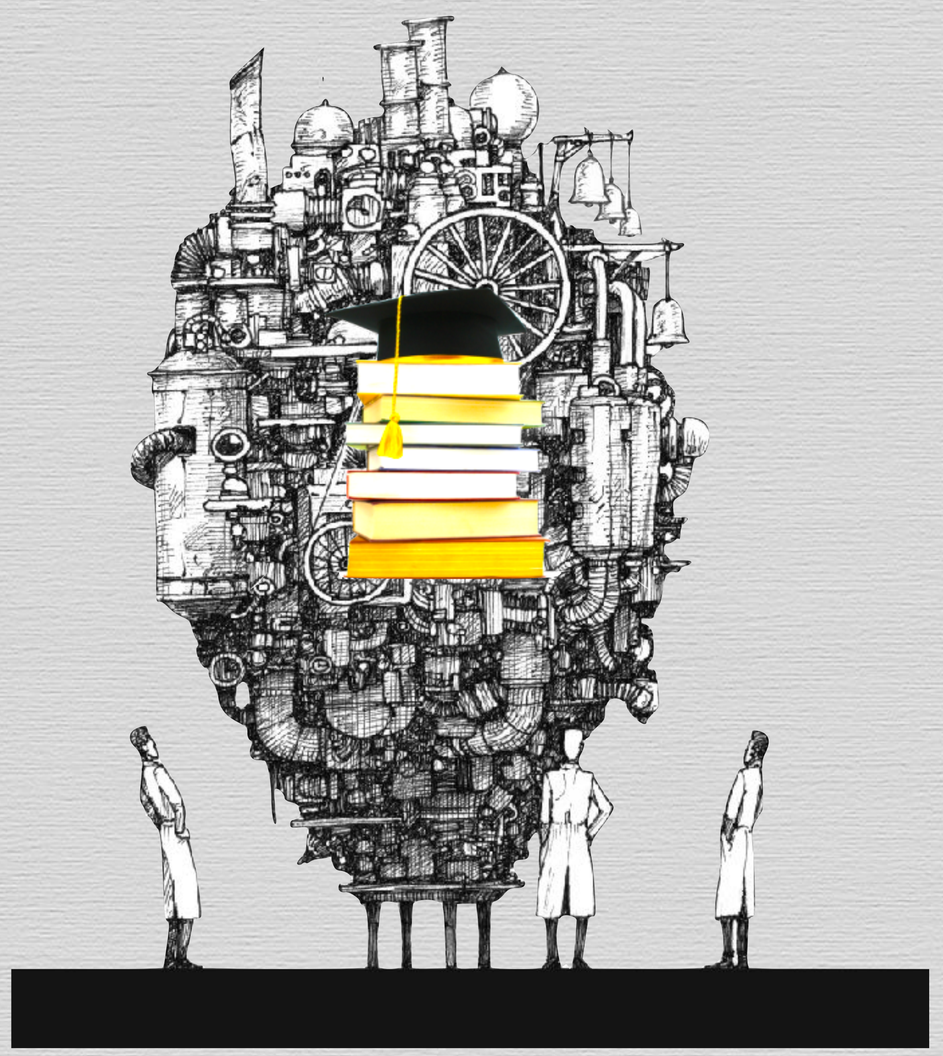
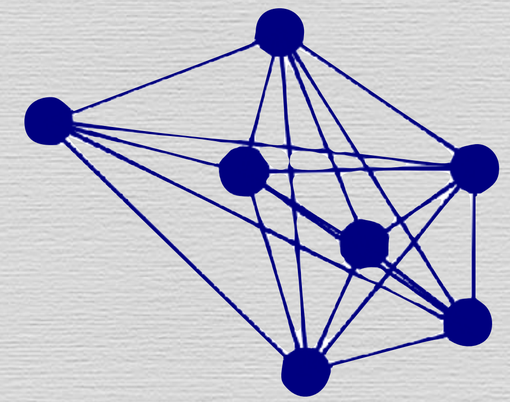


image from https://patientsafe.wordpress.com/2016/05/29/the-healthcare-safety-machine-is-broken/

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

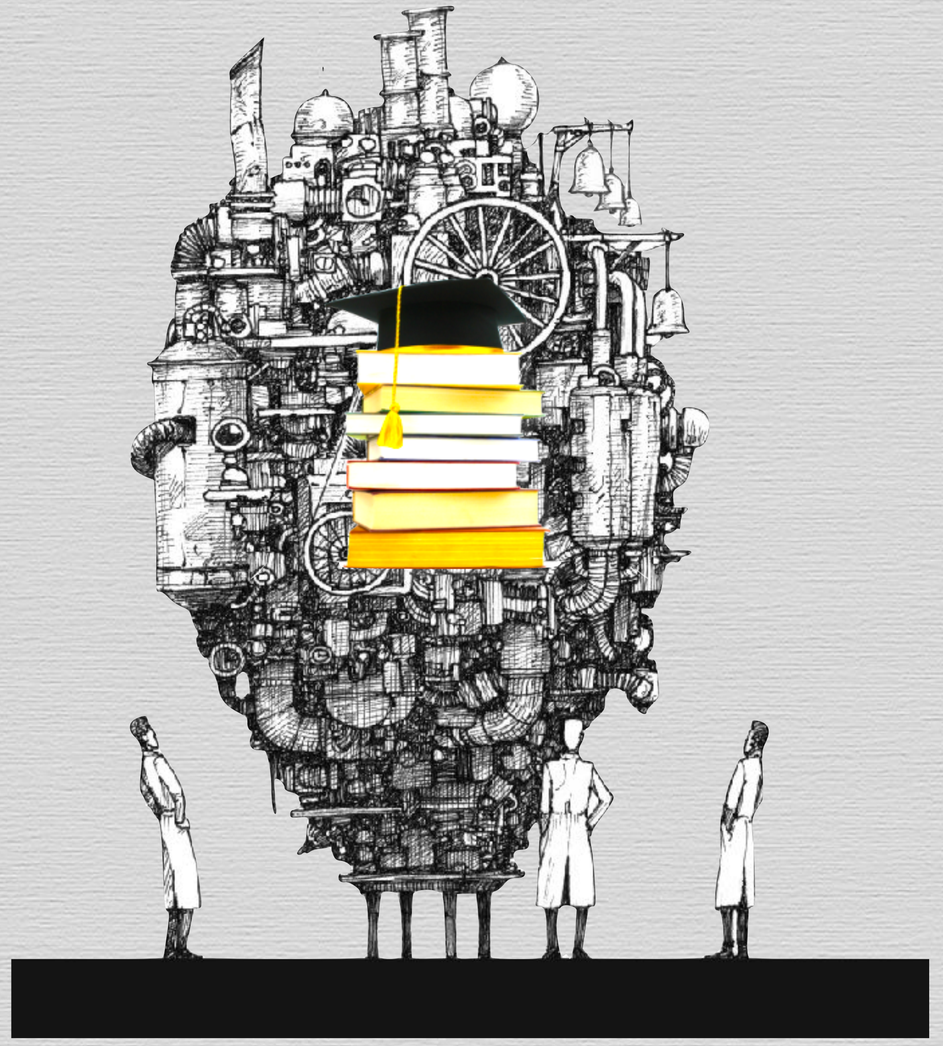
STRONG COUPLING



LOW COHESION

- difficult to maintain, test, read
- Swiss army knife modules

QC PROGRAMS

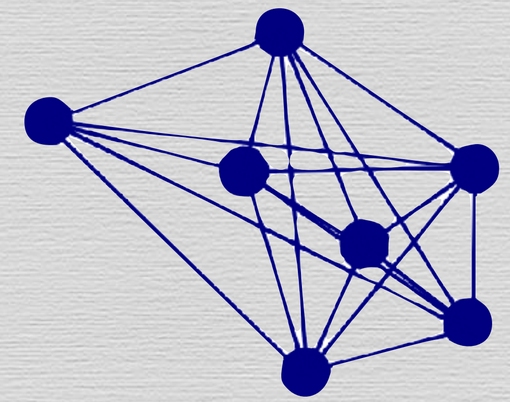


Slide LHS taken from Managing Complexity and Modular Code Development by Radovan Bast

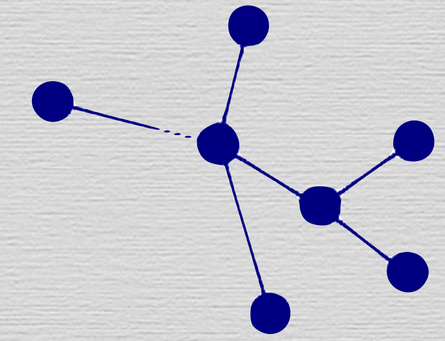
REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

STRONG COUPLING



LOOSE COUPLING



- easier to understand
- easier to compose

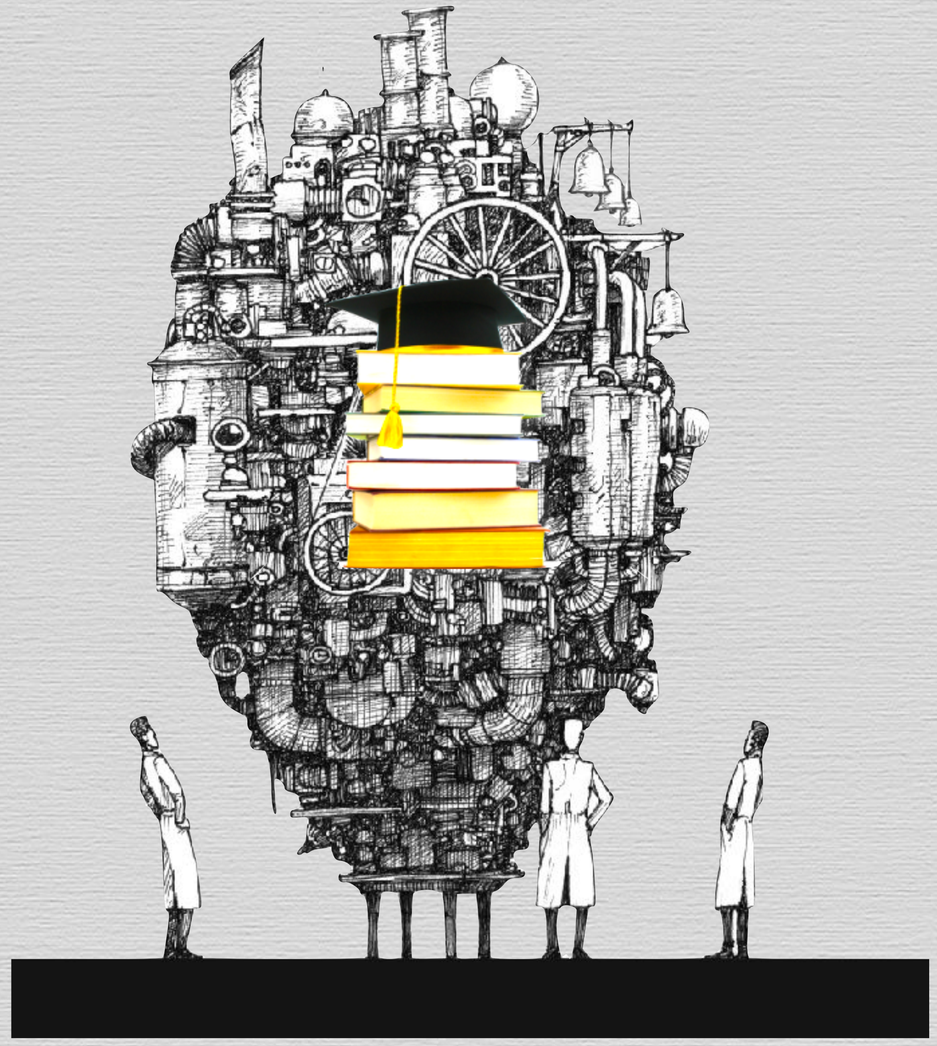
LOW COHESION

- difficult to maintain, test, read
- Swiss army knife modules

HIGH COHESION

- robust, reuseable, understandable
- stable APIs
- do one thing only and do it well (e.g., Unix command line)

QC PROGRAMS



Slide LHS taken from Managing Complexity and Modular Code Development by Radovan Bast

drivers of whole calcs
geometry optimization/IRC • MD driver
crystal lattice energies • torsion driver

processors of whole calcs
finite diff. deriv. • vibrational analysis
many-body decomposition

wfn processors, Molden file comm.
orbital analysis • multipole analysis
natural population analysis

user interface

workhorses
SCF methods • post-SCF methods
method support libraries

separable method components
dispersion corr. • basis set effects

post-SCF, SCF wfn comm.
DMRG/multiref. PT2 • arb.-order CC/CI
active-space CC & CR-CC • ADC mtds
GW • linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & frag. (EFP) potentials
relativistic contrib. • solvation corr.
polarizable embedding

tools
DFT quadrature grids • orbital opt.
symmetry analysis • k-point grid gen.

resources, API or file copy/paste
physical constants • periodic table
basis sets • norm-conserving PP

intensive libs, fast API comm.
$2e^-$ & property integrals
density functionals • grid evaluations
tensor manipulations

non-chem. intensive libs, fast API
linear algebra • binary file format
parallelism frameworks • FFT

software engineering tools
compilers • build systems
version control • testing frameworks
continuous integration services

QC PROGRAMS

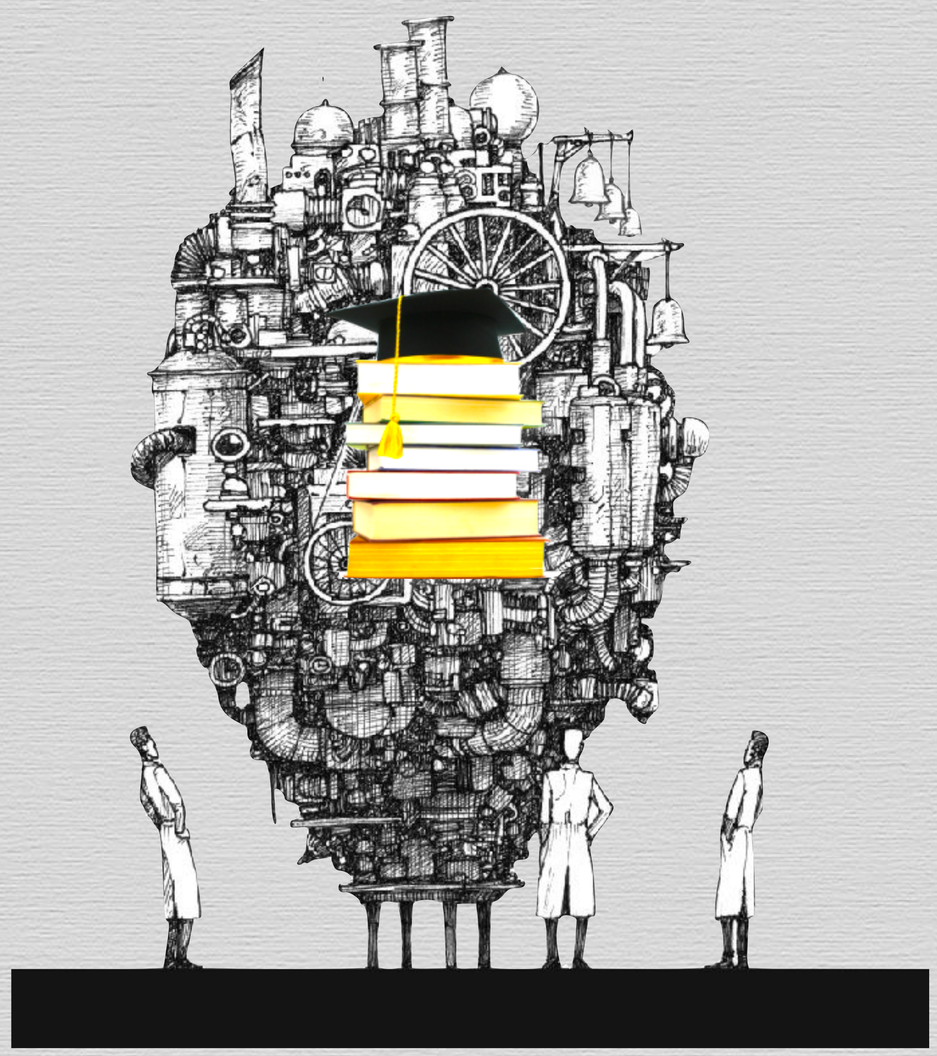


figure from Electronic Structure (2024) chapter (in press)
Blum, Windus, Burns, Oliveira, Pritchard, Slipchenko, et al.

drivers of whole calcs
geometry optimization/IRC • MD driver
crystal lattice energies • torsion driver

processors of whole calcs
finite diff. deriv. • vibrational analysis
many-body decomposition

wfn processors, Molden file comm.
orbital analysis • multipole analysis
natural population analysis

user interface

workhorses
SCF methods • post-SCF methods
method support libraries

separable method components
dispersion corr. • basis set effects

post-SCF, SCF wfn comm.
DMRG/multiref. PT2 • arb.-order CC/CI
active-space CC & CR-CC • ADC mtds
GW • linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & frag. (EFP) potentials
relativistic contrib. • solvation corr.
polarizable embedding

tools
DFT quadrature grids • orbital opt.
symmetry analysis • k-point grid gen.

resources, API or file copy/paste
physical constants • periodic table
basis sets • norm-conserving PP

intensive libs, fast API comm.
$2e^-$ & property integrals
density functionals • grid evaluations
tensor manipulations

non-chem. intensive libs, fast API
linear algebra • binary file format
parallelism frameworks • FFT

software engineering tools
compilers • build systems
version control • testing frameworks
continuous integration services

maximal observed monolithification

QC PROGRAMS

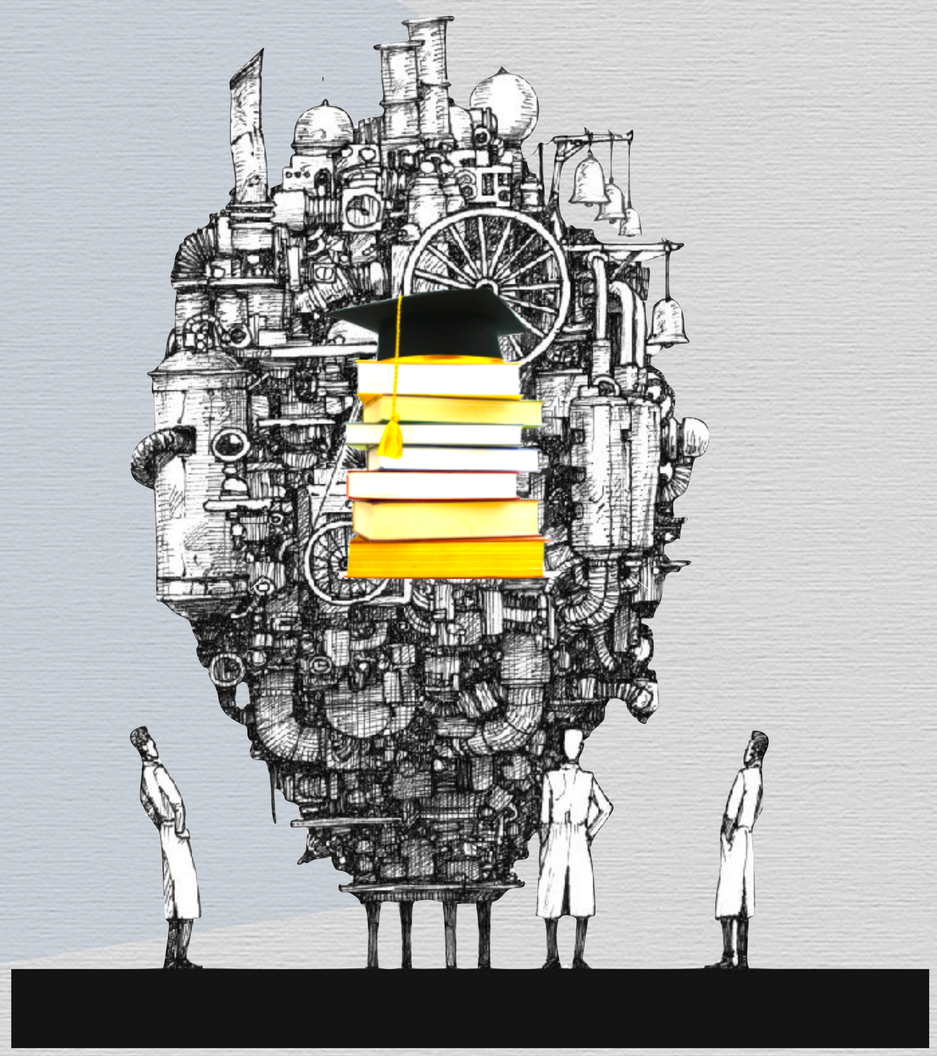


figure from Electronic Structure (2024) chapter (in press)
Blum, Windus, Burns, Oliveira, Pritchard, Slipchenko, et al.

drivers of whole calcs
geometry optimization/IRC • MD driver
crystal lattice energies • torsion driver

processors of whole calcs
finite diff. deriv. • vibrational analysis
many-body decomposition

wfn processors, Molden file comm.
orbital analysis • multipole analysis
natural population analysis

user interface

workhorses
SCF methods • post-SCF methods
method support libraries

separable method components
dispersion corr. • basis set effects

post-SCF, SCF wfn comm.
DMRG/multiref. PT2 • arb.-order CC/CI
active-space CC & CR-CC • ADC mtds
GW • linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & frag. (EFP) potentials
relativistic contrib. • solvation corr.
polarizable embedding

tools
DFT quadrature grids • orbital opt.
symmetry analysis • k-point grid gen.

resources, API or file copy/paste
physical constants • periodic table
basis sets • norm-conserving PP

intensive libs, fast API comm.
$2e^-$ & property integrals
density functionals • grid evaluations
tensor manipulations

non-chem. intensive libs, fast API
linear algebra • binary file format
parallelism frameworks • FFT

software engineering tools
compilers • build systems
version control • testing frameworks
continuous integration services

maximal observed monolithification

minimal desired monolithification

QC PROGRAMS

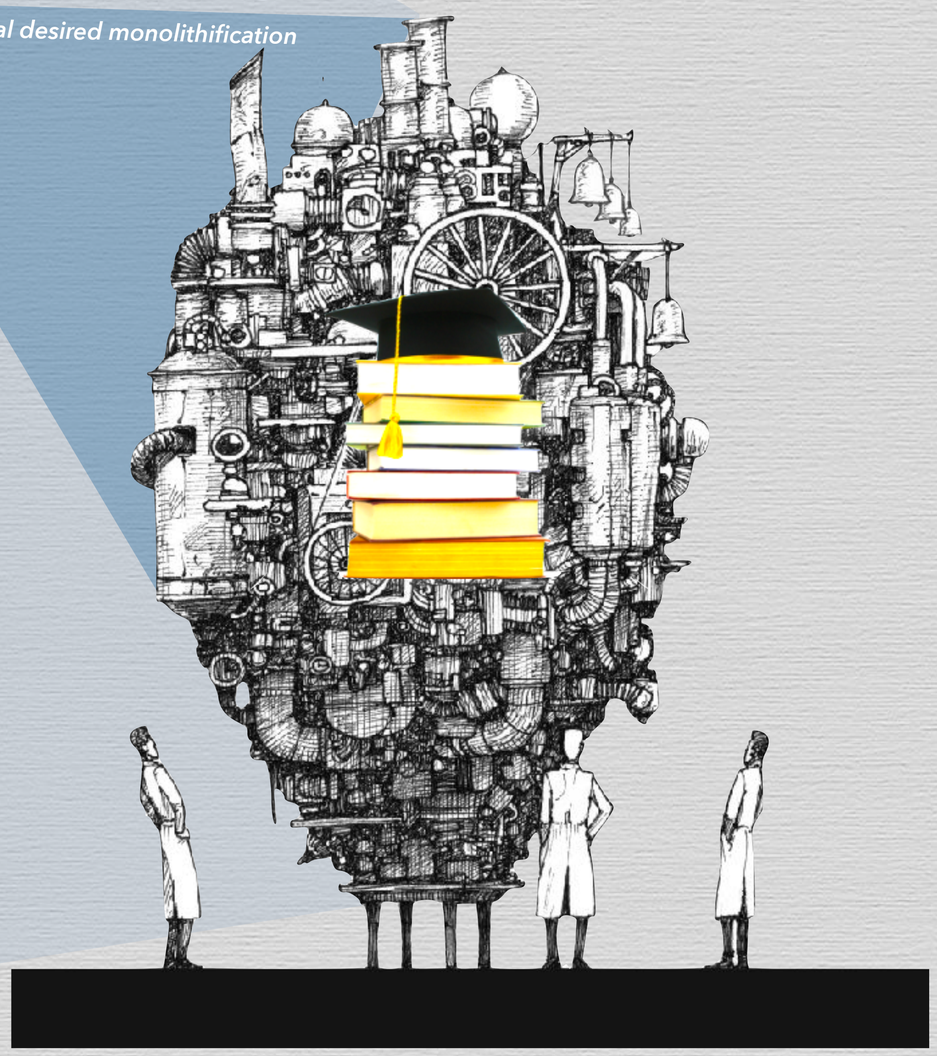
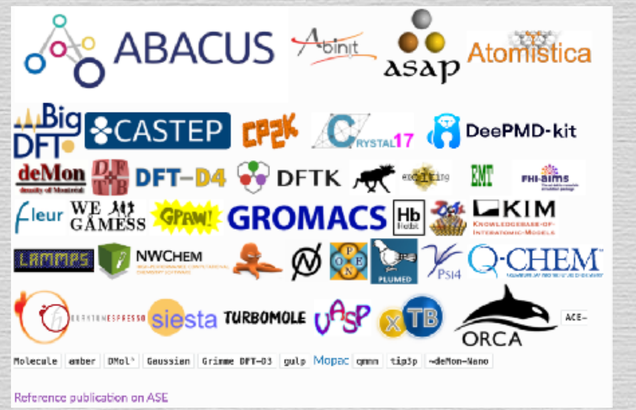


figure from Electronic Structure (2024) chapter (in press)
Blum, Windus, Burns, Oliveira, Pritchard, Slipchenko, et al.



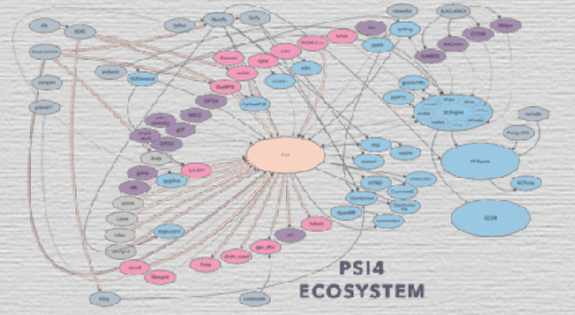


figure from Electronic Structure (2024) chapter (in press)
Blum, Windus, Burns, Oliveira, Pritchard, Slipchenko, et al.

drivers of whole calcs
geometry optimization/IRC • MD driver
crystal lattice energies • torsion driver

processors of whole calcs
finite diff. deriv. • vibrational analysis
many-body decomposition

wfn processors, Molden file comm.
orbital analysis • multipole analysis
natural population analysis

user interface

workhorses
SCF methods • post-SCF methods
method support libraries

separable method components
dispersion corr. • basis set effects

post-SCF, SCF wfn comm.
DMRG/multiref. PT2 • arb.-order CC/CI
active-space CC & CR-CC • ADC mtds
GW • linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & frag. (EFP) potentials
relativistic contrib. • solvation corr.
polarizable embedding

tools
DFT quadrature grids • orbital opt.
symmetry analysis • k-point grid gen.

resources, API or file copy/paste
physical constants • periodic table
basis sets • norm-conserving PP

intensive libs, fast API comm.
2e⁻ & property integrals
density functionals • grid evaluations
tensor manipulations

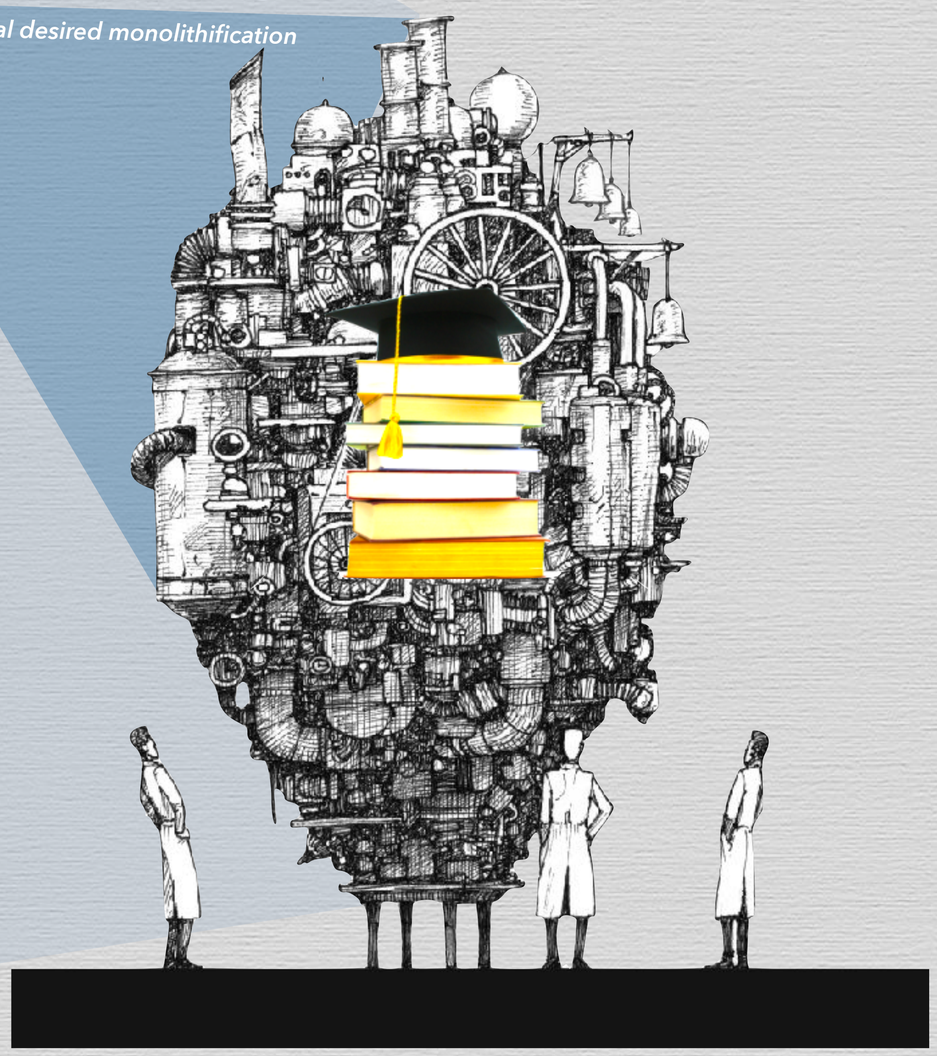
non-chem. intensive libs, fast API
linear algebra • binary file format
parallelism frameworks • FFT

software engineering tools
compilers • build systems
version control • testing frameworks
continuous integration services

maximal observed monolithification

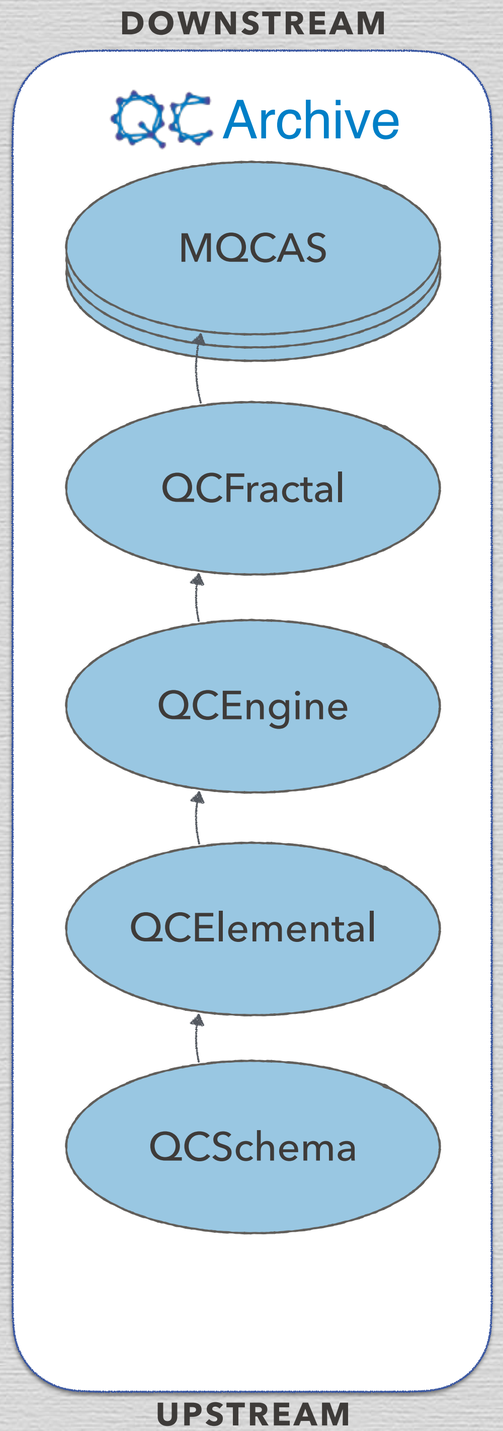
minimal desired monolithification

QC PROGRAMS



QCArchive STACK

qcarchive.molssi.org/



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- database storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

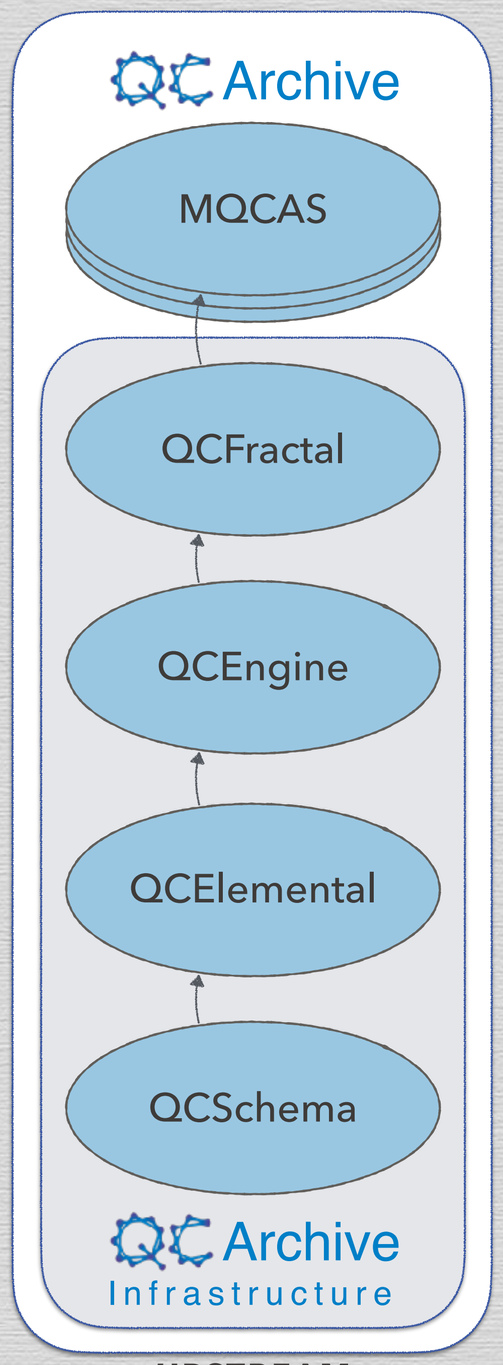
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

QCArchive STACK

qcarchive.molssi.org/

DOWNSTREAM



UPSTREAM

- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

github.com/MolSSI/QCFractal

pip install qcfractal

conda install qcfractal -c conda-forge

- batch (parallel) compute setup & management
- database storage & query of QC results

github.com/MolSSI/QCEngine

pip install qcengine

conda install qcengine -c conda-forge

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

github.com/MolSSI/QCElemental

pip install qcelemental

conda install qcelemental -c conda-forge

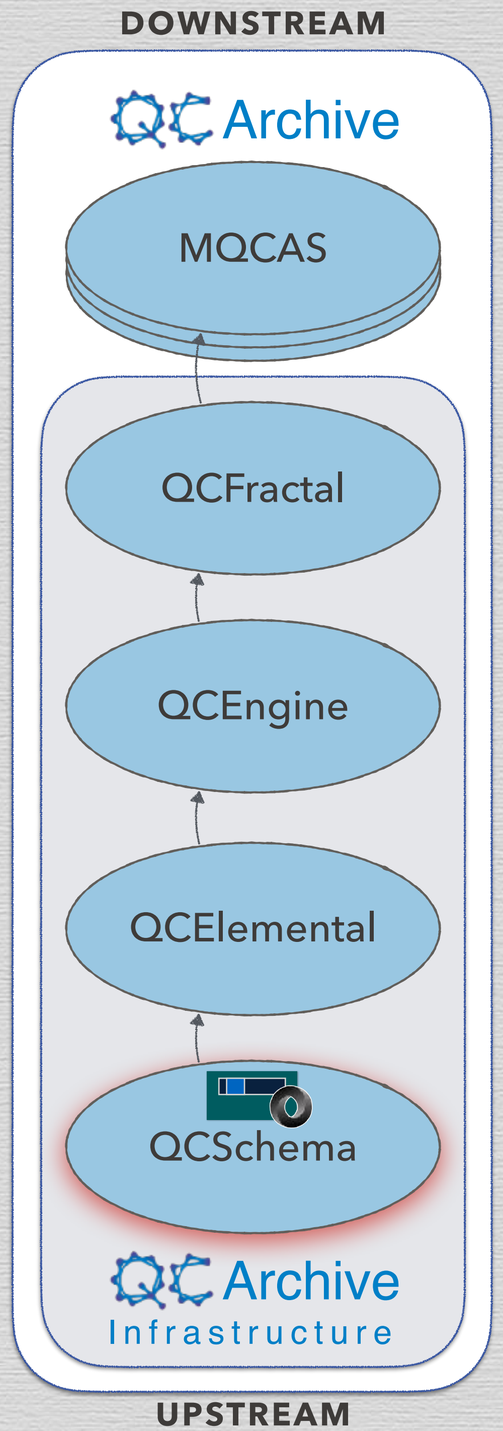
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

github.com/MolSSI/QCSchema

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

QCArchive STACK

qcarchive.molssi.org/



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

github.com/MolSSI/QCFractal

pip install qcfractal

conda install qcfractal -c conda-forge

- batch (parallel) compute setup & management
- database storage & query of QC results

github.com/MolSSI/QCEngine

pip install qcengine

conda install qcengine -c conda-forge

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

github.com/MolSSI/QCElemental

pip install qcelemental

conda install qcelemental -c conda-forge

- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

github.com/MolSSI/QCSchema

Don't visit this repository, as it's out-of-date.

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

QUANTUM CHEMISTRY SCHEMA

github.com/MolSSI/QCSchema

- **COMMUNICATION** channel between all pieces of the ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

QUANTUM CHEMISTRY SCHEMA

github.com/MolSSI/QCSchema

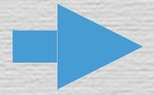
- **COMMUNICATION** channel between all pieces of the ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soo
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

```
O 0 0 0
H 2 0 0
--
@22Ne 5 0 0
units bohr
```

Snippet 1:




```
{"atom_labels": ["", "", ""],
 "atomic_numbers": [ 8,  1, 10],
 "fix_com": False,
 "fix_orientation": False,
 "fragment_charges": [0.0, 0.0],
 "fragment_multiplicities": [2, 1],
 "fragments": [[0, 1], [2]],
 "geometry": [[0., 0., 0.],
       [2., 0., 0.],
       [5., 0., 0.]],
 "mass_numbers": [16,  1, 22],
 "masses": [15.99491462,  1.00782503, 21.99138511],
 "molecular_charge": 0.0,
 "molecular_multiplicity": 2,
 "name": "HNeO",
 "provenance": {"creator": "QCElemental",
                "routine": "qcelemental.molparse.from_schema",
                "version": "v0.8.0"},
 "real": [ True,  True, False],
 "schema_name": "qcschema_molecule",
 "schema_version": 2,
 "symbols": ["O", "H", "Ne"],
 "validated": True}
```


Snippet 2: QCSchema Molecule from Snippet 1.

QUANTUM CHEMISTRY SCHEMA

github.com/MolSSI/QCSchema

- **COMMUNICATION** channel between all pieces of the ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

AtomicInput

```
{
   "molecule": {
        "geometry": [0, 0, 0, 0, 0, 1],
        "symbols": ["He", "He"]
    },
    "driver": "energy",
    "model": {
        "method": "CCSD(T)",
        "basis": "aug-cc-pVDZ",
    },
    "keywords": {},
}
```

QUANTUM CHEMISTRY SCHEMA

github.com/MolSSI/QCSchema

- **COMMUNICATION** channel between all pieces of the ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

AtomicInput

```
{
   "molecule": {
        "geometry": [0, 0, 0, 0, 0, 1],
        "symbols": ["He", "He"]
    },
    "driver": "energy",
    "model": {
        "method": "CCSD(T)",
        "basis": "aug-cc-pVDZ",
    },
    "keywords": {},
}
```

AtomicResult

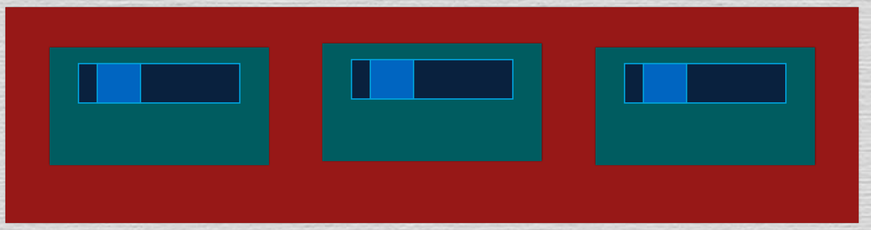
```
{
    ... AtomicInput ...
    "provenance": {
        "creator": "My QM Program",
        "version": "1.1rc1",
    },
    "properties": {
        "calcinfo_nalpha": 5,
        "scf_total_energy": -5.433191881443323,
        "nuclear_repulsion_energy": 2.11670883436,
        "scf_iterations": 8.0,
    },
    "error": null,
    "return_result": -6.5432123456,
    "success": true,
    "stdout": "...",
    "wavefunction": "..."
}
```

QUANTUM CHEMISTRY SCHEMA

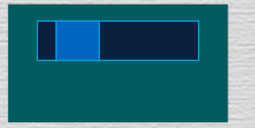
github.com/MolSSI/QCSchema

- **COMMUNICATION** channel between all pieces of the ecosystem.
- **COMMUNITY** project useful for many aspects of quantum chem.
- **JSON** nominally, but any key/value/array language like MessagePack/BSON/XML/YAML ok.
- **TIMELINE** stable since 2019. Most changes for QCA. More community participation desired but little momentum. v2 soon.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.

OptimizationResult



AtomicResult



Molecule

```
{
   "geometry": [0, 0, 0, 0, 0, 1],
   "symbols": ["He", "He"],
    ...
}
```

AtomicInput

```
{
   "molecule": {
        "geometry": [0, 0, 0, 0, 0, 1],
        "symbols": ["He", "He"]
    },
    "driver": "energy",
    "model": {
        "method": "CCSD(T)",
        "basis": "aug-cc-pVDZ",
    },
    "keywords": {},
}
```

```
{
    ... AtomicInput ...
    "provenance": {
        "creator": "My QM Program",
        "version": "1.1rc1",
    },
    "properties": {
        "calcinfo_nalpha": 5,
        "scf_total_energy": -5.433191881443323,
        "nuclear_repulsion_energy": 2.11670883436,
        "scf_iterations": 8.0,
    },
    "error": null,
    "return_result": -6.5432123456,
    "success": true,
    "stdout": "...",
    "wavefunction": "..."
}
```

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

What best describes this – Atomic, SinglePoint, other?

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

What best describes this – Atomic, SinglePoint, other?

AtomicInput

schema_version: 1

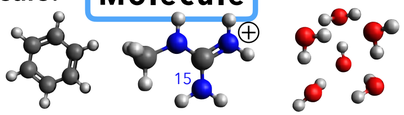
provenance: creator, version, & RT info

`QCElemental, 0.28.0`

id: tracker for databases

extras: free-form storage

molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

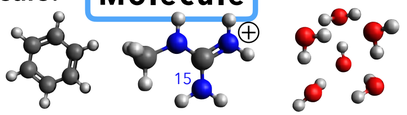
What best describes this – Atomic, SinglePoint, other?

AtomicResultProtocols

stdout: save text output
wavefunction: save orbital info
native_files: save raw prog. files
error_correction: auto-edit&rerun

AtomicInput

schema_version: 1
provenance: creator, version, & RT info
`QCElemental, 0.28.0`
id: tracker for databases
extras: free-form storage
molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF
method: B3LYP-D3, CCSD(T)
basis: 6-31G*, cc-pVQZ
keywords: knobs in SP program
`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

What best describes this – Atomic, SinglePoint, other?

AtomicResult

AtomicInput

schema_version: 1

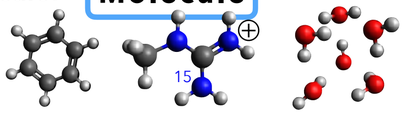
provenance: creator, version, & RT info

`QCElemental, 0.28.0`

id: tracker for databases

extras: free-form storage

molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy**
- **gradient**
- **Hessian**
- **properties**

E *G* *H*

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True

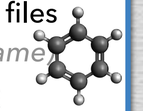
provenance: *(overwrites input)*

`xtb, 1.0.0, psinet, CPU E5-2630`

stdout/stderr: program's text output for ppl

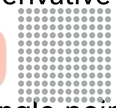
native_files: requested raw program files

molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative

E · *G* ⋮ *H*



properties: key results for single-point

AtomicResultProperties

wavefunction: orbital-level results

WavefunctionProperties

QCSchema ATOMIC

analytical single-point computations


```
'properties': {'calcinfo_nalpha': 10,
               'calcinfo_natom': 2,
               'calcinfo_nbasis': 28,
               'calcinfo_nbeta': 10,
               'calcinfo_nmo': 28,
               'nuclear_repulsion_energy': 16.666666666666668,
               'return_energy': -256.977621071098,
               'return_gradient': array([[0,  0, -6.54470201e-05],
                                         [0,  0,  6.54470212e-05]]),
               'return_hessian': None,
               'scf_dipole_moment': array([0,  0, 4.97379915e-14]),
               'scf_iterations': 5,
               'scf_one_electron_energy': -398.5650589792168,
               'scf_total_energy': -256.977621071098,
               'scf_total_gradient': array([[0,  0, -6.54470201e-05],
                                            [0,  0,  6.54470212e-05]]),
               'scf_total_hessian': None,
               'scf_two_electron_energy': 124.92077124145213},

'extras': {'qcvars': {'CURRENT DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]),
                      'CURRENT ENERGY': -256.977621071098,
                      'CURRENT GRADIENT': array([[0,  0, -6.54470201e-05],
                                                 [0,  0,  6.54470212e-05]]),
                      'CURRENT REFERENCE ENERGY': -256.977621071098,
                      'DD SOLVATION ENERGY': 0.0,
                      'HF KINETIC ENERGY': 256.97713564856093,
                      'HF POTENTIAL ENERGY': -513.9547567196589,
                      'HF TOTAL ENERGY': -256.977621071098,
                      'HF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                  [0,  0,  6.54470212e-05]]),
                      'HF VIRIAL RATIO': 2.000001888971701,
                      'NUCLEAR REPULSION ENERGY': 16.666666666666668,
                      'ONE-ELECTRON ENERGY': -398.5650589792168,
                      'PCM POLARIZATION ENERGY': 0.0,
                      'PE ENERGY': 0.0,
                      'SCF DIPOLE': array([0.00000000e+00, 0.00000000e+00, 4.97379915e-14]
                      'SCF ITERATION ENERGY': -256.977621071098,
                      'SCF ITERATIONS': 5.0,
                      'SCF TOTAL ENERGIES': array([-256.97759612, -256.97762005, -256.9776
                                                   -256.97762107, -256.97762107, -256.9776
                      'SCF TOTAL ENERGY': -256.977621071098,
                      'SCF TOTAL GRADIENT': array([[0,  0, -6.54470201e-05],
                                                   [0,  0,  6.54470212e-05]]),
                      'TWO-ELECTRON ENERGY': 124.92077124145213}},
```


AtomicResultProperties (abr.)

calcinfo_natom
calcinfo_nmo
nuclear_repulsion_energy
return_energy: *E*
return_gradient: *G*

scf_total_*egh*: ref. energy, gradient, etc.
***mtd*_correlation_energy:** post-scf energy
***mtd*_iterations:** count to convergence
***mtd*_mpole_moment:** dipole etc. array

AtomicResult

AtomicInput

schema_version: 1

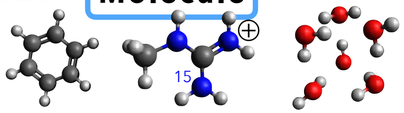
provenance: creator, version, & RT info

`QCElemental, 0.28.0`

id: tracker for databases

extras: free-form storage

molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- energy
- gradient
- Hessian
- properties

E *G* *H*

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True

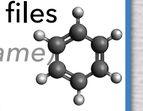
provenance: *(overwrites input)*

`xtb, 1.0.0, psinet, CPU E5-2630`

stdout/stderr: program's text output for ppl

native_files: requested raw program files

molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative

E · *G* ⋮⋮ *H* ▦

properties: key results for single-point

AtomicResultProperties

wavefunction: orbital-level results

WavefunctionProperties

QCSchema ATOMIC

analytical single-point computations

- **AtomicInput** designed as a job directive to express any **SINGLE-GEOMETRY** analytic derivative or property calc. (as defined by `driver`) among CMS workhorses
- in practice, it can hide more complicated computations (e.g., finite difference frequencies), and advanced Result schema often seemingly inherit from **AtomicResult**
- note that QCSchema controls **LAYOUT** and "manages" values for **Molecule** and `driver` with model and enum respectively. All other values are free-form strings in CMS native syntax. See "Data Layouts" slide later.
- **AtomicResult APPENDS** to **AtomicInput** with output, simple scalar and array results, and `wavefunction` results.
- `provenance` tracks who wrote the data – program version, module, compute information
- note **FLAT** structure where Result inherits from Input
- overwhelmingly the most used of the calculation schema. These are stored by the **MILLIONS** at MolSSI
- Called "Atomic" for "fundamental"

What best describes this – Atomic, SinglePoint, other?

AtomicResult

AtomicInput

schema_version: 1

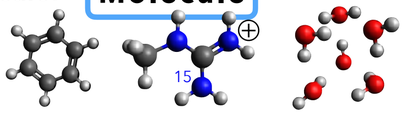
provenance: creator, version, & RT info

`QCElemental, 0.28.0`

id: tracker for databases

extras: free-form storage

molecule: **Molecule**



protocols: customize return layout

AtomicResultProtocols

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

success: True

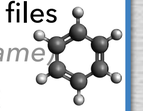
provenance: *(overwrites input)*

`xtb, 1.0.0, psinet, CPU E5-2630`

stdout/stderr: program's text output for ppl

native_files: requested raw program files

molecule: *(overwrites inp w/result frame)*



return_result: single-point derivative

E · *G* ⋮⋮ *H*

properties: key results for single-point

AtomicResultProperties

wavefunction: orbital-level results

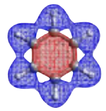
WavefunctionProperties

WavefunctionProperties (abr.)

basis: **BasisSet**

restricted: alpha == beta

orbitals_a/b



density_a/b

fock_a/b

eigenvalues_a/b

occupations_a/b

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

⟨CC|CC⟩

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

GAMESS

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

PSI4

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

NWChem

```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

QCSchema: AtomicInput

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

```
                                    ⟨CC|CC⟩
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

```
$ccinp ncore=0 $end                  GAMESS
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

```
molecule {                           PSI4
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

```
geometry units bohr                  NWChem
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

DATA LAYOUT TRANSLATION

QCSchema: AtomicInput

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {




                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {




                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {




                        }}
```

```
{ 'molecule':     ,
  'driver':         ,
  'model': {
    'method':       ,
    'basis':              },
  'keywords': {




                        }}
```

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

⟨CC|CC⟩

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

GAMESS

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

Psi4

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

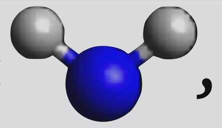
NWChem

```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

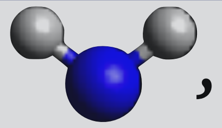
DATA LAYOUT TRANSLATION

MOL & DRIVER STANDARDIZATION

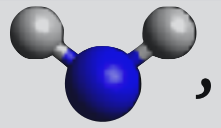
QCSchema: AtomicInput



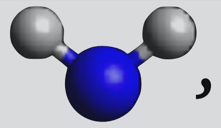
```
{ 'molecule':     ,
  'driver': 'energy',
  'model': {
    'method':         ,
    'basis':              },
  'keywords': {

                          }}
```



```
{ 'molecule':     ,
  'driver': 'energy',
  'model': {
    'method':         ,
    'basis':              },
  'keywords': {

                          }}
```



```
{ 'molecule':     ,
  'driver': 'energy',
  'model': {
    'method':         ,
    'basis':              },
  'keywords': {

                          }}
```



```
{ 'molecule':     ,
  'driver': 'energy',
  'model': {
    'method':         ,
    'basis':              },
  'keywords': {

                          }}
```

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more; QCEngine will not standardize input

DOMAIN-SPECIFIC ASCII

all-electron restricted-open-shell CCSD/aug-cc-pVDZ energy of NH_2 molecule

```
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014

*CFOUR(REFERENCE=ROHF
BASIS=AUG-PVDZ,CALC_LEVEL=CCSD
COORDINATES=CARTESIAN,UNITS=BOHR
CHARGE=0,MULTIPLICITY=2)
```

```
$ccinp ncore=0 $end
$basis gbasis=accd $end
$contrl cctyp=ccsd coord=prinaxis
 icharg=0 ispher=1 mult=2
 runtyp=energy scftyp=rohf
 units=bohr $end
$data

C1
N 7  0.000  0.000 -0.146
H 1  0.000 -1.511  1.014
H 1  0.000  1.511  1.014
$end
```

```
molecule {
0 2
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
units au
}

set reference rohf

energy('ccsd/aug-cc-pvdz')
```

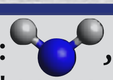
```
geometry units bohr
N  0.000  0.000 -0.146
H  0.000 -1.511  1.014
H  0.000  1.511  1.014
end
charge 0
basis spherical
  h library aug-cc-pvdz
  n library aug-cc-pvdz
end
scf
  rohf
  nopen 1
end
tce
  ccsd
end
task tce energy
```

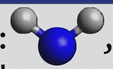
DATA LAYOUT TRANSLATION

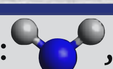
MOL & DRIVER STANDARDIZATION

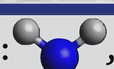
BASIS & KEYWORDS STANDARDIZATION

QCSchema: AtomicInput

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-pvdz'    },
  'keywords': {
    'reference': 'rohf'

                          }}
```

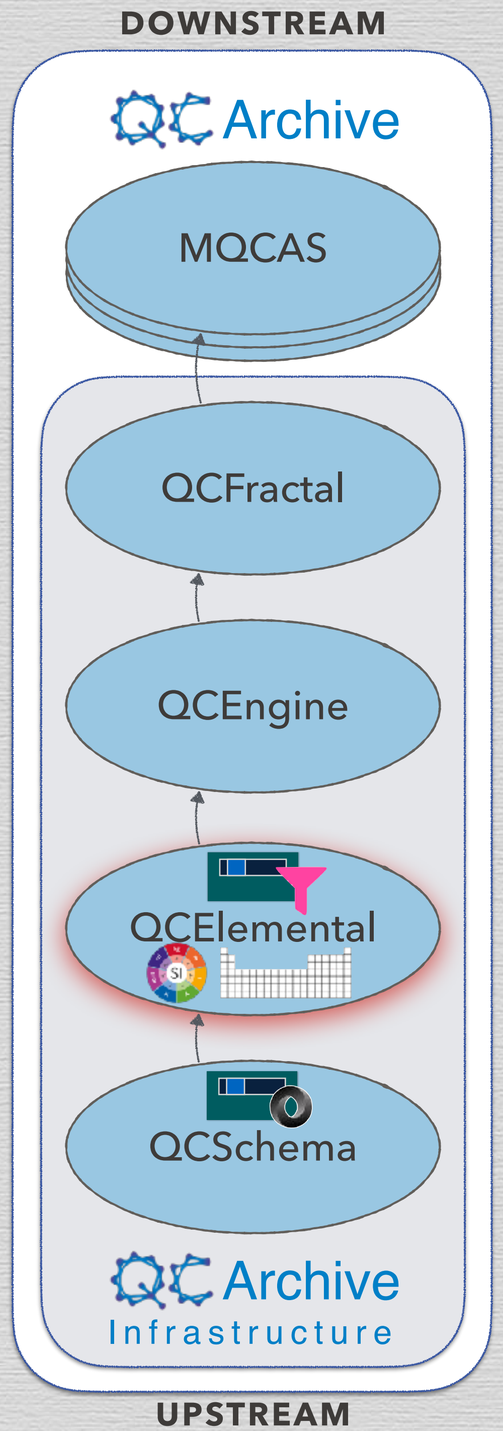
```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method':'ccsd' ,
    'basis': 'accd'        },
  'keywords': {
    'contrl__ispher': 1,
    'contrl__scftyp': 'rohf',
    'ccinp__ncore': 0      }}
```

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-cc-pvdz'},
  'keywords': {
    'reference': 'rohf'

                          }}
```

```
{ 'molecule':  ,
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'aug-cc-pvdz'},
  'keywords': {
    'basis__spherical': True,
    'scf__rohf': True,
    'qc_module': 'tce'     }}
```

QCArchive STACK

qcarchive.molssi.org/



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- database storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

QCElemental implements QCSchema

with strong validation through `pydantic`

SIX FLAWED MOLECULES

```
> Molecule(geometry=[5, 6], symbols=['O'])
ValueError: cannot reshape array of size 2 into shape (3)

> Molecule(geometry=[4, 4, 4], symbols=['Z'])
NotAnElementError: Z

> Molecule(geometry=[0, 1, 2, 3, 4, 5], symbols=['H', 'H', 'H'])
ValidationError: dropped atoms! nat = 3 != 2

> Molecule(geometry=[0, 0.001, 0, 0, 0, 0], symbols=['N', 'N'])
ValidationError: Following atoms are too close: [(0, 1, 0.001)]

> Molecule(geometry=[0, 0, 0], symbols=['He'], molecular_charge=0, molecular_multiplicity=2)
ValidationError: Inconsistent or unspecified chg/mult: sys chg: 0, frag chg: [None], sys mult: 2,
frag mult: [None]

> Molecule(geometry=[0, 0, 0], symbols=['He'], fragments=[[0], [1]])
ValidationError: dropped atoms! nat = 2 != 1
```

THAT RAISE HELPFUL ERRORS IN QCElemental

QCElemental provides datasets

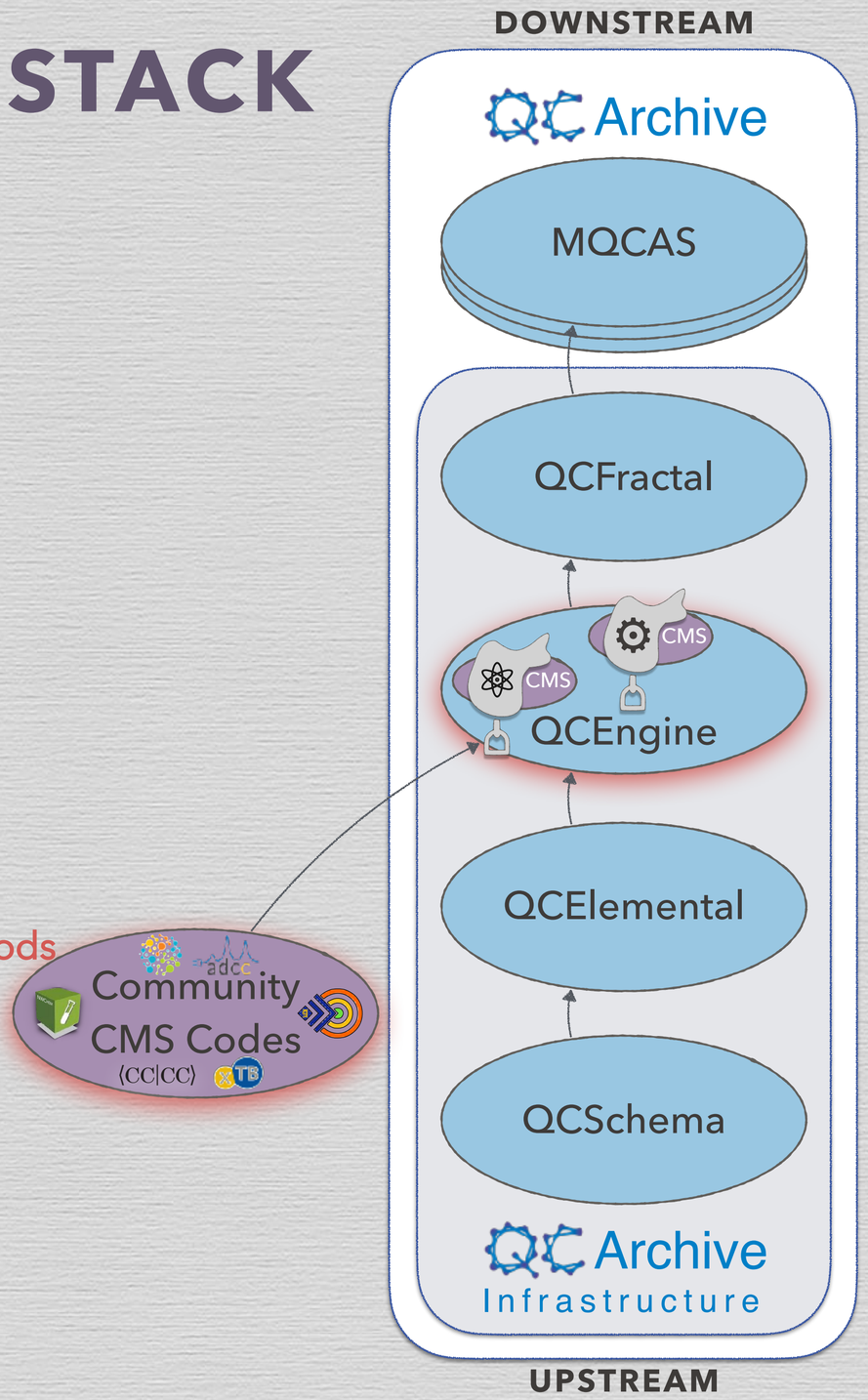
NIST CODATA, covalent and vdW radii, Periodic Table

- **STRUCTURED DATA** processed into light Python API directly from NIST structured data, if available.
- **PRECISION** original significant figures value always available.
- **VERSIONED** when reproducibility demands old versions, accessing both helps transitions.
- **EXPORTS** to C and Fortran headers.
- **CONVERSION** with `pint` module, internally consistent unit conversions available.
- **STORAGE/USER UNITS** maintain AU in QCSchema, while presenting results in customary units.

```
> import qcelemental as qcel
> qcel.constants.Hartree_energy_in_eV
27.21138602
> pc = qcel.constants.get('hartree ENERGY in ev', return_tuple=True)
> pc.label
'Hartree energy in eV'
> pc.data
Decimal('27.21138602')
> pc.units
'eV'
> pc.comment
'uncertainty=0.000 000 17'
> qcel.constants.conversion_factor("bohr", "miles")
3.2881547429884475e-14
```

```
> import qcelemental as qcel
> qcel.periodictable.to_E('KRYPTON')
'Kr'
> qcel.periodictable.to_element(36)
'Krypton'
> qcel.periodictable.to_Z('kr84')
36
> qcel.periodictable.to_A('Kr')
84
> qcel.periodictable.to_mass('Kr86')
85.9106106269
```

QCArchive STACK



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- database storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

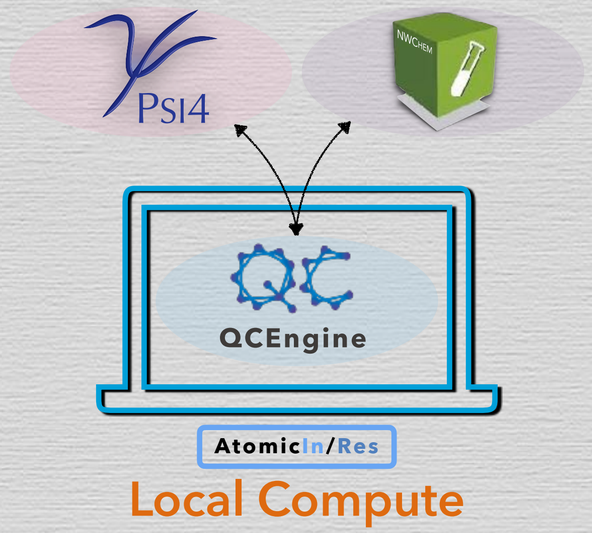
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

- the difficult part – coded QC methods
- structured output uncommon
- DSL input; API/schema uncommon

QCENGINE: PYTHON QCSCHEMA RUNNER

github.com/MolSSI/QCEngine



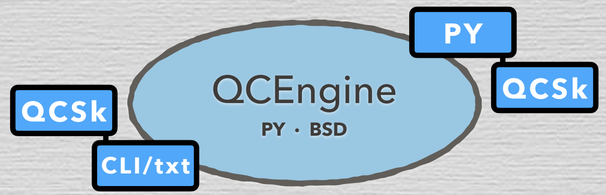
CMDLINE

```
                  cfour
> qcengine run    molpro  atomicinput
                  psi4
                  dftd4
```

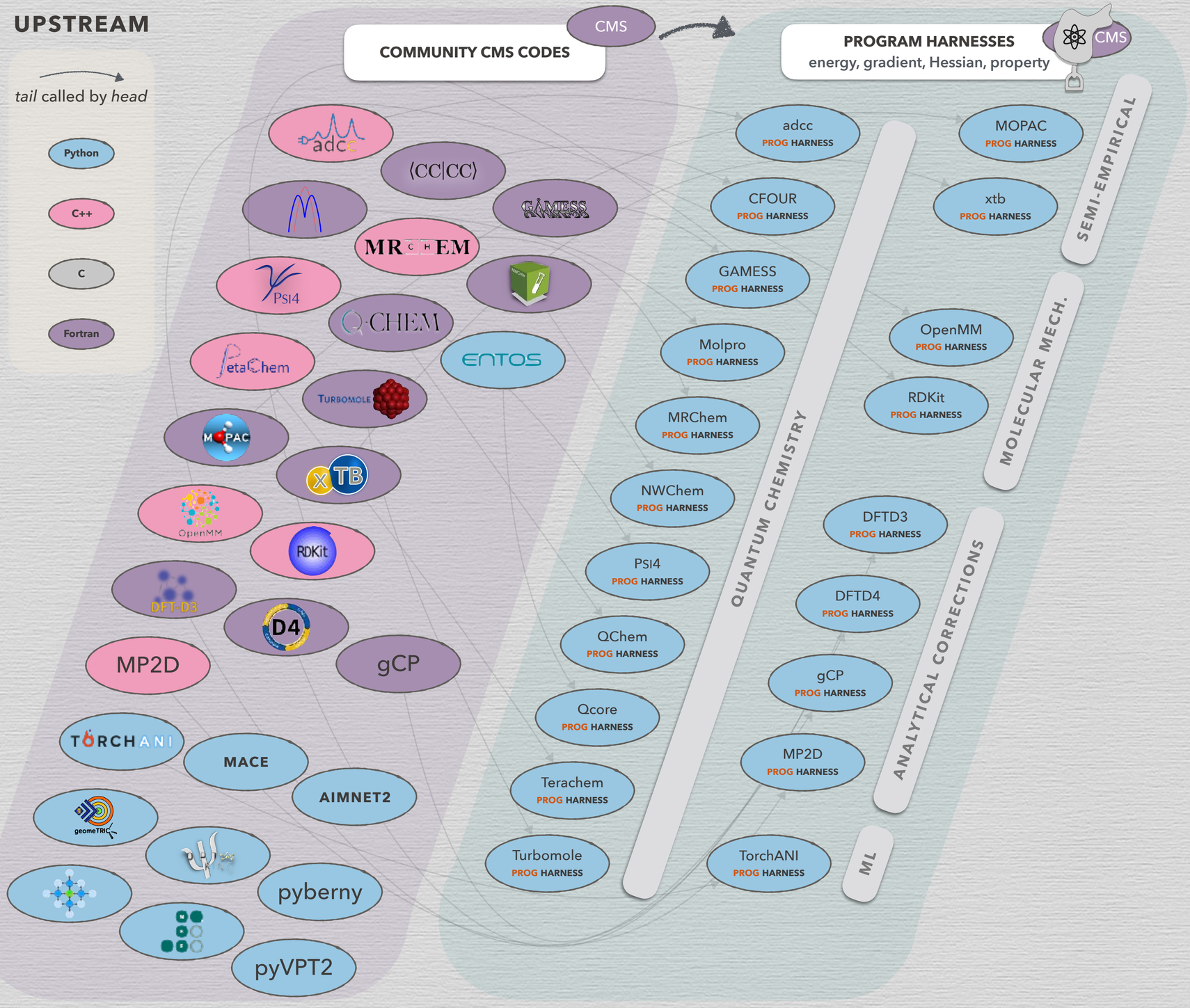
PYAPI

```
                                  cfour
qcengine.compute(atomicinput,  "molpro", task_config({"nnodes": 4, "memory": 10})
                                  psi4
                                  dftd4
```

```
sh> qcengine run cmscode atomicinput.json
```



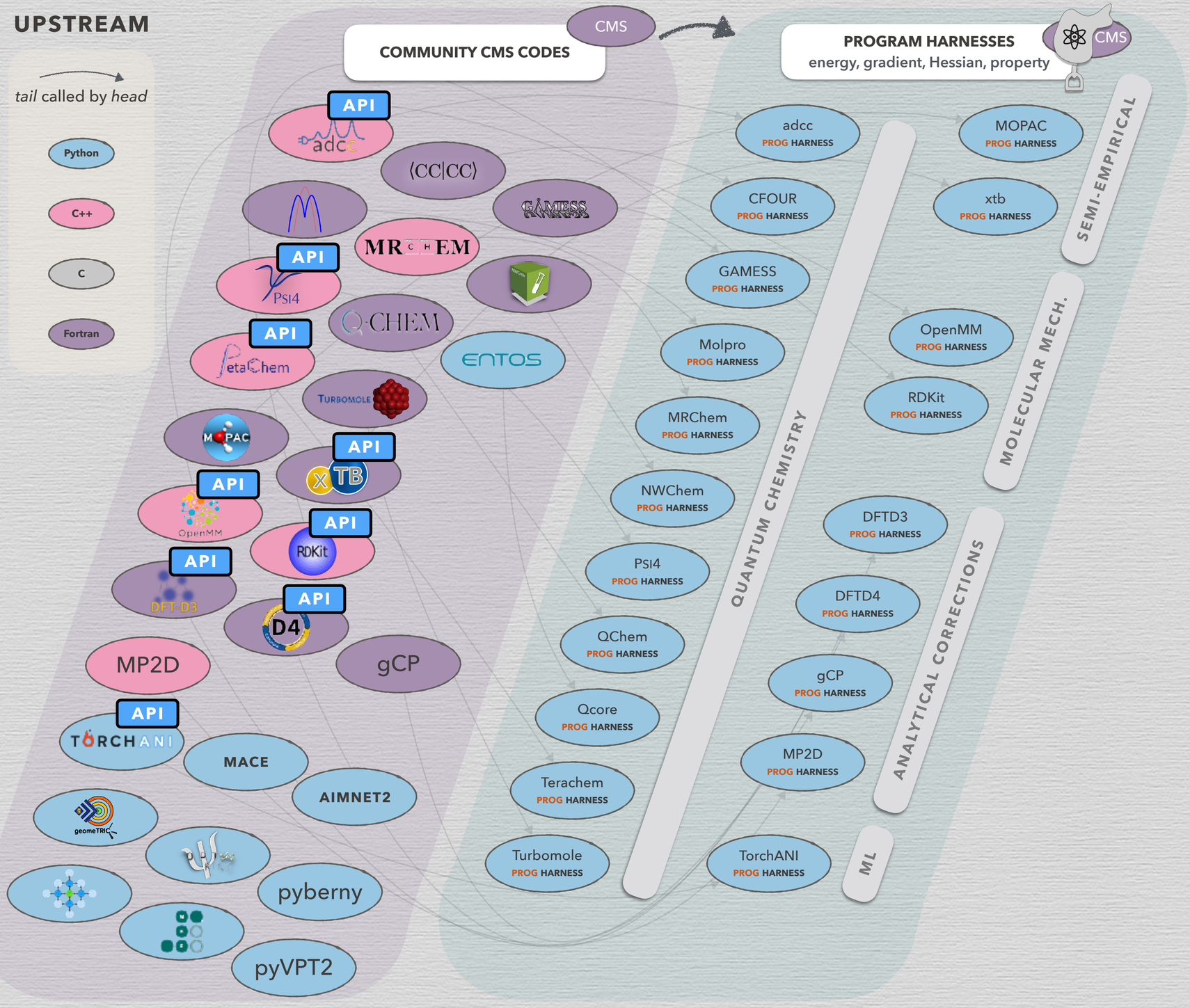

```
py> qcengine.compute("cmscode", atomicinput)
```



QCENGINE: INTERNAL MAP

github.com/MolSSI/QCEngine

- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.



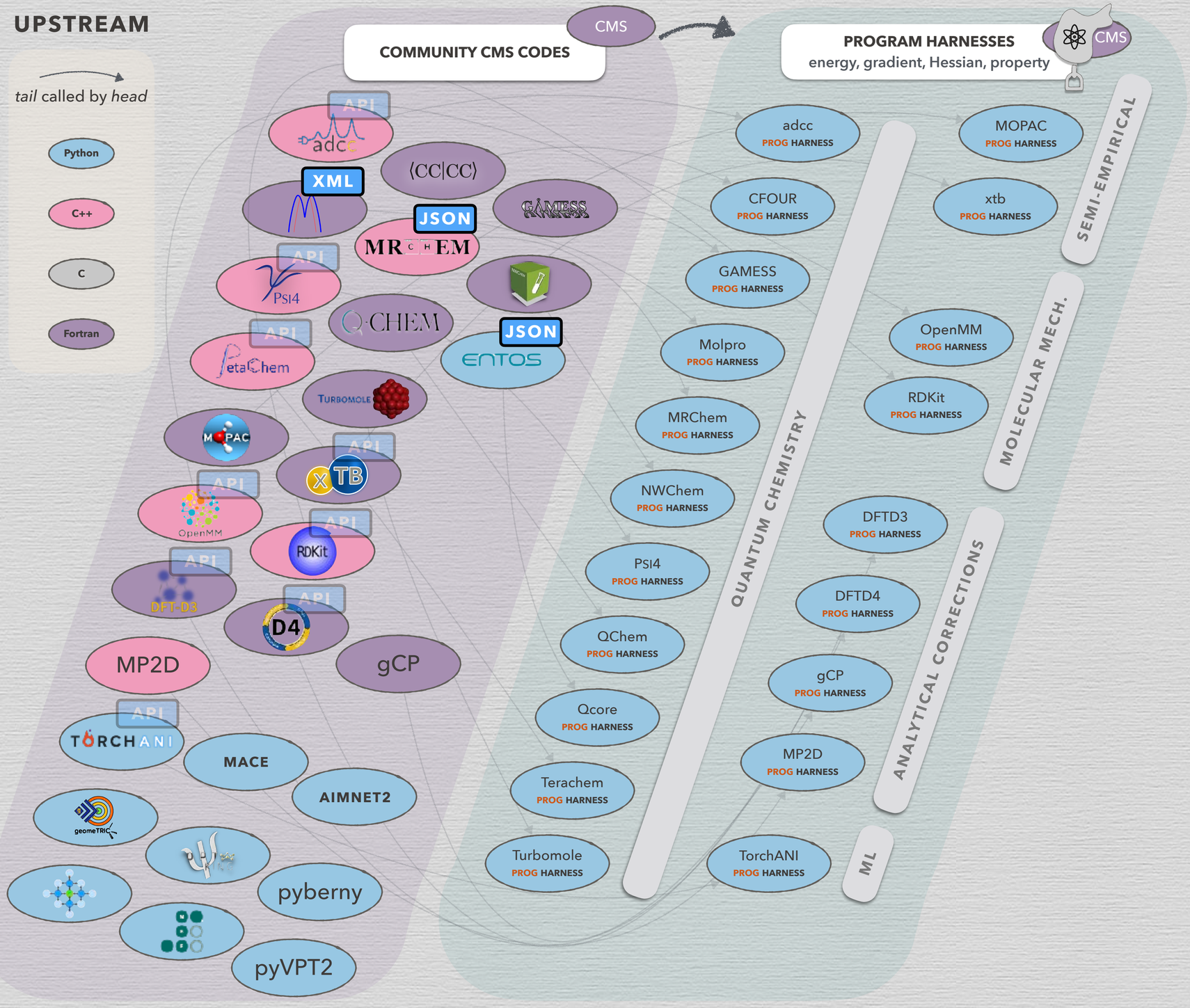
QCENGINE: INTERNAL MAP

github.com/MolSSI/QCEngine

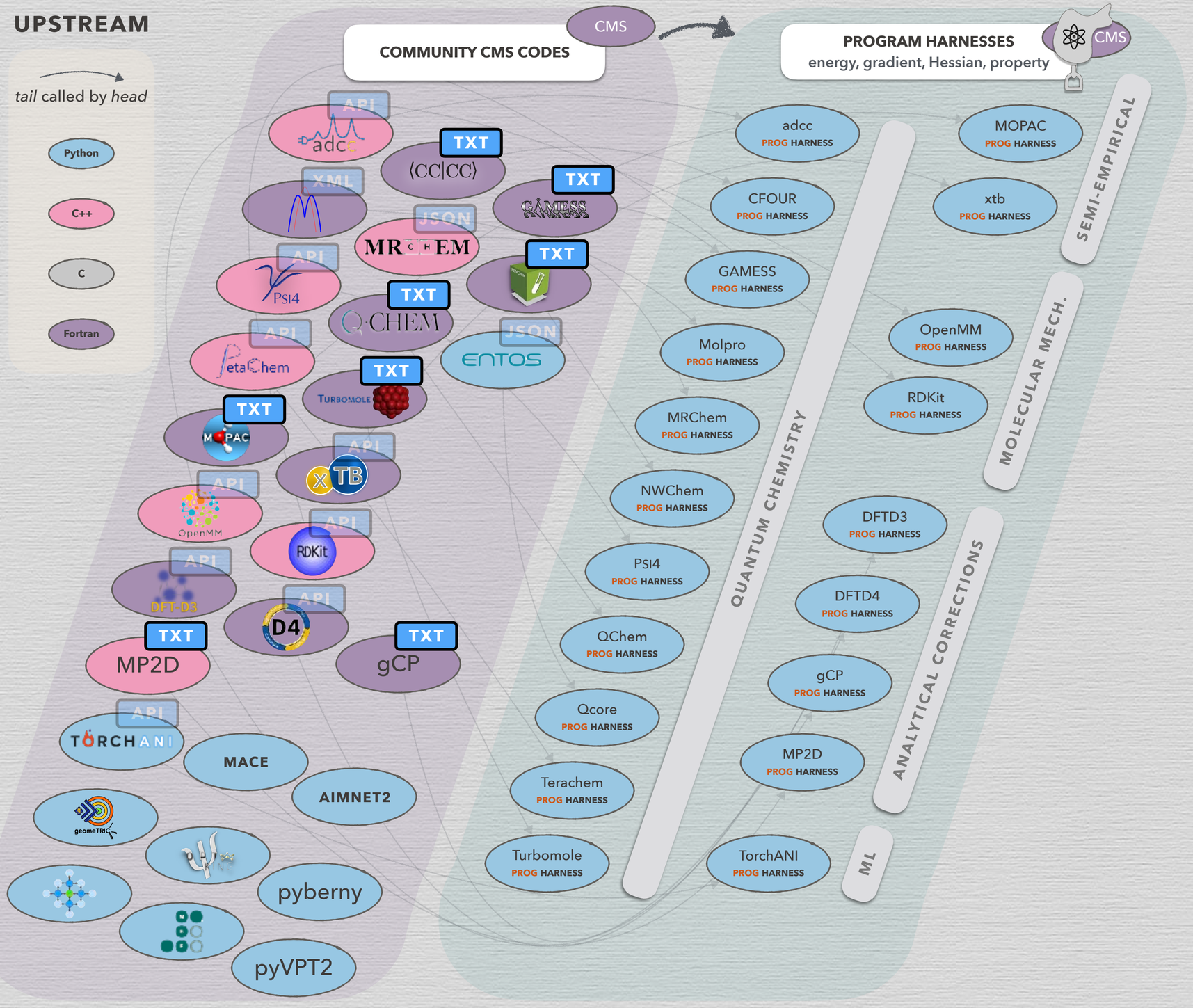
- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.

QCEngine: INTERNAL MAP

github.com/MolSSI/QCEngine



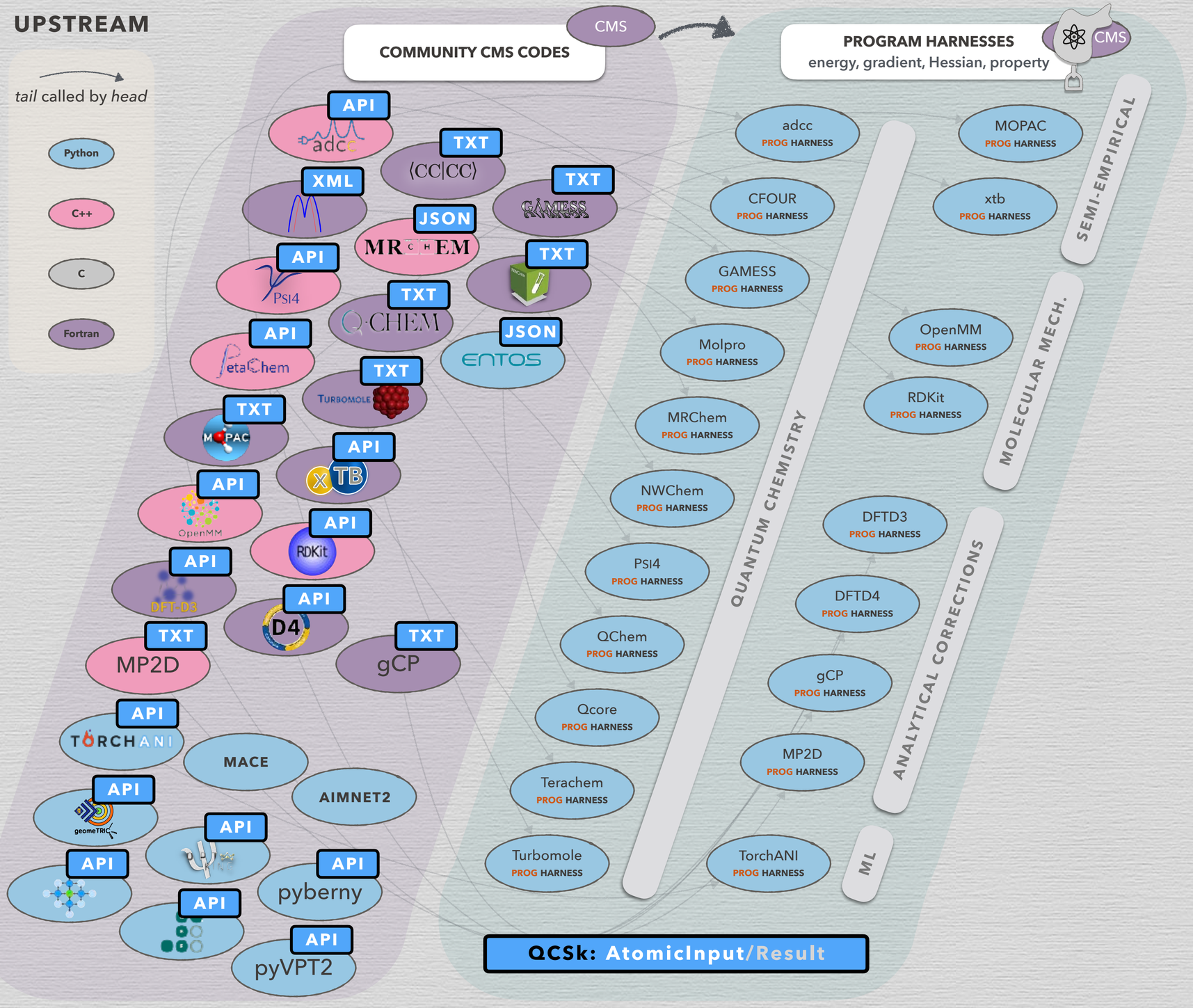
- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.



QCEngine: Internal Map

github.com/MolSSI/QCEngine

- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.

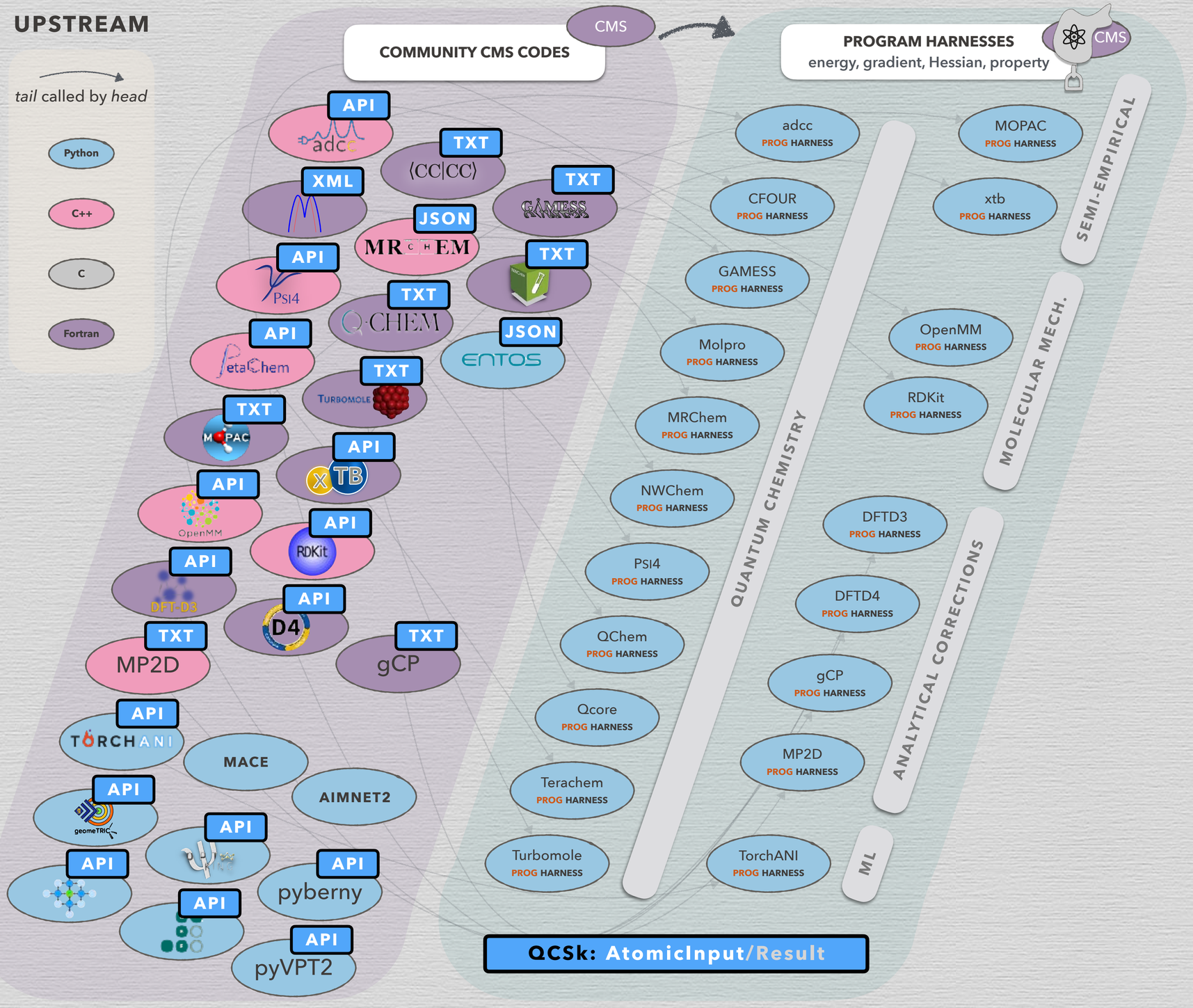


QCENGINE: INTERNAL MAP

github.com/MolSSI/QCEngine

- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.

DOWNSTREAM

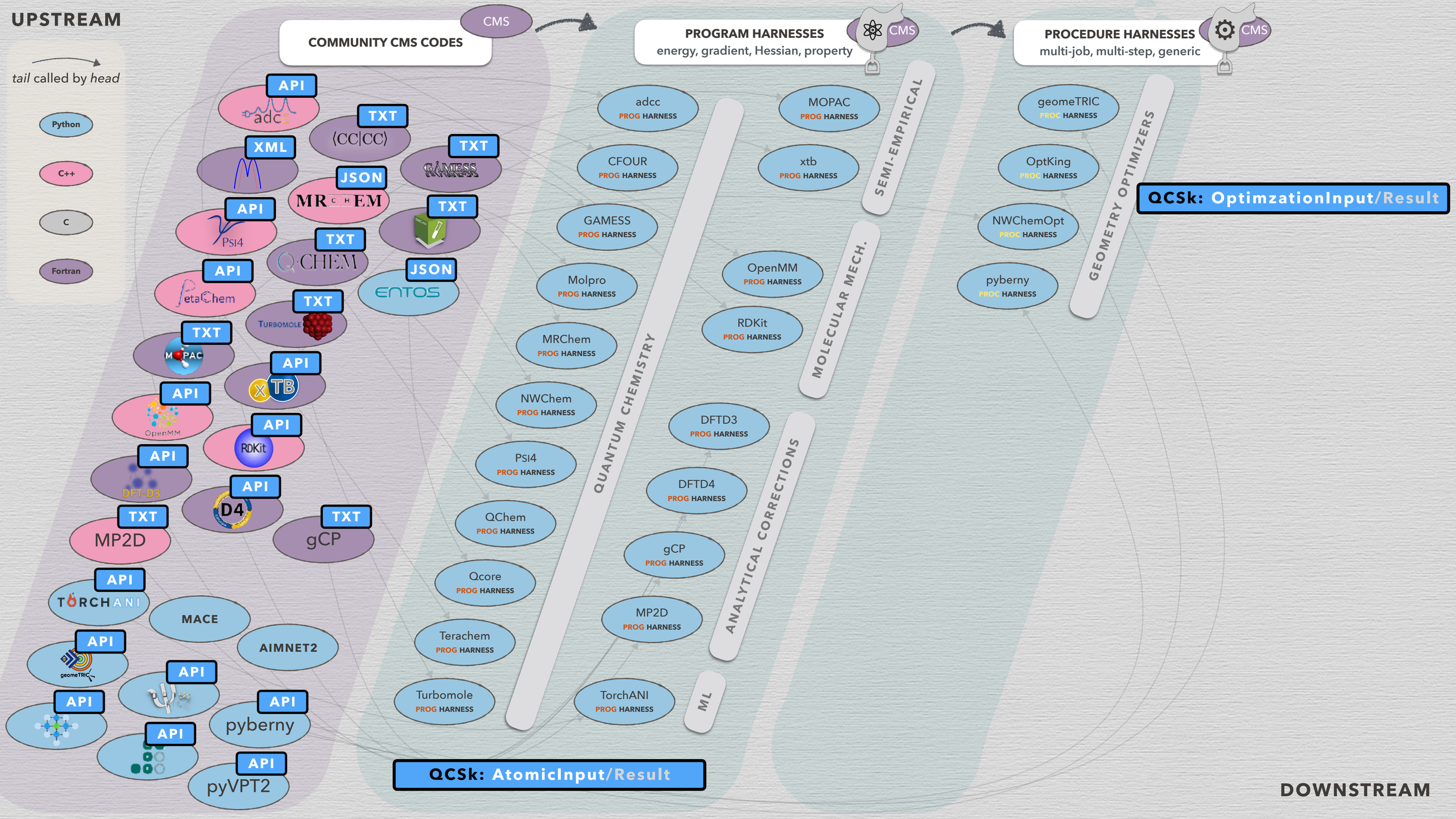


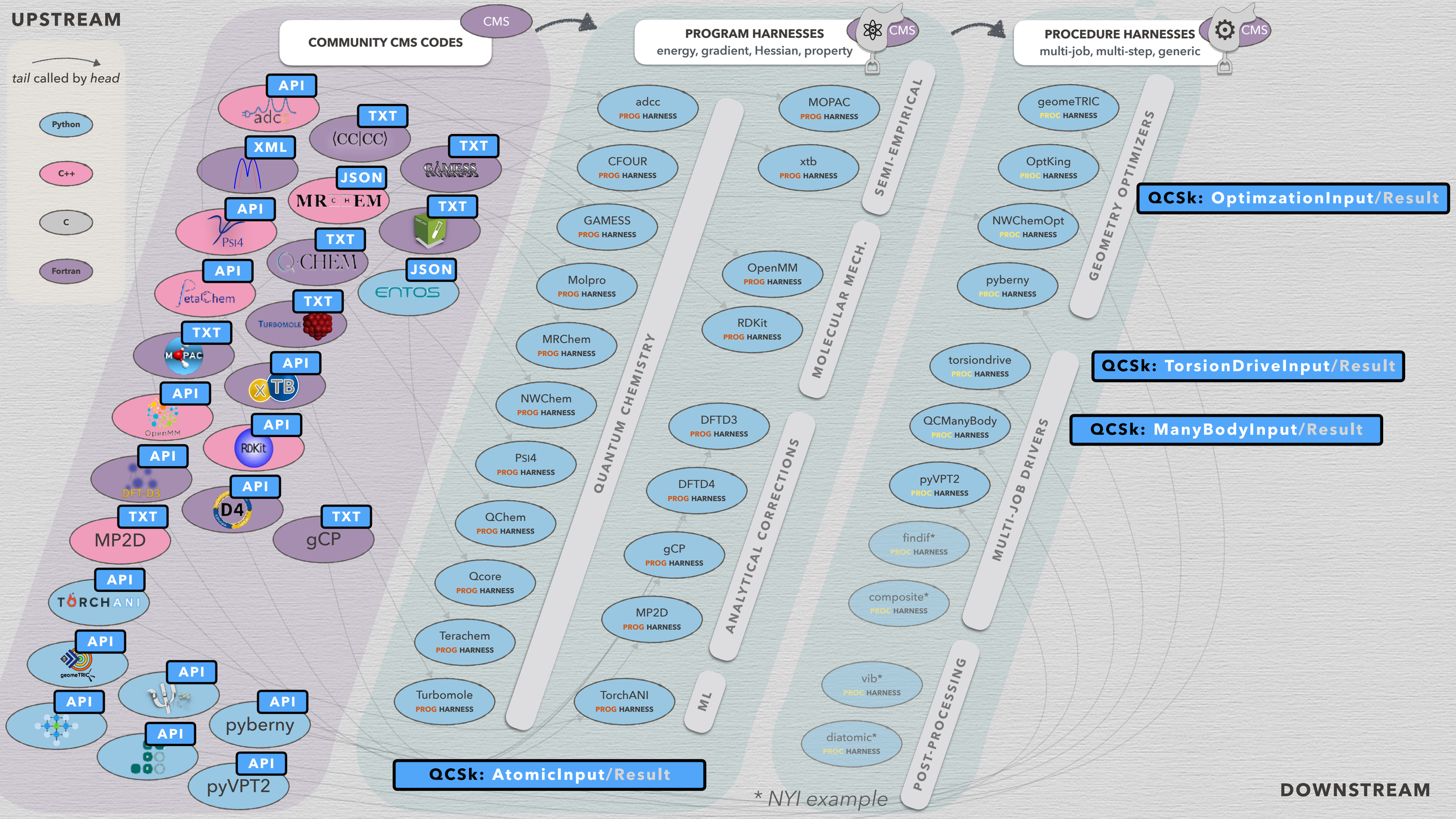
QCENGINE: INTERNAL MAP

github.com/MolSSI/QCEngine

- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also SE, MM, partial, or ML
- **INTERFACE** in variety of ways from API to structured data to regex. Former preferred for numerical precision.

DOWNSTREAM

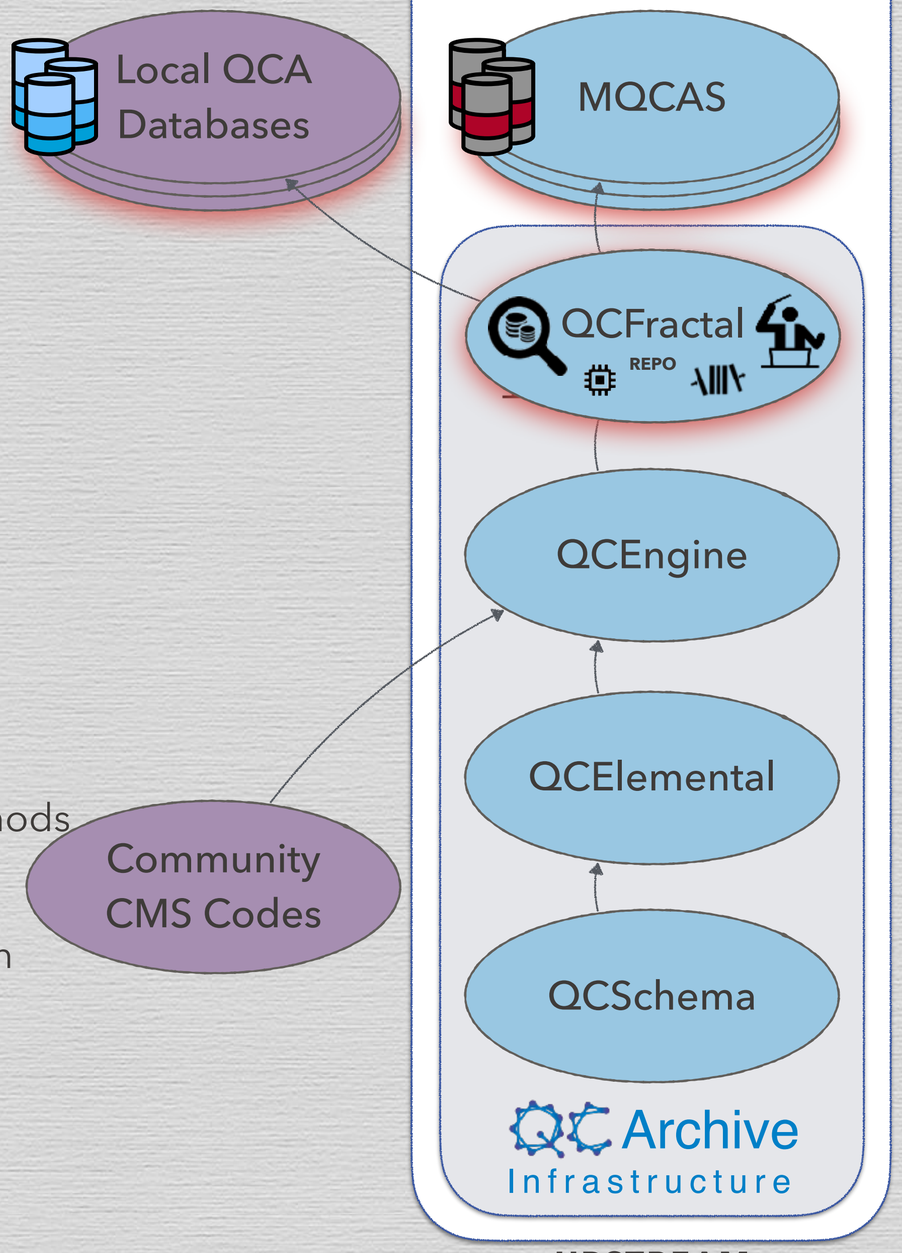

UPSTREAM
tail called by head
Python
C++
C
Fortran
COMMUNITY CMS CODES
CMS
PROGRAM HARNESSES
energy, gradient, Hessian, property
CMS
PROCEDURE HARNESSES
multi-job, multi-step, generic
CMS
API
adcc
TXT
CC
XML
JSON
MRCHEM
TXT
GAMESS
API
PSI4
TXT
Q-CHEM
TXT
JSON
ENTOS
API
PetaChem
TXT
TURBOMOLE
TXT
MOPAC
API
XTB
API
OpenMM
API
RDKit
API
DFT-D3
API
D4
TXT
MP2D
TXT
gCP
API
TORCHANI
MACE
AIMNET2
API
geomeTRIC
API
API
API
pyberny
API
API
pyVPT2
adcc
PROG HARNESS
CFOUR
PROG HARNESS
GAMESS
PROG HARNESS
Molpro
PROG HARNESS
MRChem
PROG HARNESS
NWChem
PROG HARNESS
PSI4
PROG HARNESS
QChem
PROG HARNESS
Qcore
PROG HARNESS
Terachem
PROG HARNESS
Turbomole
PROG HARNESS
QUANTUM CHEMISTRY
MOPAC
PROG HARNESS
xtb
PROG HARNESS
SEMI-EMPIRICAL
OpenMM
PROG HARNESS
RDKit
PROG HARNESS
MOLECULAR MECH.
DFTD3
PROG HARNESS
DFTD4
PROG HARNESS
gCP
PROG HARNESS
MP2D
PROG HARNESS
ANALYTICAL CORRECTIONS
TorchANI
PROG HARNESS
ML
geomeTRIC
PROC HARNESS
OptKing
PROC HARNESS
NWChemOpt
PROC HARNESS
pyberny
PROC HARNESS
GEOMETRY OPTIMIZERS
QCSk: OptimzationInput/Result
QCSk: AtomicInput/Result
DOWNSTREAM


UPSTREAM
tail called by head
Python
C++
C
Fortran
COMMUNITY CMS CODES
CMS
PROGRAM HARNESSES
energy, gradient, Hessian, property
CMS
PROCEDURE HARNESSES
multi-job, multi-step, generic
CMS
API
TXT
XML
JSON
adcc
(CC|CC)
GAMESS
MRCHEM
PSI4
Q-CHEM
ENTOS
PetaChem
TURBOMOLE
MOPAC
xTB
OpenMM
RDKit
DFT-D3
D4
MP2D
gCP
TORCHANI
geomeTRIC
pyberny
pyVPT2
adcc PROG HARNESS
CFOUR PROG HARNESS
GAMESS PROG HARNESS
Molpro PROG HARNESS
MRChem PROG HARNESS
NWChem PROG HARNESS
PSI4 PROG HARNESS
QChem PROG HARNESS
Qcore PROG HARNESS
Terachem PROG HARNESS
Turbomole PROG HARNESS
MOPAC PROG HARNESS
xtb PROG HARNESS
OpenMM PROG HARNESS
RDKit PROG HARNESS
DFTD3 PROG HARNESS
DFTD4 PROG HARNESS
gCP PROG HARNESS
MP2D PROG HARNESS
TorchANI PROG HARNESS
QUANTUM CHEMISTRY
SEMI-EMPIRICAL
MOLECULAR MECH.
ANALYTICAL CORRECTIONS
ML
QCSk: AtomicInput/Result
geomeTRIC PROC HARNESS
OptKing PROC HARNESS
NWChemOpt PROC HARNESS
pyberny PROC HARNESS
torsiondrive PROC HARNESS
QCManyBody PROC HARNESS
pyVPT2 PROC HARNESS
findif* PROC HARNESS
composite* PROC HARNESS
vib* PROC HARNESS
diatomic* PROC HARNESS
GEOMETRY OPTIMIZERS
MULTI-JOB DRIVERS
POST-PROCESSING
QCSk: OptimzationInput/Result
QCSk: TorsionDriveInput/Result
QCSk: ManyBodyInput/Result
* NYI example
DOWNSTREAM

QCArchive STACK

DOWNSTREAM

- database, permissioned query
- fully powerful as MQCAS but locally controlled



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- database storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

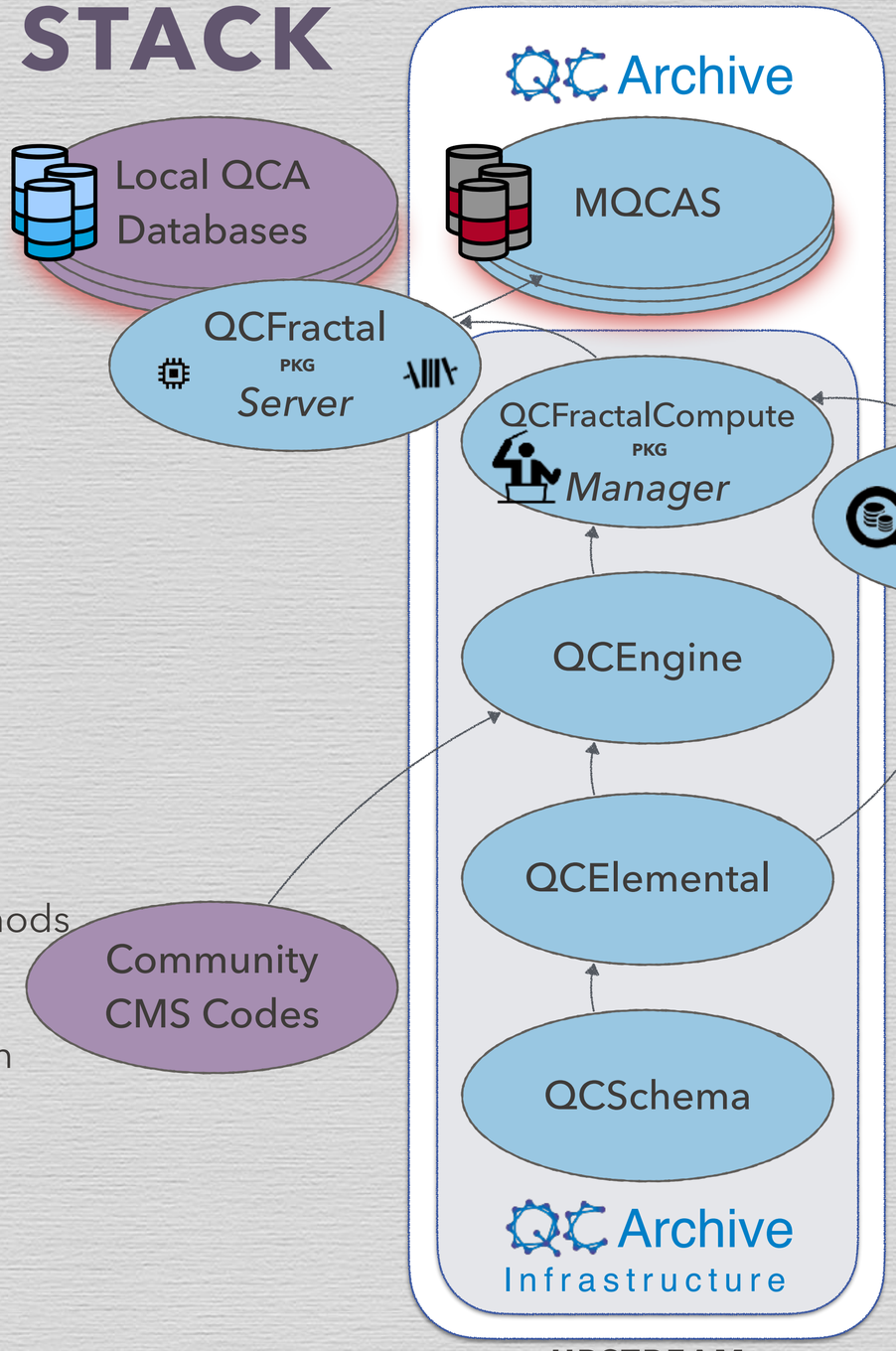
- the difficult part – coded QC methods
- structured output uncommon
- DSL input; API/schema uncommon

UPSTREAM

QCArchive STACK

DOWNSTREAM

- database, permissioned query
- fully powerful as MQCAS but locally controlled



- database, universally queryable for CMS results
- MolSSI QCArchive Server ("em-quacks")

- batch (parallel) compute setup & management
- storage & query of QC results

- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- QC data layout & descriptions
- e.g., molecule, DFT properties, grad. input, opt. output
- language agnostic

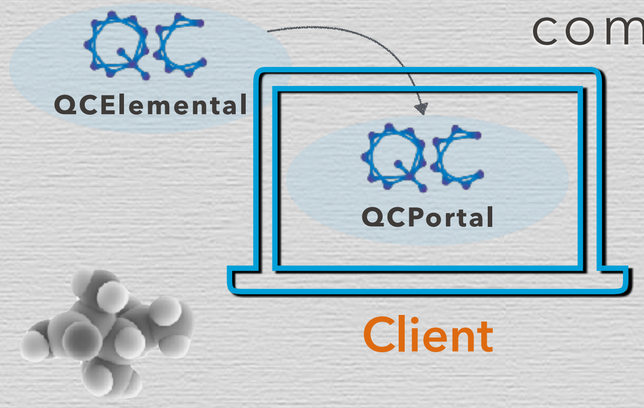
- the difficult part – coded QC methods
- structured output uncommon
- DSL input; API/schema uncommon

UPSTREAM

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org

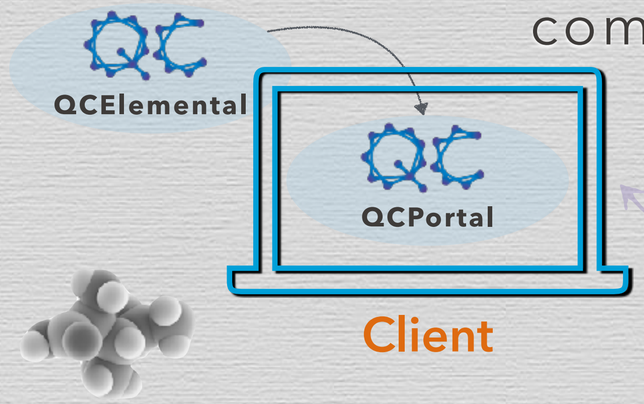


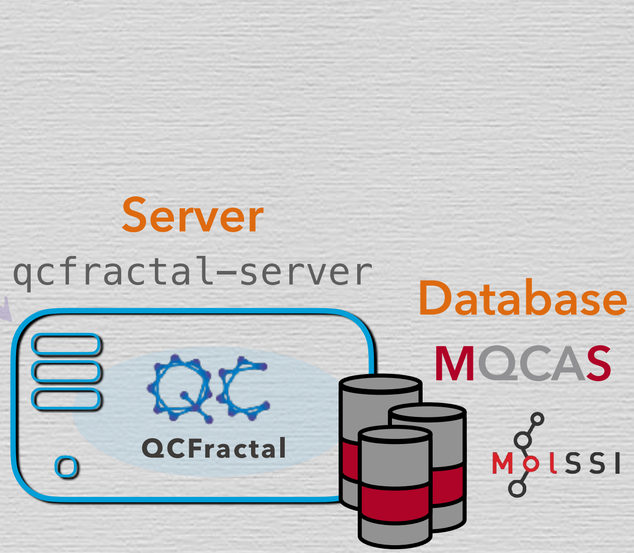
- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** — up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** — runs on laptops, campus clusters, and leadership-class supercomputers

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



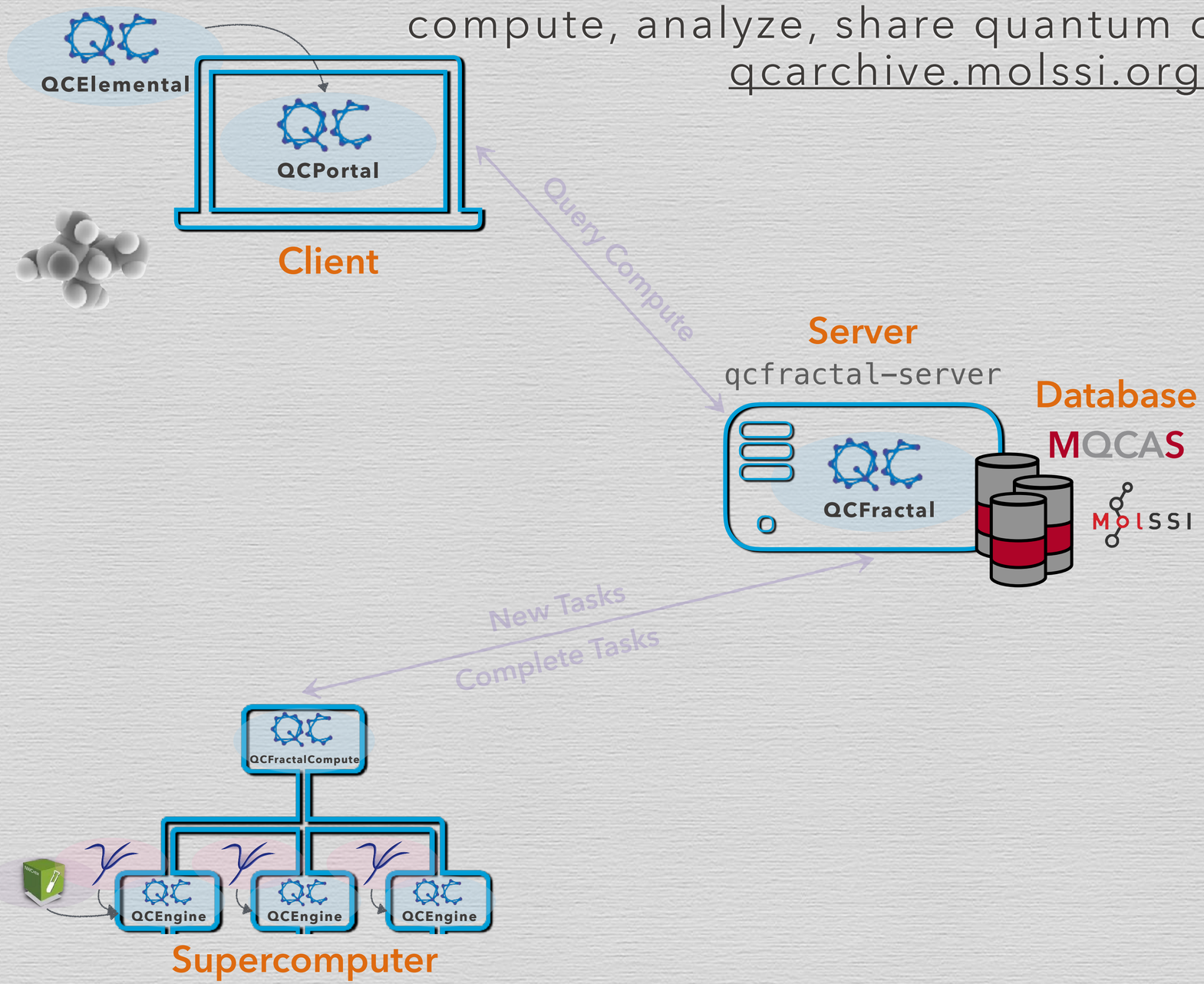


- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** — up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** — runs on laptops, campus clusters, and leadership-class supercomputers

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

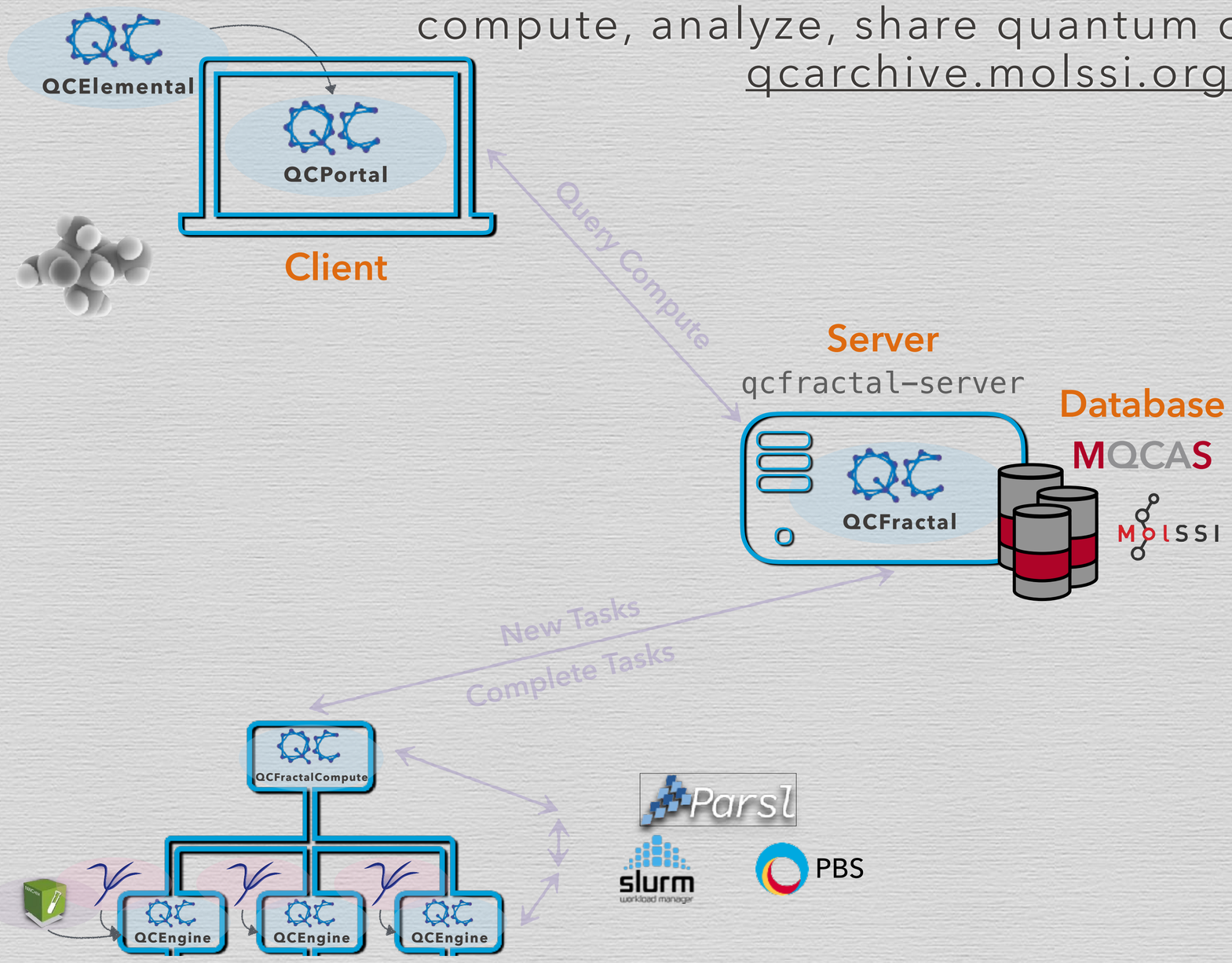
`qcfractal-compute-manager` **Manager**

worker **Compute Nodes**

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter Task Management

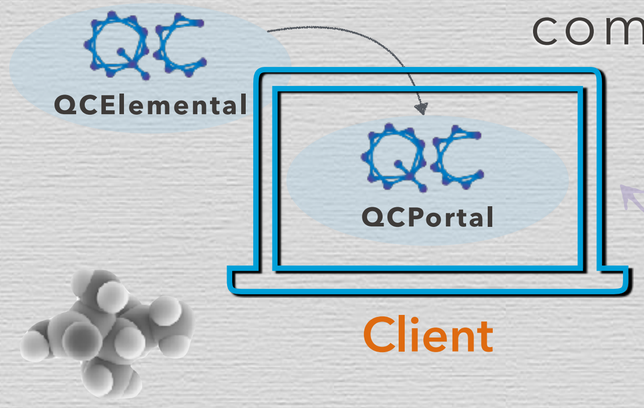
scheduler Task Scheduling

worker Compute Nodes

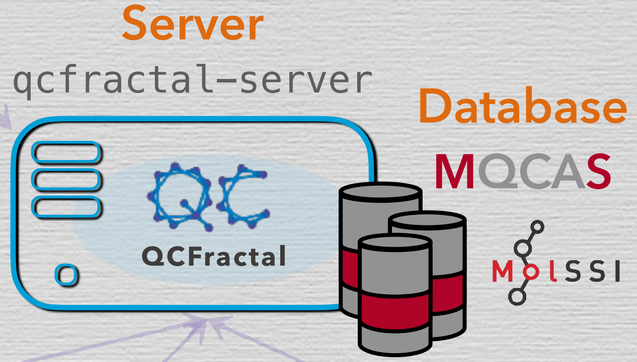
QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org

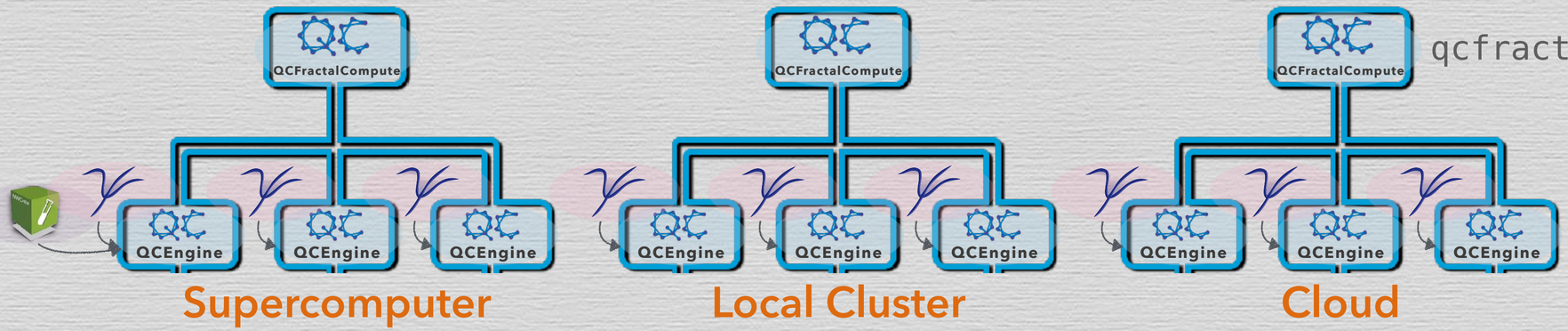


Query Compute



New Tasks

Complete Tasks



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter **Task Management**

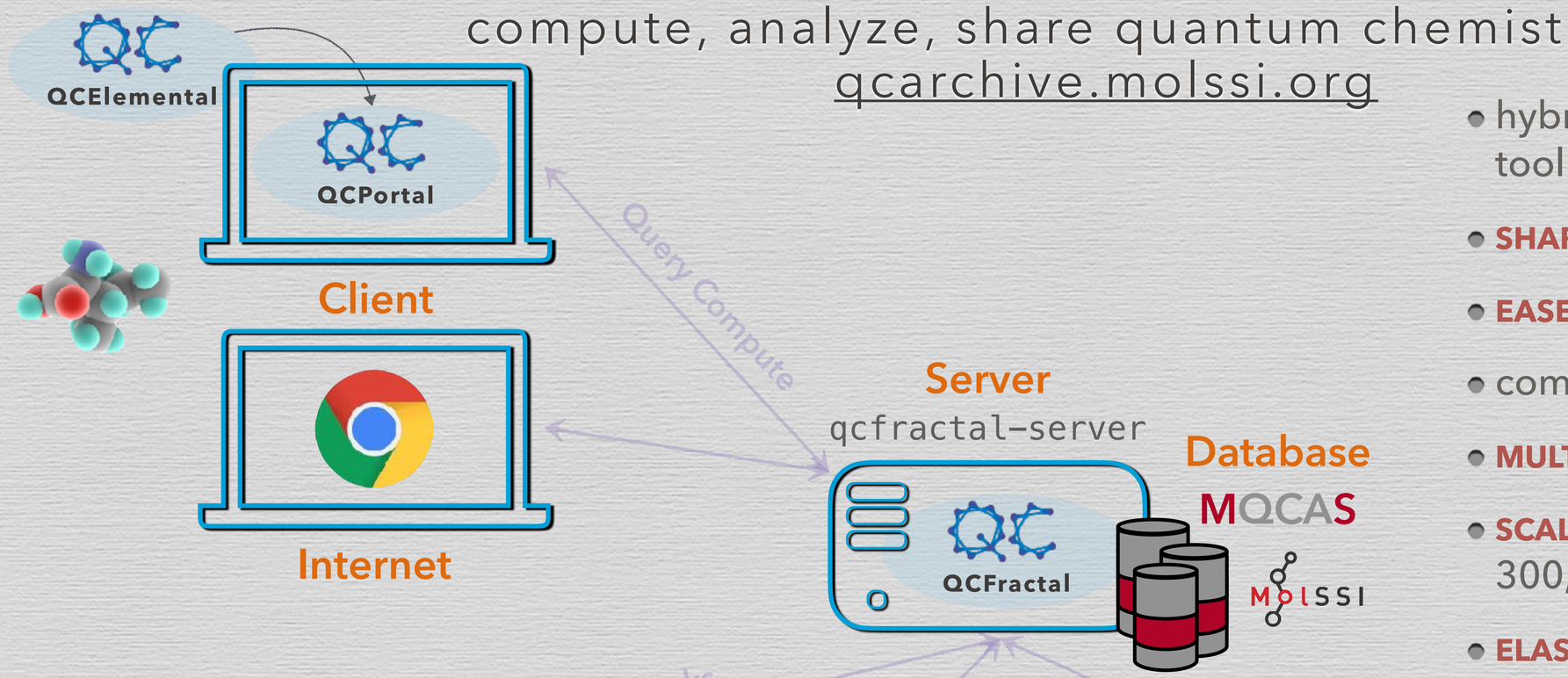
scheduler **Task Scheduling**

worker **Compute Nodes**

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter Task Management

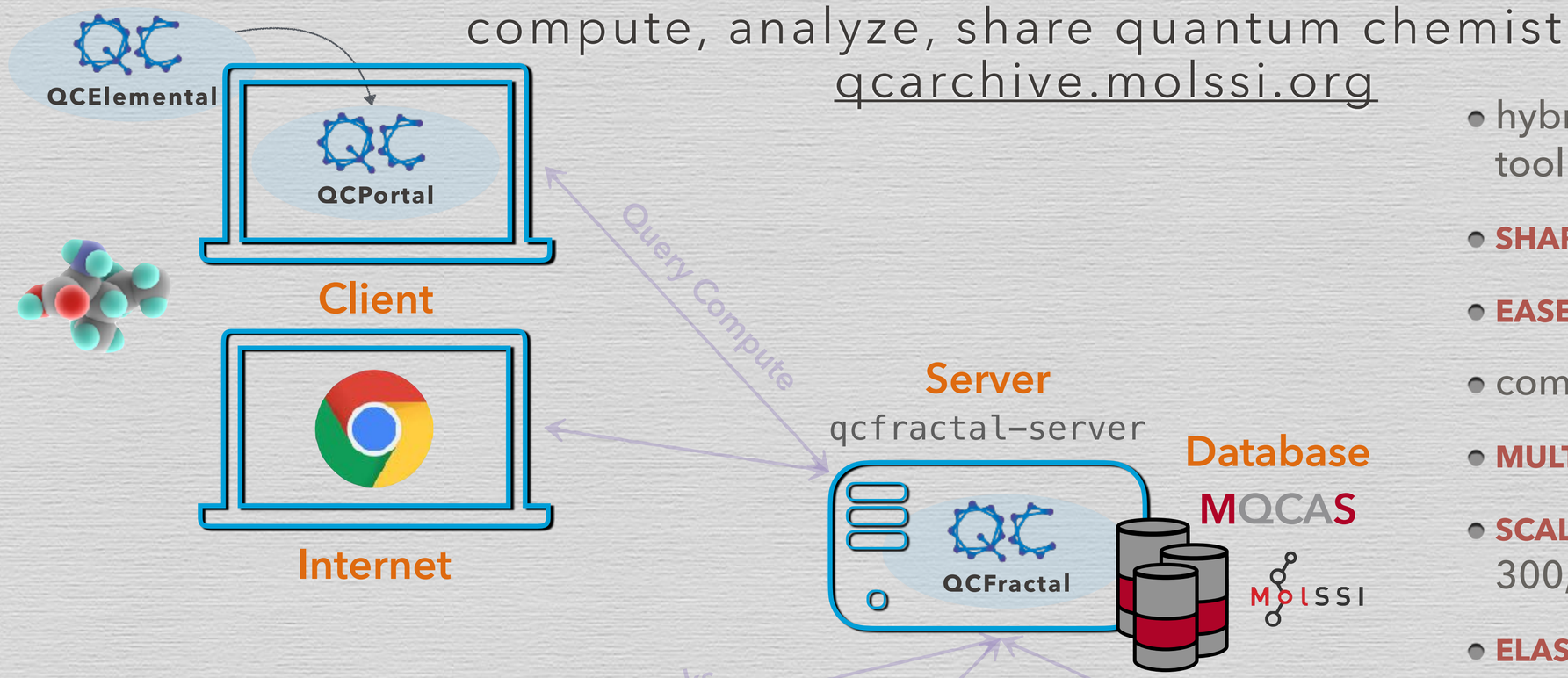
scheduler Task Scheduling

worker Compute Nodes

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

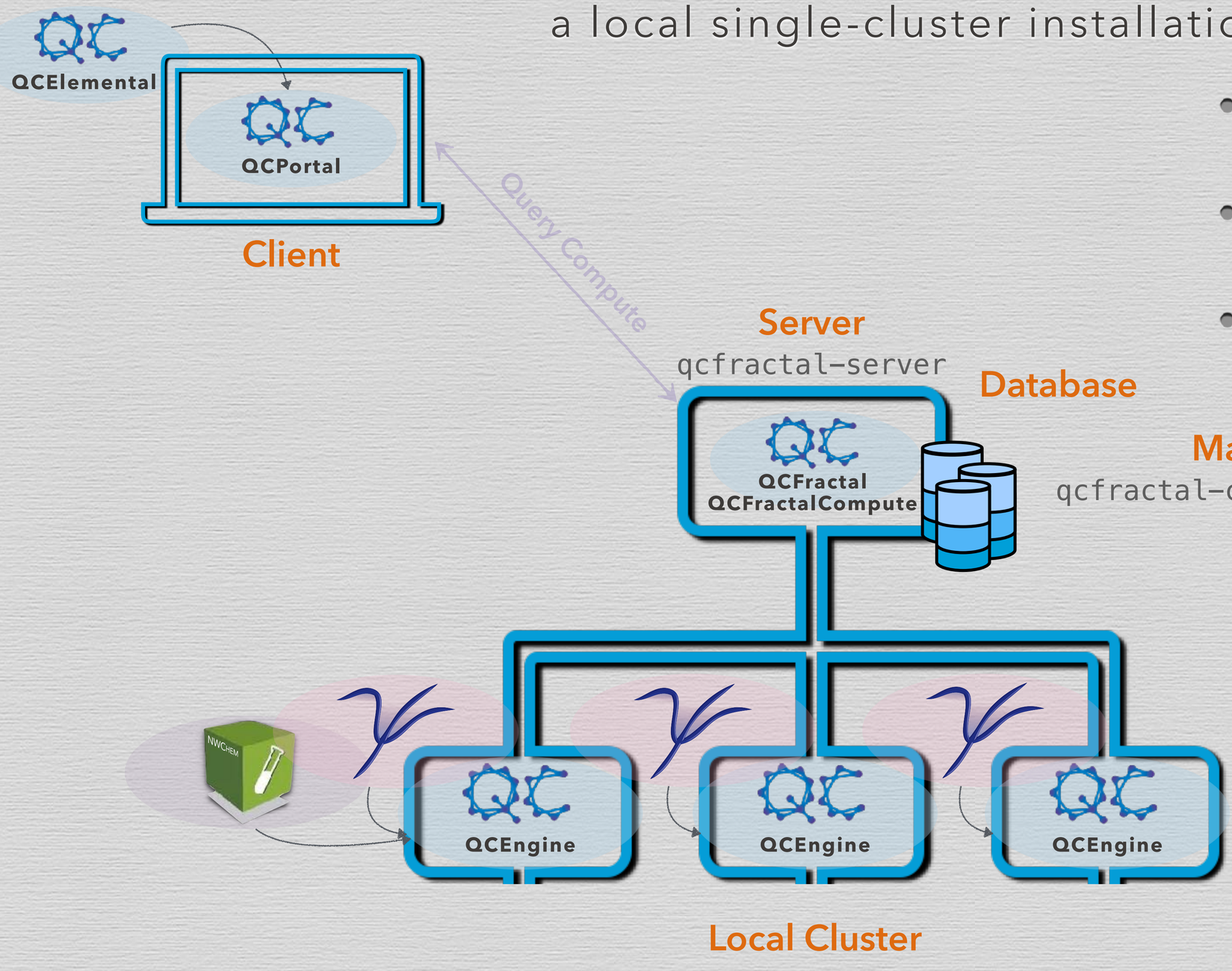
adapter Task Management

scheduler Task Scheduling

worker Compute Nodes

QCARCHIVE, SMALLER

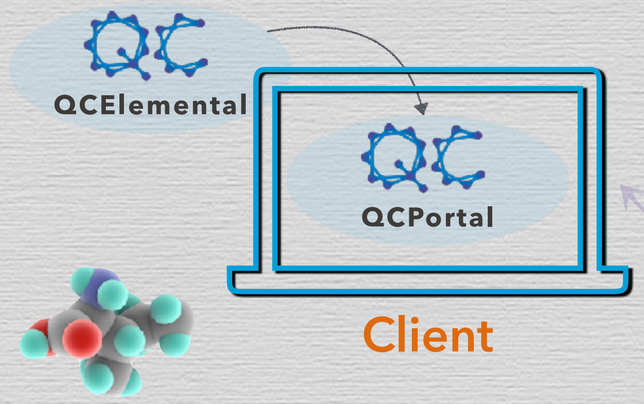
a local single-cluster installation

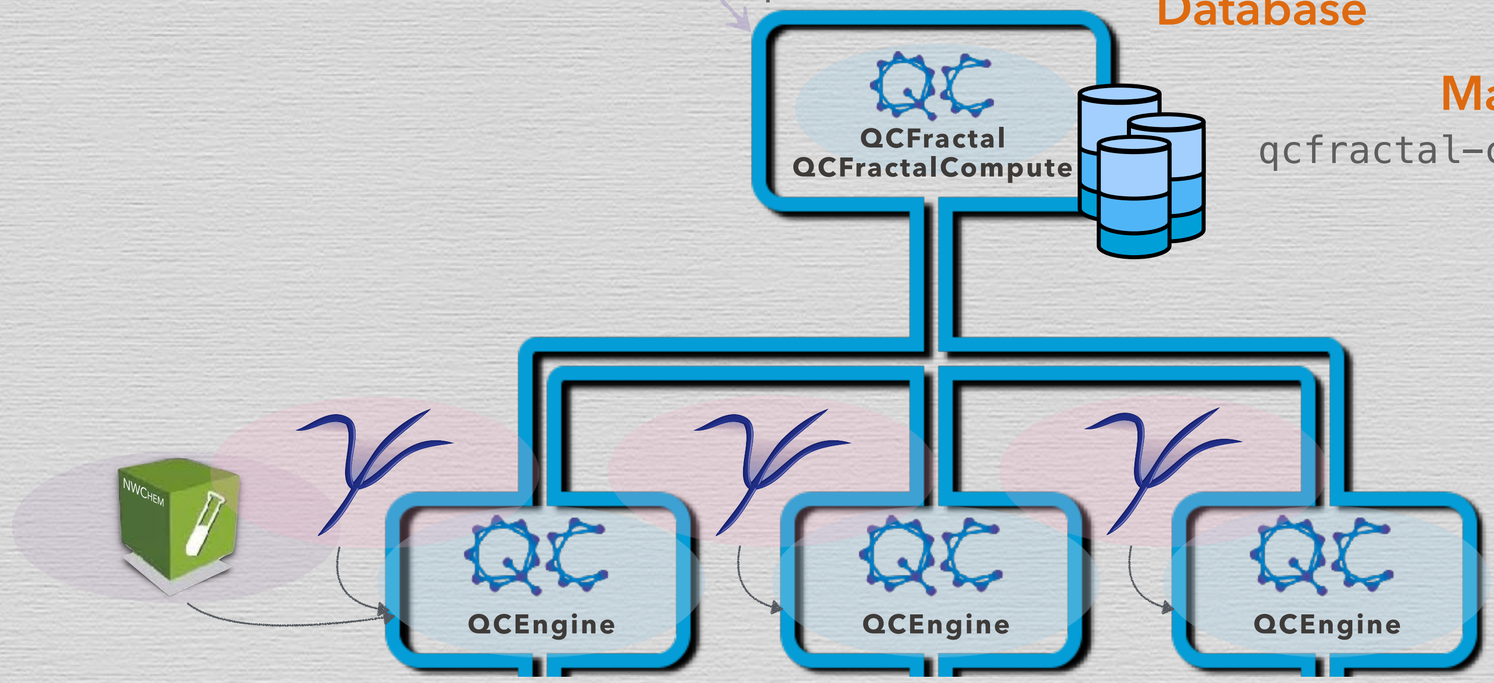


- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal and database for task submission and orchestration
- **LOCAL** database fully powerful as MQCAS but locally controlled

QCARCHIVE, SMALLER

a local single-cluster installation

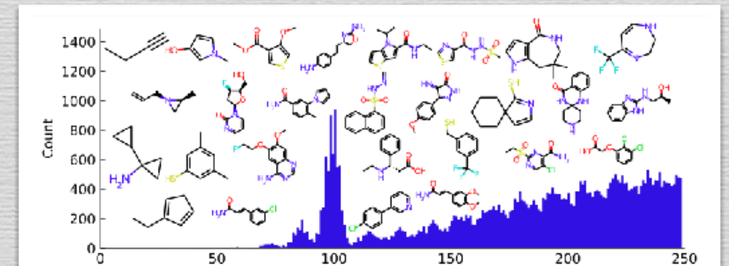


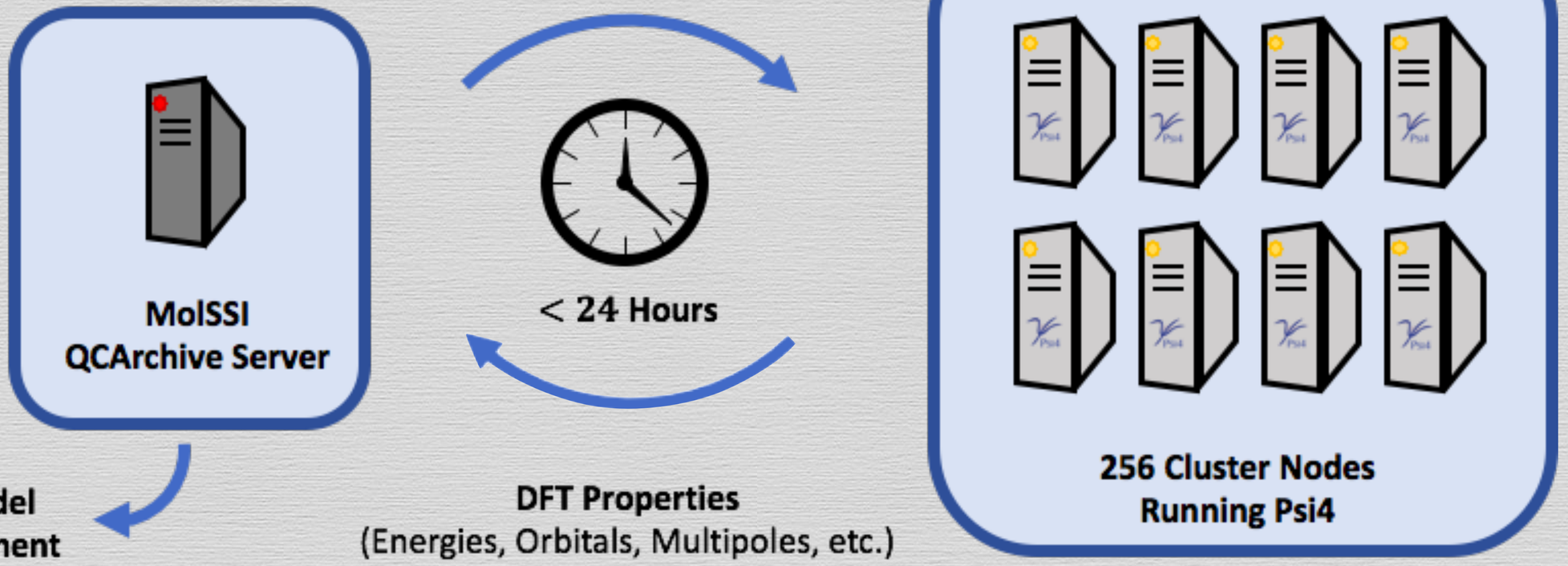


- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal and database for task submission and orchestration
- **LOCAL** database fully powerful as MQCAS but locally controlled

APP: QCARCHIVE FOR THROUGHPUT & REPRODUCIBILITY

HPC message-passing neural network computations preserved on public server





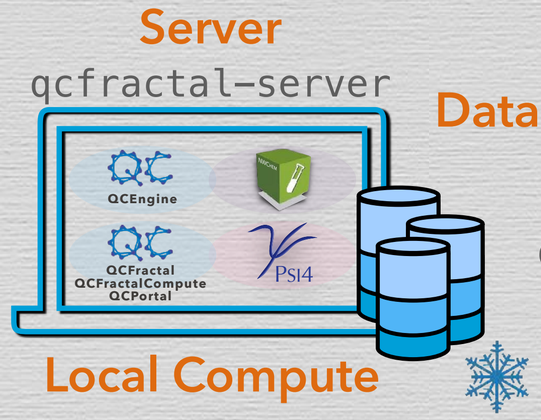
Zach Glick
GaTech → Lavo

Z. L. Glick, A. Koutsoukas, D. L. Cheney, and C. D. Sherrill, J. Chem. Phys 143 224013 (2021).

QCARCHIVE, SMALLEST

a local single-box installation

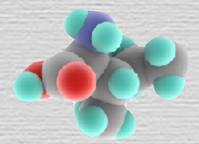
- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal, database and QC codes for self-contained setup
- **SNOWFLAKE** a still-simpler mode where the database does not persist

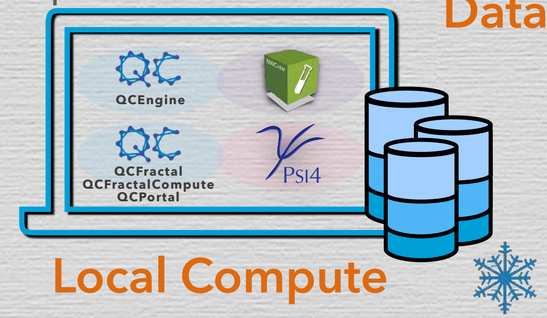


QCARCHIVE, SMALLEST

a local single-box installation

- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal, database and QC codes for self-contained setup
- **SNOWFLAKE** a still-simpler mode where the database does not persist





Server

`qcfractal-server`

Database

Manager

`qcfractal-compute-manager`

Local Compute

worker

APP: PSI4 DISTRIBUTED DRIVER

post-processing flexibility vs. automation: an unwelcome choice

```
psi4.energy("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule=  )
```

read: *water dimer at double-triple-zeta extrapolated MP2 with a coupled-cluter correction, counterpoise corrected*

Local Compute

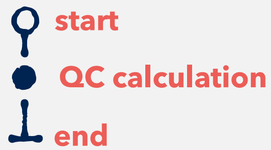
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by sum of all calcs since run sequentially.
- **RETRIEVAL** from filesystem difficult since many calcs in aggregated outfile.
- **FLEXIBILITY** limited to PSI4 only.

PSI4 DISTRIBUTED DRIVER

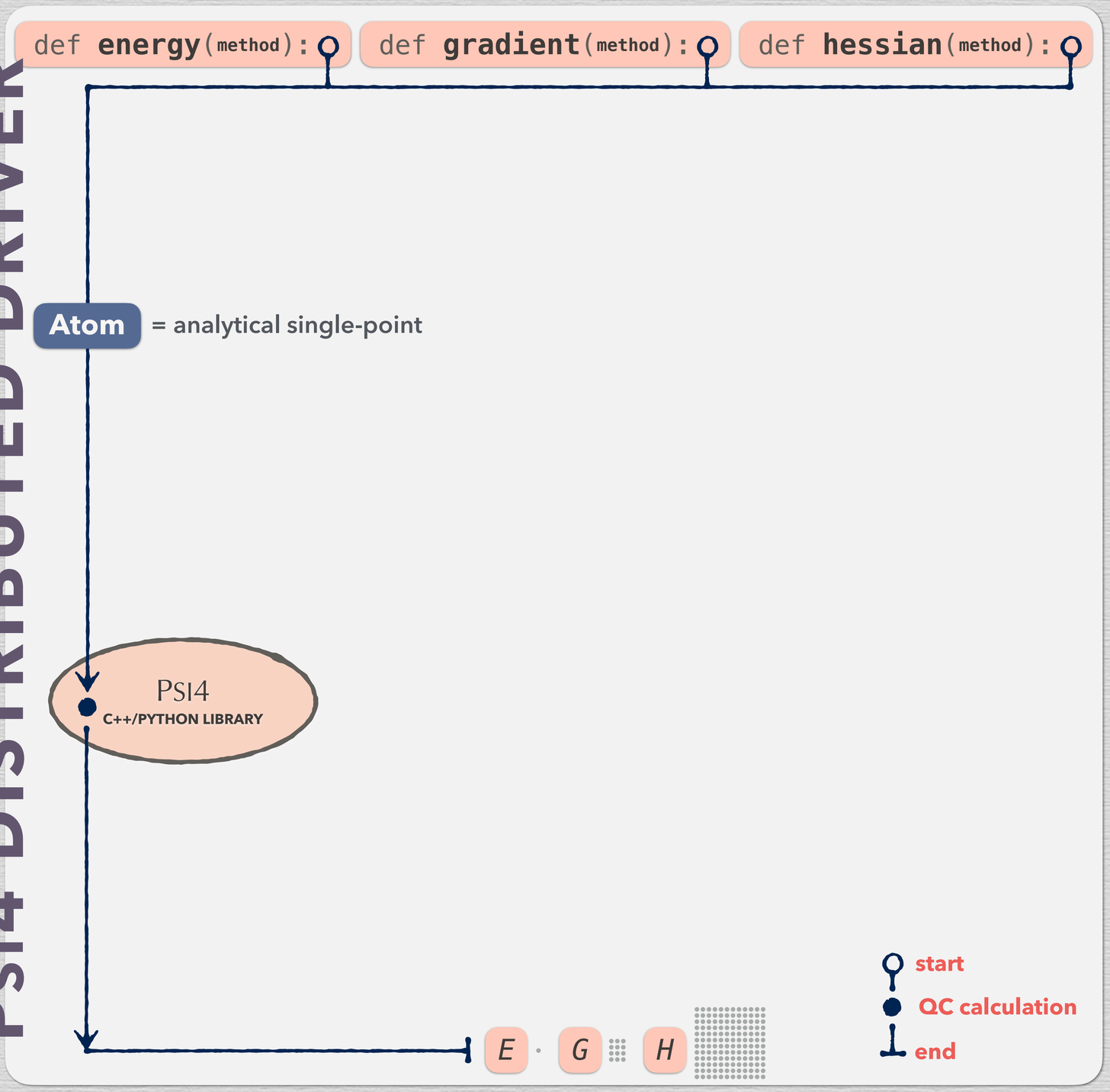
```
def energy(method):
```

```
def gradient(method):
```

```
def hessian(method):
```



PSI4 DISTRIBUTED DRIVER



PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

Atom = analytical single-point

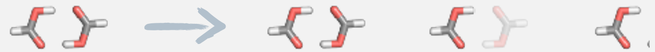
PSI4
C++/PYTHON LIBRARY

E · *G* *H*

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

Atom = analytical single-point

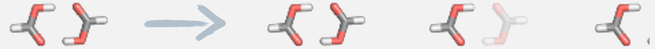
PSI4
C++/PYTHON LIBRARY

E · *G* ⋮ *H*

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil.
Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

Atom = analytical single-point

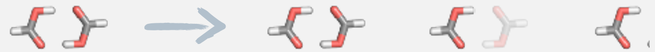
PSI4
C++/PYTHON LIBRARY

E · *G* ⋮ *H*

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil.
Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

`class CompositeComputer ():`

PLAN Separate **`method`** into method, basis, & extrapolations. **`molecule`** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

ASM Assemble extrapolations & total results from modelchems.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

Atom = analytical single-point

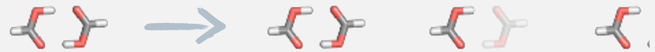
PSI4
C++/PYTHON LIBRARY

E · *G* ⋮ *H*

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil. Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

`class CompositeComputer ():`

PLAN Separate **`method`** into method, basis, & extrapolations. **`molecule`** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

ASM Assemble extrapolations & total results from modelchems.

`class AtomicComputer ():`

PLAN **`molecule`** & **`method`** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

MBE FD CBS Atom

POOL

I

QCSchema: AtomicInput

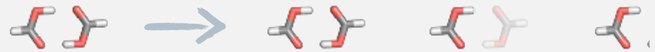
PSI4
C++/PYTHON LIBRARY

E · G ⋮ H

start
QC calculation
end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil. Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

`class CompositeComputer ():`

PLAN Separate **`method`** into method, basis, & extrapolations. **`molecule`** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ` `MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

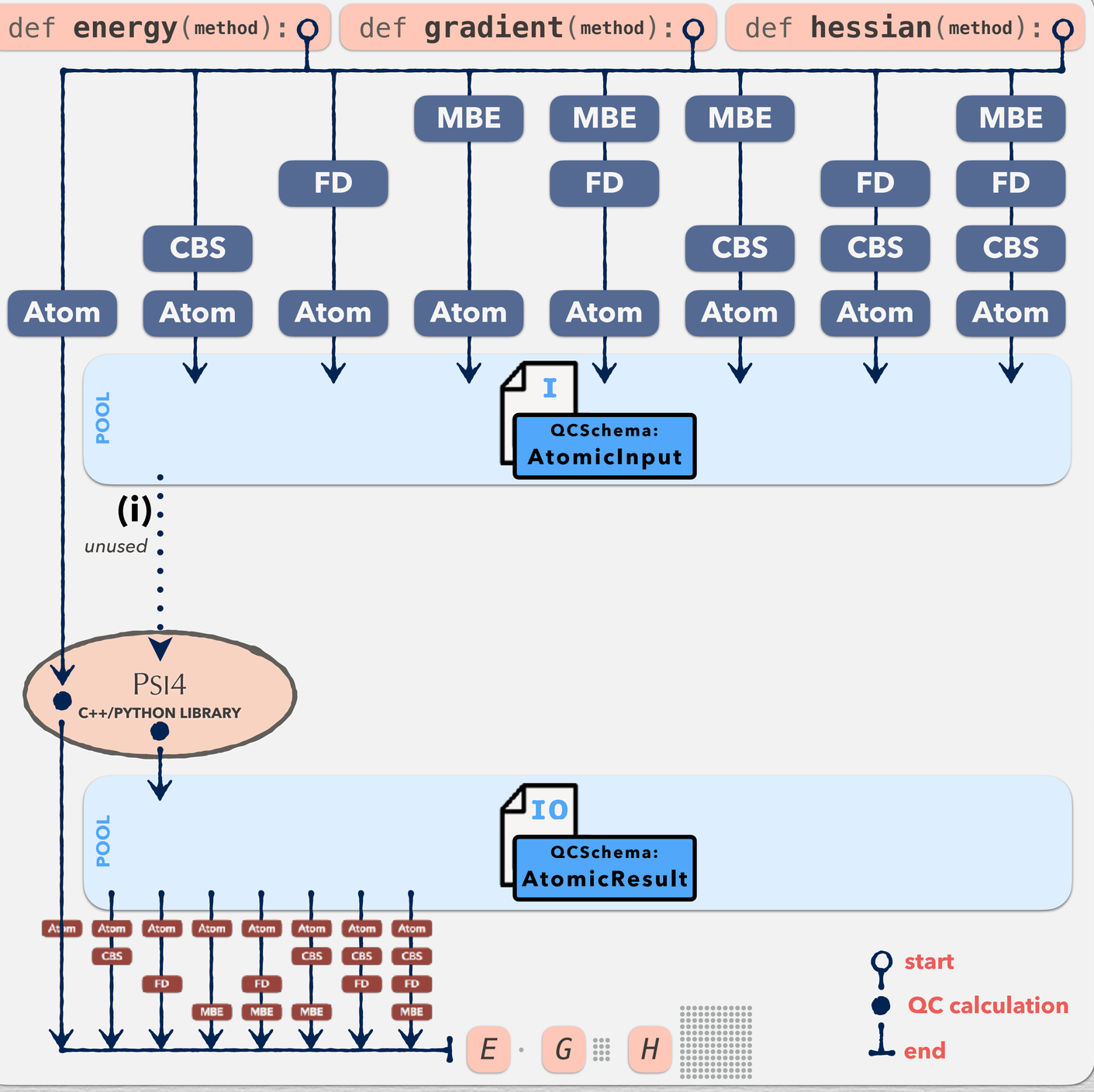
ASM Assemble extrapolations & total results from modelchems.

`class AtomicComputer ():`

PLAN **`molecule`** & **`method`** unchanged. `return qcschema` QCSchema

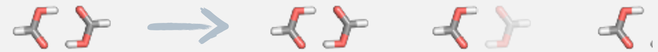
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

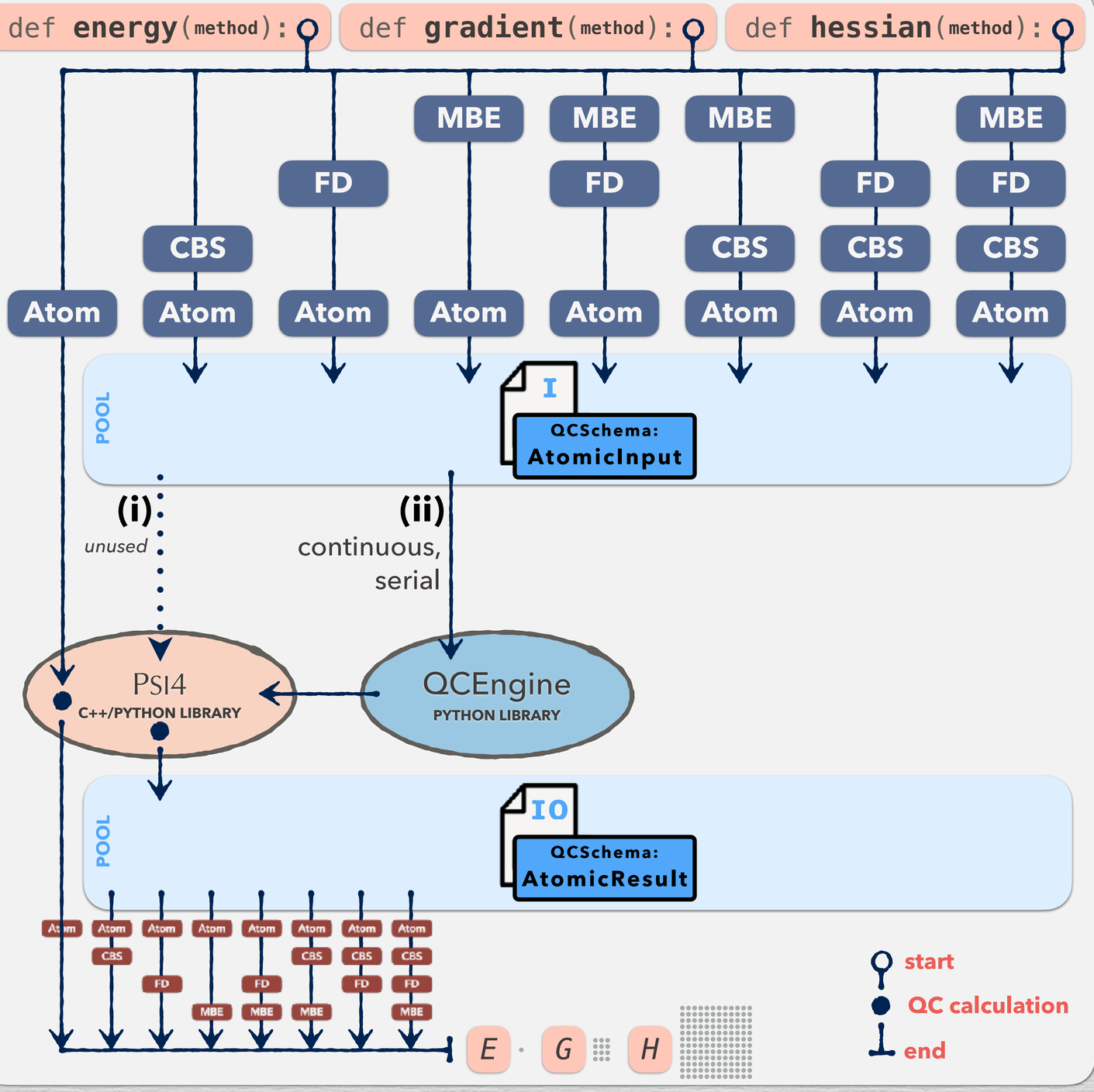
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

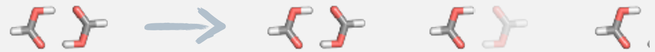
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

```
for disp in displacements: return qcschema
```

QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

```
mp2/cc-pv[tq]z  →  MP2 TOTAL ENERGY/cc-pVTZ
                   MP2 TOTAL ENERGY/cc-pVQZ
for mc in modelchems: return qcschema
```

QCSchema

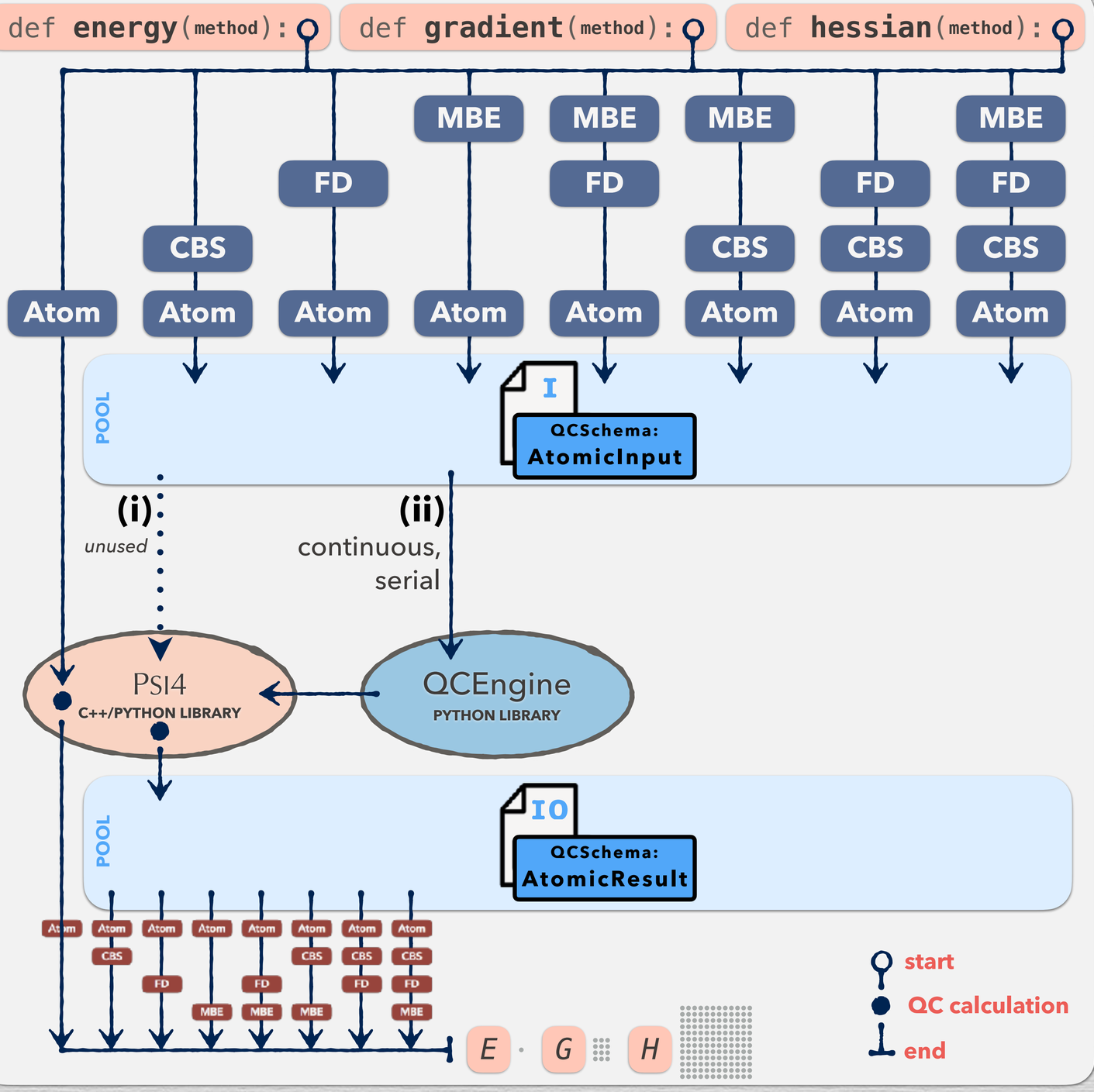
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

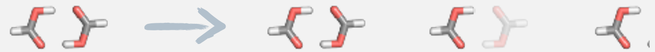
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

```
for disp in displacements: return qcschema
```

QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

```
mp2/cc-pv[tq]z  →  MP2 TOTAL ENERGY/cc-pVTZ
                   MP2 TOTAL ENERGY/cc-pVQZ
for mc in modelchems: return qcschema
```

QCSchema

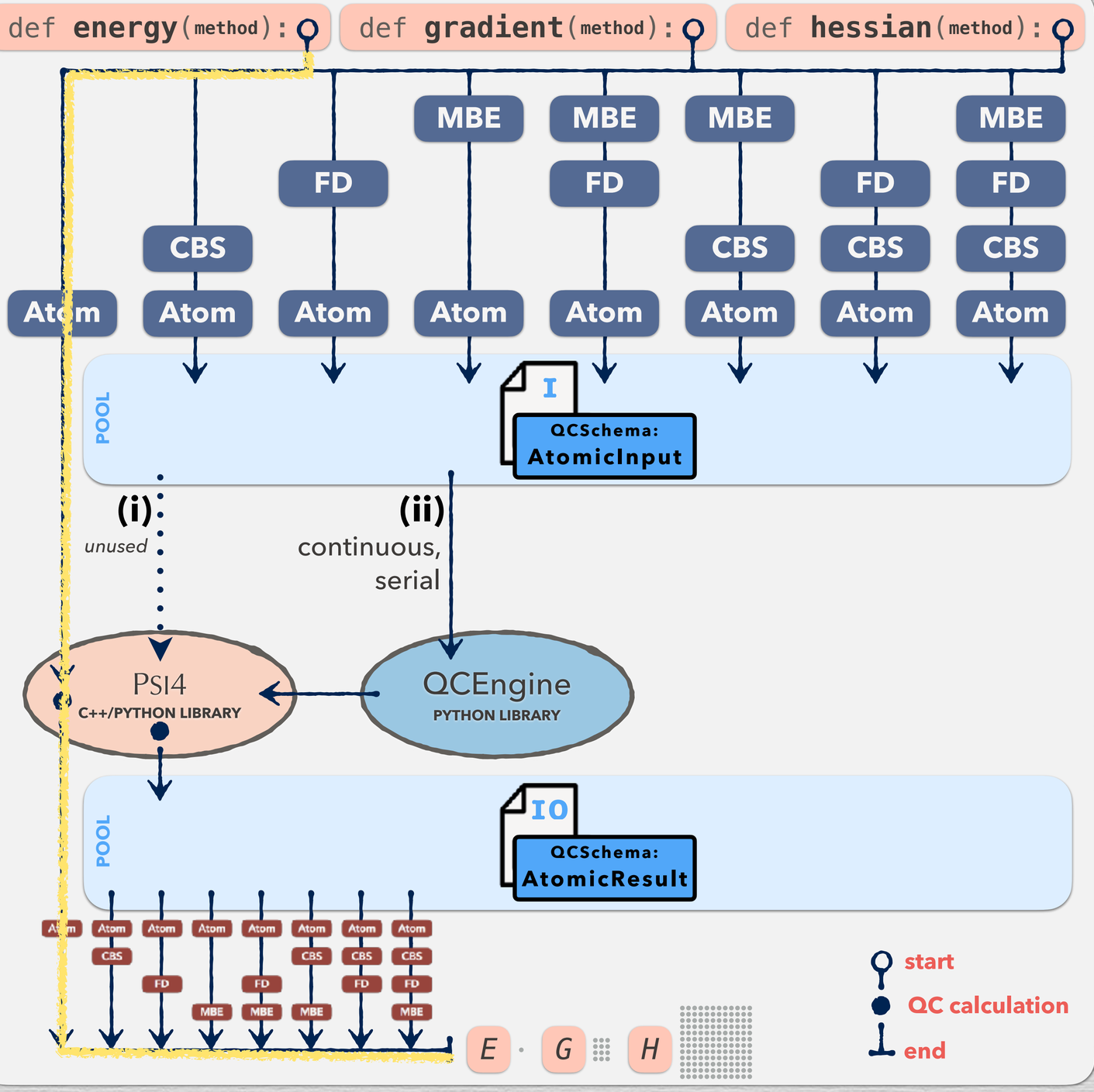
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

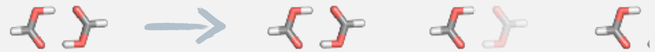
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

```
psi4.energy("mp2/cc-pvdz")
```

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

```
mp2/cc-pv[tq]z  →  MP2 TOTAL ENERGY/cc-pVTZ
                   MP2 TOTAL ENERGY/cc-pVQZ
for mc in modelchems: return qcschema
```

QCSchema

ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER

```
def energy(method):
def gradient(method):
def hessian(method):
```

MBE · FD · CBS · Atom

POOL

I — QCSchema: AtomicInput

(i) *unused*

(ii) continuous, serial

PSI4 — C++/PYTHON LIBRARY

QCEngine — PYTHON LIBRARY

POOL

IO — QCSchema: AtomicResult

start

QC calculation

end

E · G · H

```
class ManyBodyComputer ():
```

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.

for frag in fragments: return qcschema

ASM Assemble n-body & interaction results from fragments.

```
psi4.energy("mp2/cc-pvdz")
```

```
psi4.energy("mp2/cc-pv[dt]z", bsse_type="cp")
```

```
class CompositeComputer ():
```

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ` `MP2 TOTAL ENERGY/cc-pVQZ`

for me in model_chem: return qcschema

ASM Assemble extrapolations & total results from model_chems.

```
class AtomicComputer ():
```

PLAN **molecule** & **method** unchanged. return qcschema

ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER

`def energy(method):` `def gradient(method):` `def hessian(method):`

MBE · FD · CBS · Atom

POOL

I — QCSchema: AtomicInput

(i) *unused*

(ii) continuous, serial

PSI4 — C++/PYTHON LIBRARY

QCEngine — PYTHON LIBRARY

POOL

IO — QCSchema: AtomicResult

Atom · CBS · FD · MBE

E · G · H

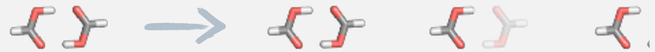
start

QC calculation

end

`class ManyBodyComputer ():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

`class FiniteDifferenceComputer ():`

PLAN Displace **`molecule`** according to stencil. Reference **`molecule`** & **`method`** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

`class CompositeComputer ():`

PLAN Separate **`method`** into method, basis, & extrapolations. **`molecule`** unchanged.

`mp2/cc-pv[tq]z` → **`MP2 TOTAL ENERGY/cc-pVTZ`** **`MP2 TOTAL ENERGY/cc-pVQZ`**

`for mc in modelchems: return qcschema` QCSchema

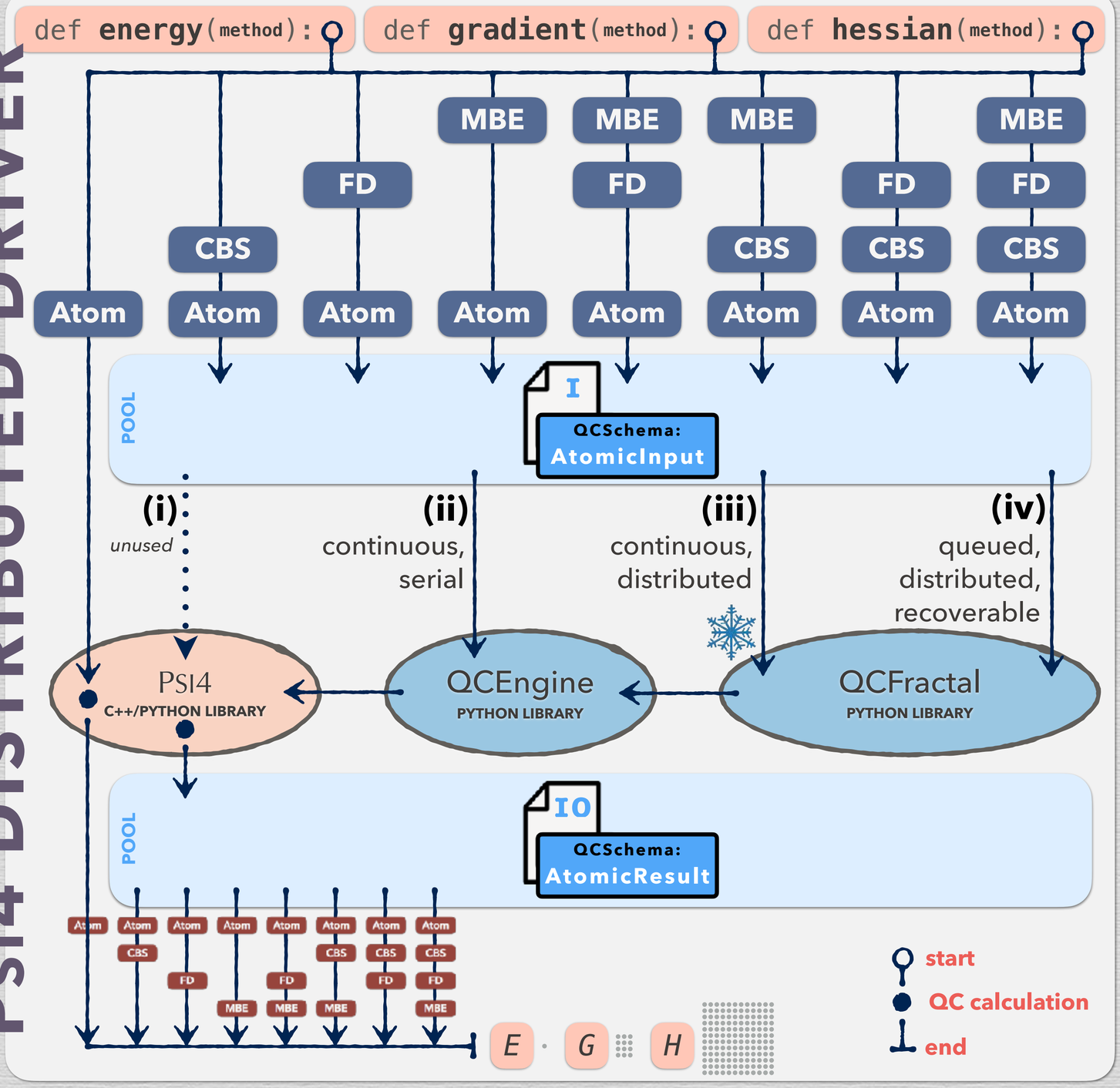
ASM Assemble extrapolations & total results from modelchems.

`class AtomicComputer ():`

PLAN **`molecule`** & **`method`** unchanged. `return qcschema` QCSchema

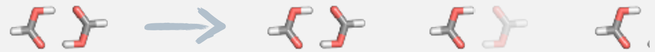
ASM Return analytic energy, gradient, or Hessian.

PSI4 DISTRIBUTED DRIVER



class ManyBodyComputer ():

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



`for frag in fragments: return qcschema` QCSchema

ASM Assemble n-body & interaction results from fragments.

class FiniteDifferenceComputer ():

PLAN Displace **molecule** according to stencil. Reference **molecule** & **method** unchanged.

`for disp in displacements: return qcschema` QCSchema

ASM Assemble derivative results from displacements.

class CompositeComputer ():

PLAN Separate **method** into method, basis, & extrapolations. **molecule** unchanged.

`mp2/cc-pv[tq]z` → `MP2 TOTAL ENERGY/cc-pVTZ`
`MP2 TOTAL ENERGY/cc-pVQZ`

`for mc in modelchems: return qcschema` QCSchema

ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

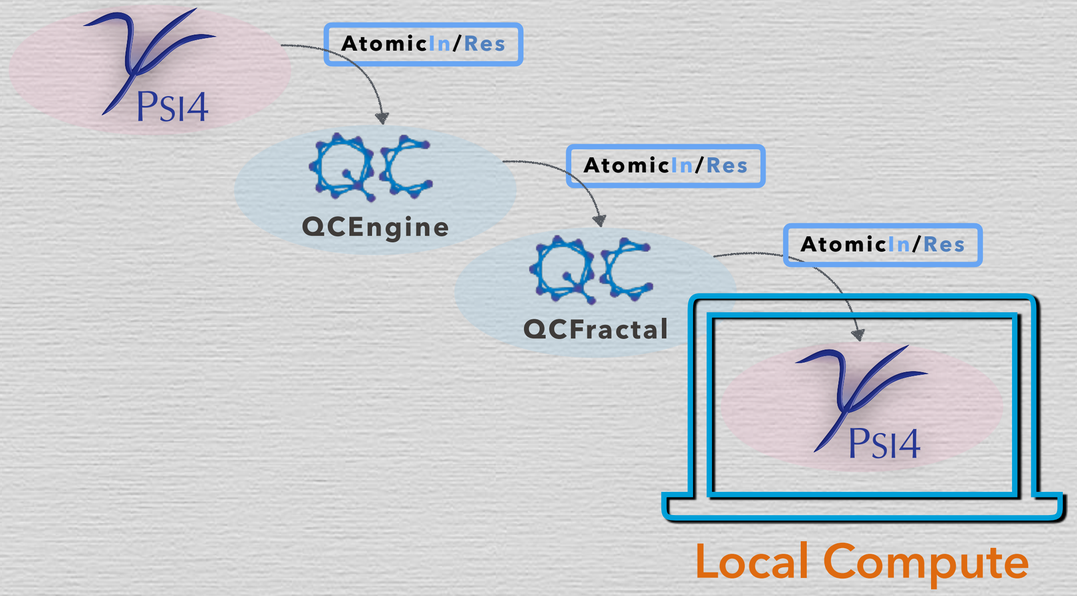
PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

APP: PSI4 DISTRIBUTED DRIVER + QCARCHIVE

distributed, searchable, and error-resistant

```
psi4.energy("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule=  )
```



(4) **PSI4** driver (core) runs single-point jobs

(5) **QCEngine** recieves jobs as **AtomicResult QCSchema**

(3) **QCEngine** recieves jobs as **AtomicInput QCSchema**

(6) **QCFractal** stores jobs in DB & retrives on query

(2) **QCFractal** orchestrates job running among attached resources

(7) **PSI4** driver assembles results into usual API

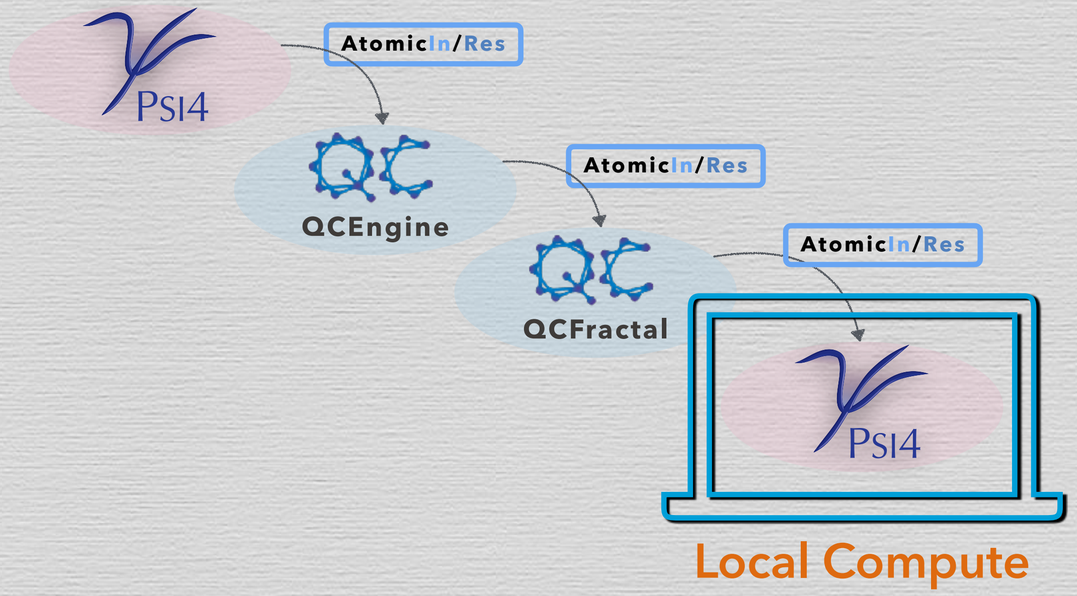
(1) **PSI4** driver (post-proc. manager) plans single-point jobs from API input

- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** limited to PSI4 only.

APP: PSI4 DISTRIBUTED DRIVER + QCARCHIVE

distributed, searchable, and error-resistant

```
psi4.energy("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule=  )
```



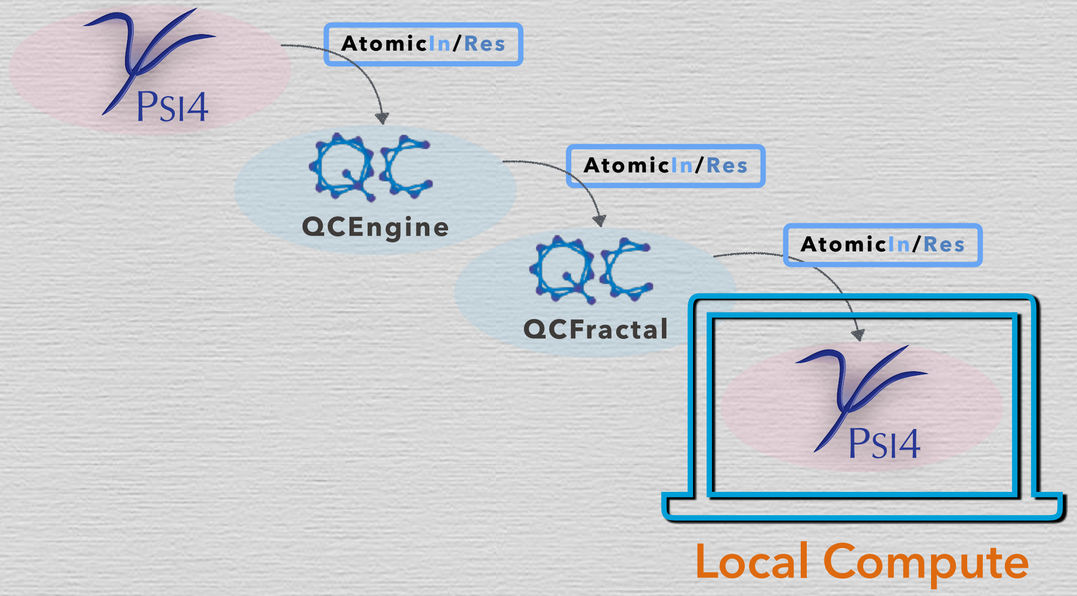

```
import psi4
from qcportal import PortalClient

client = PortalClient("http://localhost:7777")

plan = psi4.energy(
        "MP2/cc-pV[DT]Z + d:CCSD(T)/cc-pVDZ",
        bsse_type="cp",
        return_plan=True,
        molecule=  )
plan.compute(client)

# re-run file after jobs complete for final processing

ene = plan.get_psi4_results(client)
print(ene)  # CP IE
```

- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** limited to PSI4 only.

APP: PSI4 DISTRIBUTED DRIVER + QCARCHIVE

distributed, searchable, and error-resistant

```
psi4.energy("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule=      )
```




```
import psi4
from qcportal import PortalClient

client = PortalClient("http://localhost:7777")

plan = psi4.energy(
        "MP2/cc-pV[DT]Z + d:CCSD(T)/cc-pVDZ",
        bsse_type="cp",
        return_plan=True,
        molecule=      )
plan.compute(client)

# re-run file after jobs complete for final processing

ene = plan.get_psi4_results(client)
print(ene)  # CP IE
```

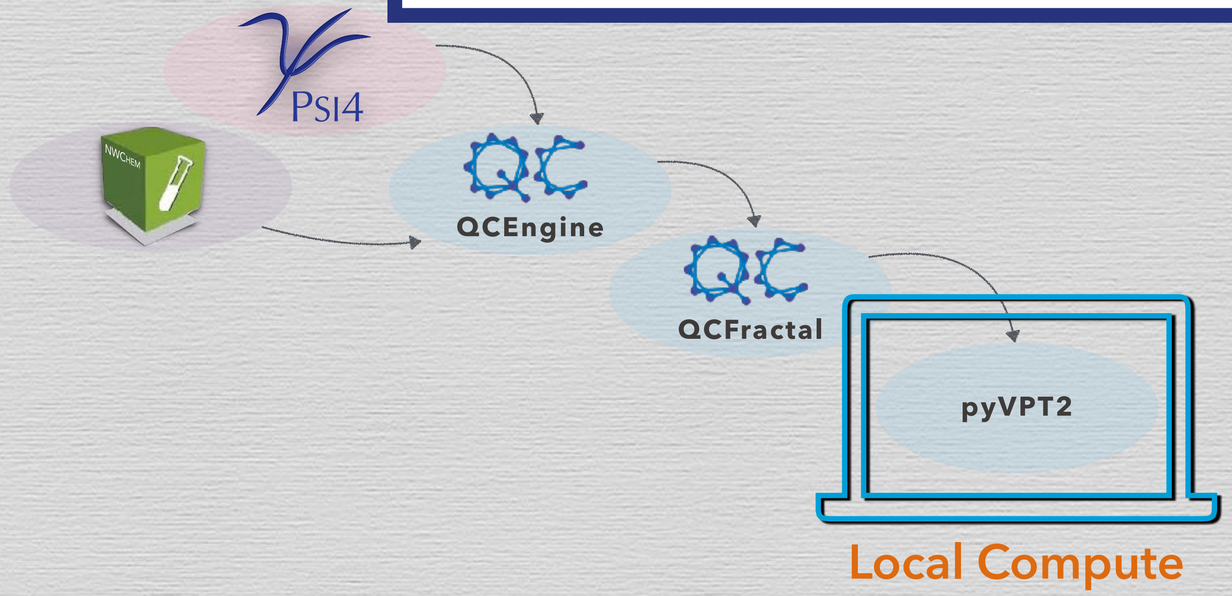
- **SPECIFICATION** through single file. Procedure automated, so low ri
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel i
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** limited to Psi4 only.

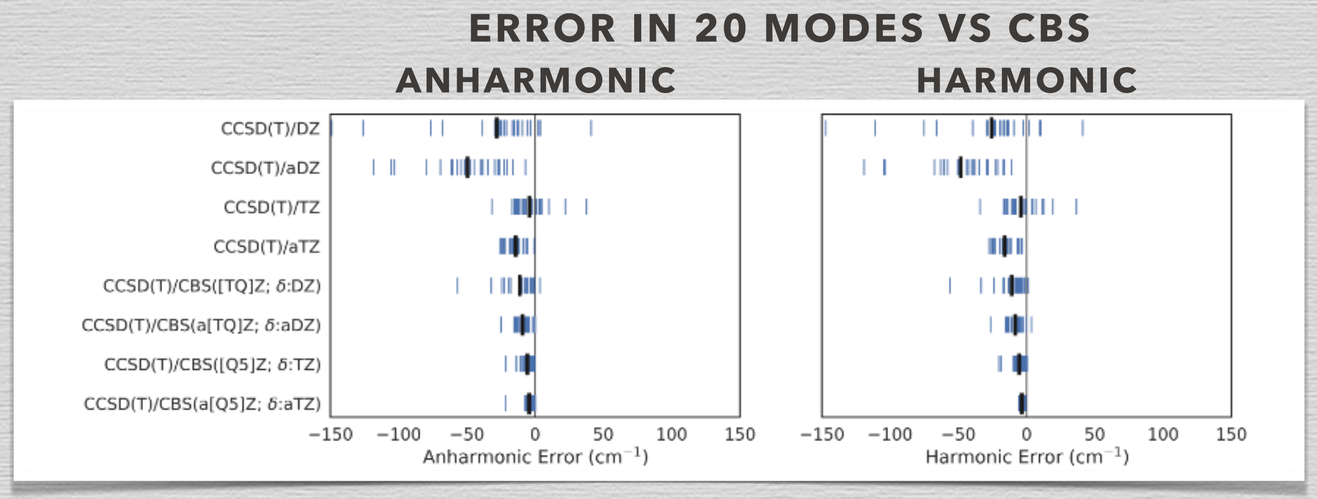
```
plan = psi4.energy("CCSD(T)/cc-pVDZ",
                   return_plan=True,
                   molecule=      )
plan.compute(client)  # free! calcs in database

ene, wfn = plan.get_psi4_results(client, return_wfn=True)
print(ene)  # CCSD(T) energy
```

APP: PSI4 & PYVPT2

using QCSchema for generic access to single-points

```
pyvpt2.vpt2("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", molecule= )
```





- pyvpt2: **AVAILABLE** with pyvpt2 (unreleased) & Psi4 & QCEngine & QCFractal.
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** multiple implementations, calling multiple QC backends.

Philip Nelson
GaTech

Carlos Borca
GaTech → DeepCureAI

Zach Glick
GaTech → Lavo

Derek Metcalf
GaTech → Lavo

APP: CRYSTALATTE

aka Crystal Lattice Energies aka CLE

$$E = \sum_{I}^{N} E_I + \sum_{I<J}^{N} \Delta E_{IJ} + \sum_{I<J<K}^{N} \Delta E_{IJK} + \cdots.$$

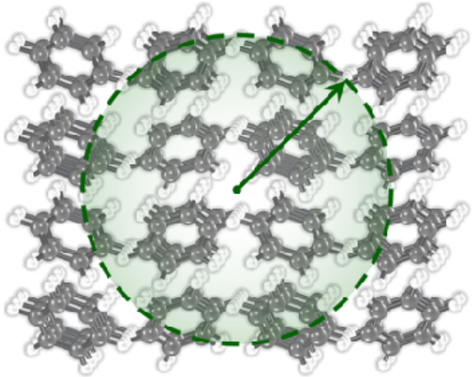


FIG. 2. Illustration of the monomer cutoff. From the center of the supercell, any monomer that includes at least one atom inside a sphere of radius equivalent to the user-specified monomer cutoff will be stored; otherwise, the monomer will be discarded.

$$\mathcal{C}_{N-\mathrm{mer}} = \mathcal{R}_{N-\mathrm{mer}} \times \frac{\Delta E^{(N)}_{I\ldots N}}{N},$$

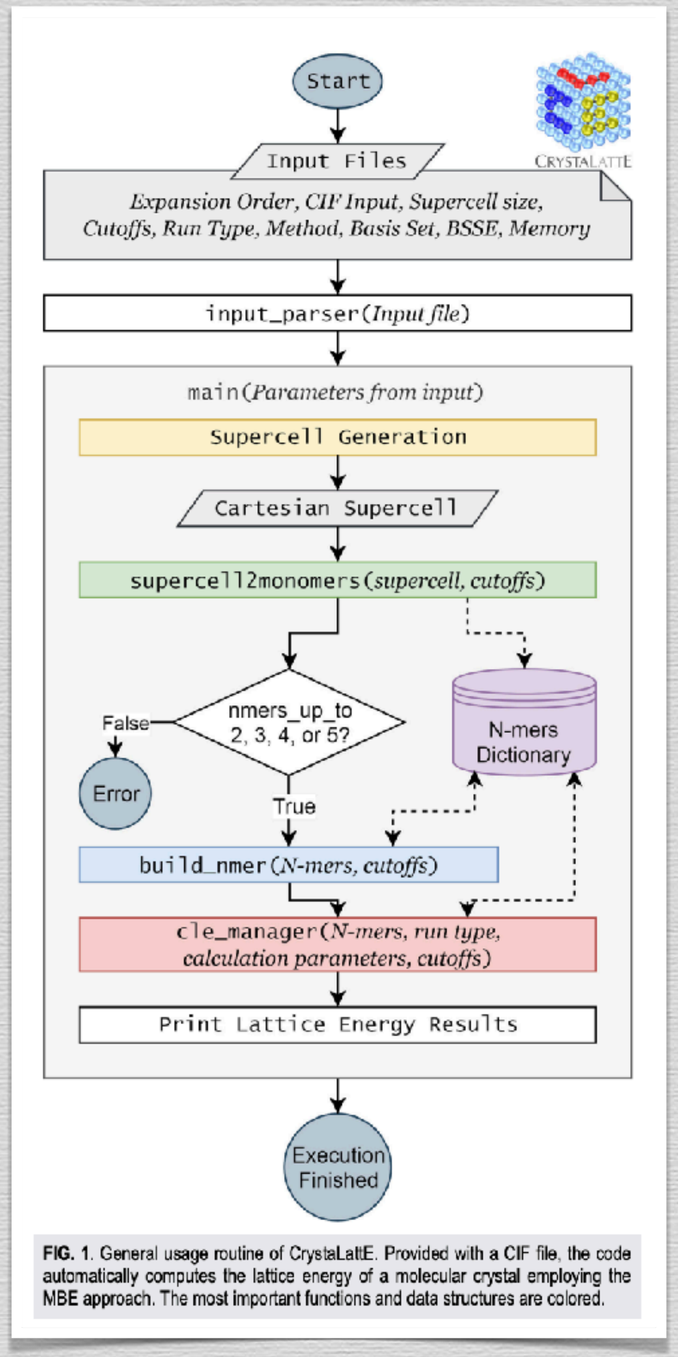


FIG. 1. General usage routine of CrystaLatlE. Provided with a CIF file, the code automatically computes the lattice energy of a molecular crystal employing the MBE approach. The most important functions and data structures are colored.

Borca, Bakr, Burns, & Sherrill, J. Chem. Phys. 151, 144103 (2019).
Borca, Glick, Metcalf, Burns, & Sherrill, J. Chem. Phys. 158, 234102 (2023).

APP: CRYSTALATTE

aka Crystal Lattice Energies aka CLE

Carlos Borca
GaTech → DeepCureAI

Zach Glick
GaTech → Lavo

Derek Metcalf
GaTech → Lavo

$$E = \sum_{I}^{N} E_I + \sum_{I<J}^{N} \Delta E_{IJ} + \sum_{I<J<K}^{N} \Delta E_{IJK} + \cdots.$$

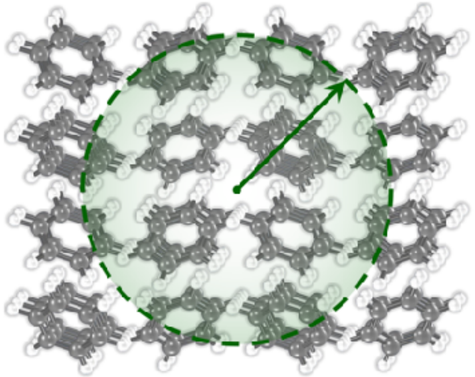


FIG. 2. Illustration of the monomer cutoff. From the center of the supercell, any monomer that includes at least one atom inside a sphere of radius equivalent to the user-specified monomer cutoff will be stored; otherwise, the monomer will be discarded.

$$\mathcal{C}_{N-\mathrm{mer}} = \mathcal{R}_{N-\mathrm{mer}} \times \frac{\Delta E^{(N)}_{I\ldots N}}{N},$$

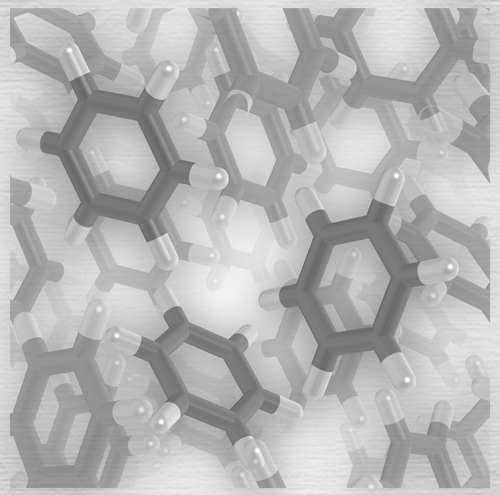


TABLE VI. Benchmark-level results for the lattice energy of crystalline benzene using a many-body approach. Contributions to the lattice energy are given for each non-additive *N*-body term. Only symmetry-unique structures are counted. Cutoff distances are in Å, and energies are in kJ mol^{-1}.[a]

N	Level of theory	Cutoff	Structures	Total contribution
2	CCSD(T)/CBS(aC[Q5]Z; δ:aCTZ)	$R \le 8$	17	−56.39
2	CCSD(T)/CBS(a[TQ]Z; δ:aDZ)	$8 < R \le 30$	403	−1.16
2	Total	$R \le 30$	420	−57.55
3	CCSD(T)/CBS(a[Q5]Z; δ:aTZ)	$R \le 3$	4	2.61
3	CCSD(T)/CBS(a[TQ]Z; δ:aDZ)	$3 < R \le 15$	1973	0.96
3	Total	$R \le 15$	1977	3.57
4	CCSD(T)/CBS(a[TQ]Z; δ:aDZ)	$R \le 5.3$	17	−0.23
	MP2/aTZ	$5.3 < R \le 10$	1315	0.20
4	Total	$R \le 10$	1332	−0.03
	Total			−54.01

estimated experimental value of −55.3 ± 2.2 kJ mol^{-1}

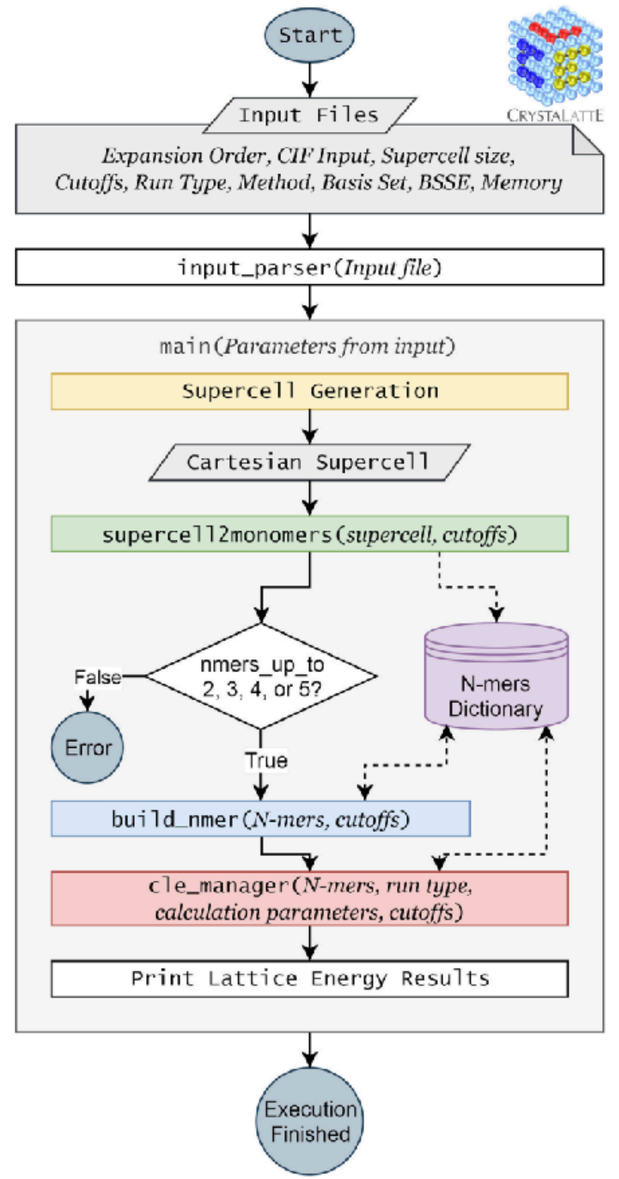


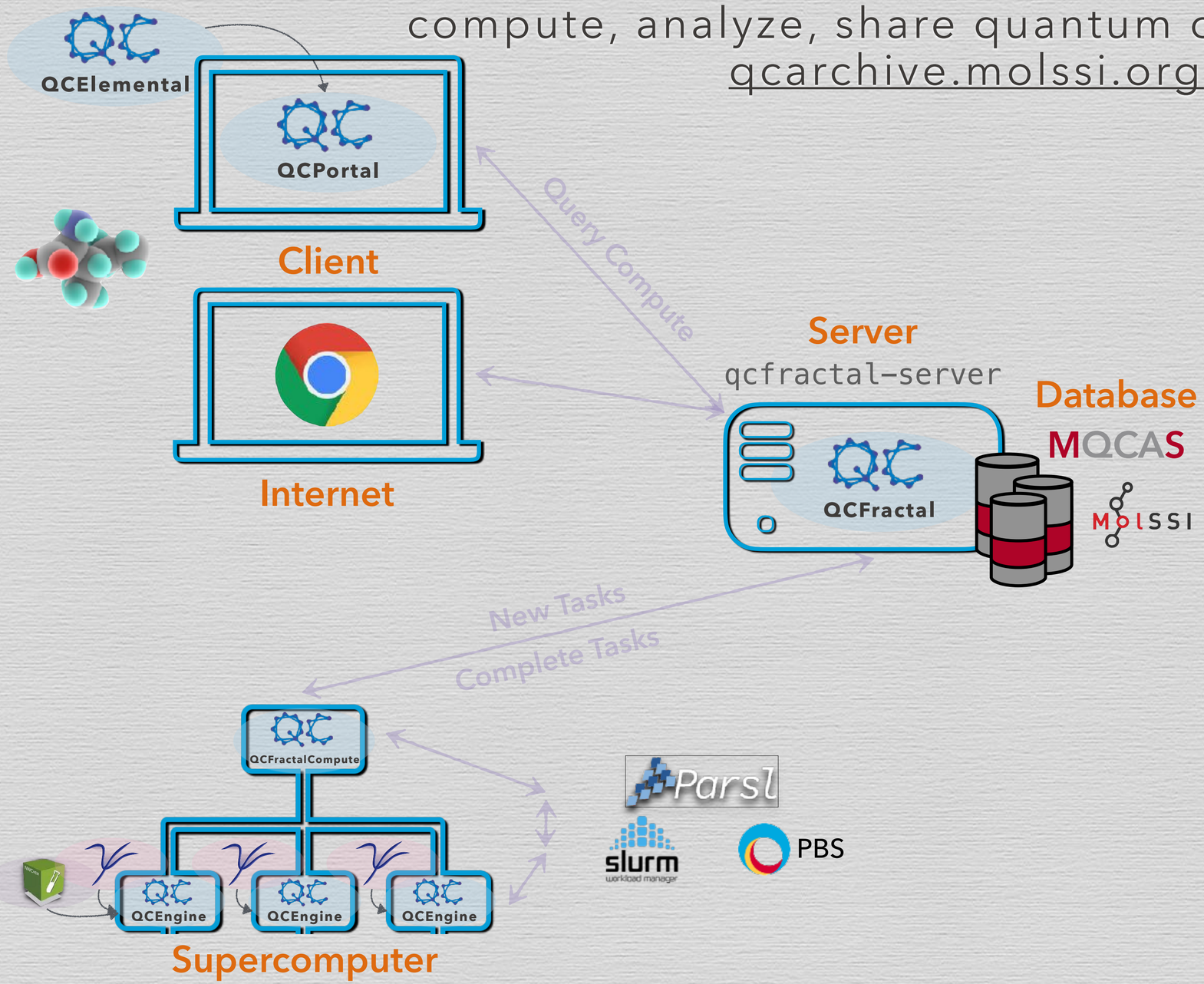
FIG. 1. General usage routine of CrystaLattE. Provided with a CIF file, the code automatically computes the lattice energy of a molecular crystal employing the MBE approach. The most important functions and data structures are colored.

Borca, Bakr, Burns, & Sherrill, J. Chem. Phys. 151, 144103 (2019).
Borca, Glick, Metcalf, Burns, & Sherrill, J. Chem. Phys. 158, 234102 (2023).

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter Task Management

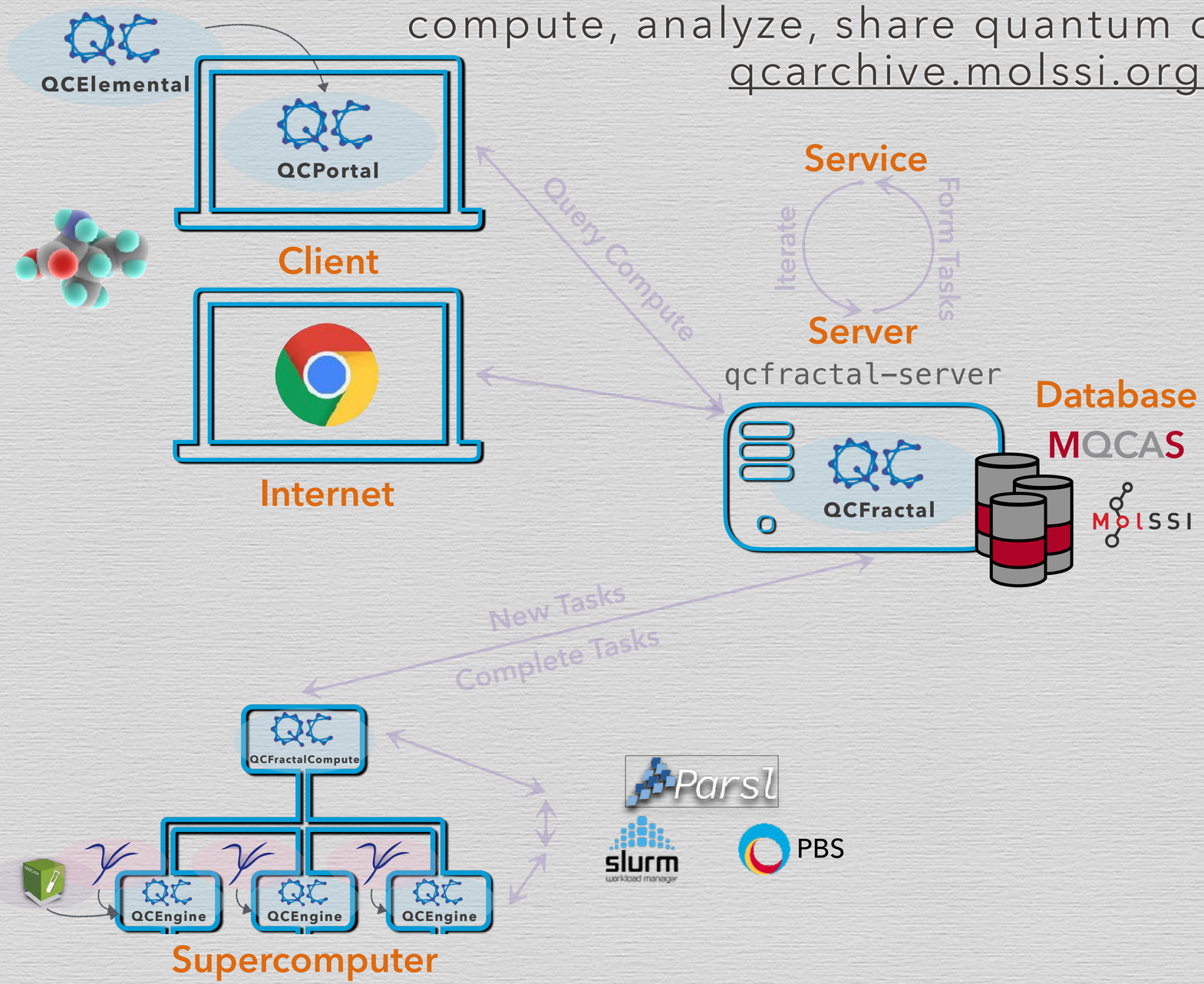
scheduler Task Scheduling

worker Compute Nodes

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE** – up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC** – runs on laptops, campus clusters, and leadership-class supercomputers

`qcfractal-compute-manager` **Manager**

adapter Task Management

scheduler Task Scheduling

worker Compute Nodes

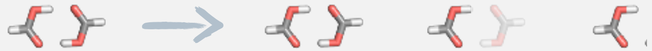
EXTRACTING THE QCMANYBODY PROCEDURE

multiple entry points for flexibility

Psi4

`class` **`ManyBodyComputer`** `():`

PLAN Separate **`molecule`** into subsystems. CP, noCP, VMFC basis. **`method`** unchanged.



`for frag in fragments: return qcschema`

QCSchema

ASM Assemble n-body & interaction results from fragments.

QCManyBody

- noCP, SSFC, VMFC
- energy, gradient, Hess.
- IE or MBE
- indep. or schema intf.
- intermolecular only
- multiple model chemistry MBE

Ben Pritchard
MolSSI

Asem Alenaizan
GaTech → KFUPM

Daniel Smith
MolSSI → Abiologics

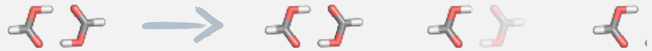
EXTRACTING THE QCMANYBODY PROCEDURE

multiple entry points for flexibility

Psi4

```
class ManyBodyComputer ():
```

PLAN Separate **molecule** into subsystems. CP, noCP, VMFC basis. **method** unchanged.



```
for frag in fragments: return qcschema
```

QCSchema

ASM Assemble n-body & interaction results from fragments.

QCManyBody

- noCP, SSFC, VMFC
- energy, gradient, Hess.
- IE or MBE
- indep. or schema intf.
- intermolecular only
- multiple model chemistry MBE

Ben Pritchard
MolSSI

Asem Alenaizan
GaTech → KFUPM

Daniel Smith
MolSSI → Abiologics

QCManyBody direct

QCFractal service

MBE backend

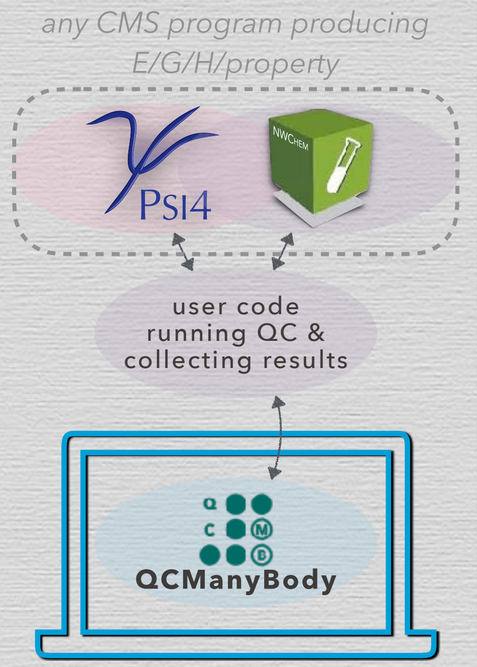
QCEngine procedure

Psi4 distributed driver

Optimizer backend

TWO INTERFACES FOR QCMANYBODY

independent or integrated with QCArchive



CORE INTERFACE

```python
from qcmanybody import ManyBodyCore

mbc = ManyBodyCore(
        molecule= Ne — Ne — Ne,
        bsse_type=["cp"],
        levels={1: "ccsd/cc-pvtz",
                2: "mp2/cc-pvdz",
                3: "mp2/cc-pvdz"},
        return_total_data=True,
        supersystem_ie_only=False,
        embedding_charges=None)

calculation_results = {}
for chem, lbl, imol in mbc.iterate_molecules():
    # use chem, lbl, imol to construct inputs

    # run them & fill in results
    calculation_results[lbl] = {"energy": None}

results = mbc.analyze(calculation_results)

print(results["ret_energy"])
#> -384.1286
```

QCManyBody core interface

- user-prov. serial/parallel
- user runs CMS backends

TWO INTERFACES FOR QCMANYBODY

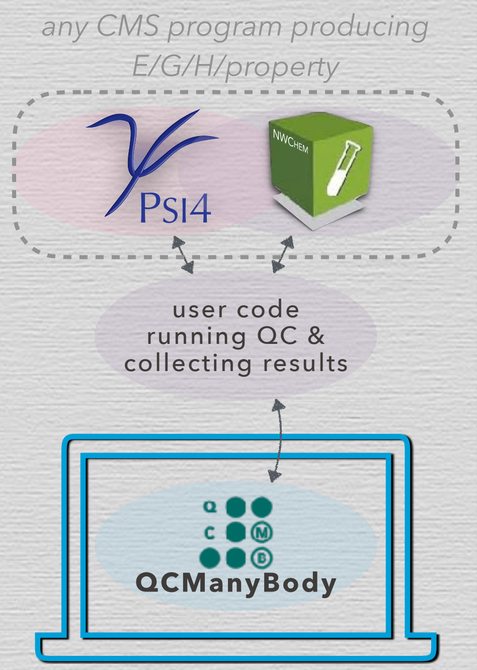
independent or integrated with QCArchive

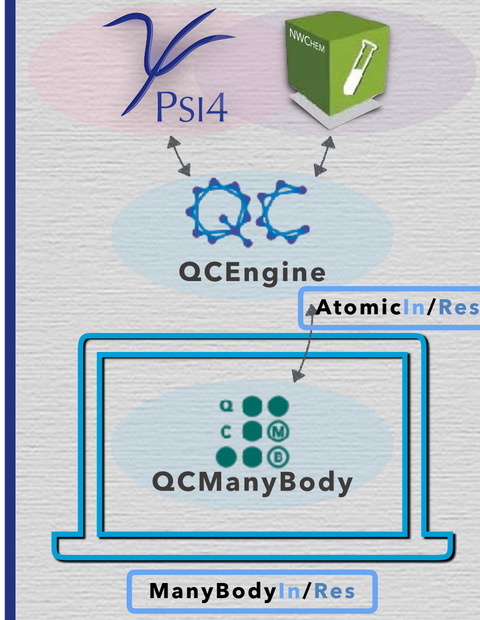
CORE INTERFACE

```python
from qcmanybody import ManyBodyCore

mbc = ManyBodyCore(
        molecule= Ne — Ne — Ne,
        bsse_type=["cp"],
        levels={1: "ccsd/cc-pvtz",
                2: "mp2/cc-pvdz",
                3: "mp2/cc-pvdz"},
        return_total_data=True,
        supersystem_ie_only=False,
        embedding_charges=None)

calculation_results = {}
for chem, lbl, imol in mbc.iterate_molecules():
    # use chem, lbl, imol to construct inputs

    # run them & fill in results
    calculation_results[lbl] = {"energy": None}

results = mbc.analyze(calculation_results)

print(results["ret_energy"])
#> -384.1286
```

HIGH-LEVEL INTERFACE

```python
from qcmanybody import ManyBodyComputer
from qcmanybody.models import ManyBodyInput

mbin = ManyBodyInput(**{
  "molecule": Ne — Ne — Ne,
  "specification": {
    "driver": "energy",
    "keywords": {
      "bsse_type": ["cp"],
      "levels": {
        1: "nwc-ccsd/tz",
        3: "p4-mp2/dz"}},
    "specification": {
      "nwc-ccsd/tz": {
        "program": "nwchem",
        "model": {
          "method": "ccsd",
          "basis": "cc-pvtz"}},
      "p4-mp2/dz": {
        "program": "psi4",
        "model": {
          "method": "mp2",
          "basis": "cc-pvdz"}},
}})

# runs with QCEngine
ret = ManyBodyComputer.from_manybodyinput(mbin)

print(ret.return_result)
#> -384.1286
```





QCManyBody core interface

- user-prov. serial/parallel
- user runs CMS backends

QCManyBody high-level interface

- serial
- multi CMS backends

QCSchema ManyBody

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

ManyBodyInput

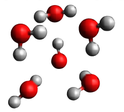
schema_version: 1

provenance: creator name and version info

```
QCManyBody, 0.3.0
```

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodySpecification

schema_version: 1

program: any MBE speaking QCSchema

protocols: customize MBE return layout

ManyBodyProtocols

component_results: save cluster data

extras: free-form storage

driver: MBE target derivative

- energy
- gradient
- Hessian

E *G* *H*

keywords: knobs in the MBE program

ManyBodyKeywords

specification: how to run MBE *subsystems*

- single

AtomicSpecification

- mapping

{hi: **AtomicSpecification**,
lo: **AtomicSpecification** }

ManyBodyInput

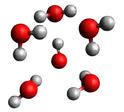
schema_version: 1

provenance: creator name and version info

```
QCManyBody, 0.3.0
```

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyKeywords

bsse_type: any combination of
- nocp/mbe
- cp/ssfc
- vmfc

embedding_charges

return_total_data: IE or total

levels:
- {2: hi, 4: lo}
- {1: hi, 2: md, supersystem: lo}

max_nbody: truncate MBE early

supersystem_ie_only: no MBE analysis

ManyBodySpecification

schema_version: 1

program: any MBE speaking QCSchema

protocols: customize MBE return layout

ManyBodyProtocols

component_results: save cluster data

extras: free-form storage

driver: MBE target derivative
- energy
- gradient
- Hessian

E *G* *H*

keywords: knobs in the MBE program

ManyBodyKeywords

specification: how to run MBE *subsystems*
- single

AtomicSpecification
- mapping

{hi: **AtomicSpecification**,
lo: **AtomicSpecification** }

ManyBodyInput

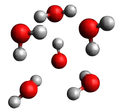
schema_version: 1

provenance: creator name and version info

```
QCManyBody, 0.3.0
```

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

AtomicSpecification

schema_version: 1

program: any SP in QCEngine

protocols: customize return layout

AtomicProtocols

stdout: save text output

wavefunction: save orbital info

native_files: save raw prog. files

error_correction: auto-edit&rerun

extras: free-form storage

driver: single-point derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*
- **properties**

model: model chemistry or FF

method: B3LYP-D3, CCSD(T)

basis: 6-31G*, cc-pVQZ

keywords: knobs in SP program

`{scftyp: uhf, cc_conv: 7, docc:[4,1]}`

ManyBodySpecification

schema_version: 1

program: any MBE speaking QCSchema

protocols: customize MBE return layout

ManyBodyProtocols

component_results: save cluster data

extras: free-form storage

driver: MBE target derivative

- **energy** *E*
- **gradient** *G*
- **Hessian** *H*

keywords: knobs in the MBE program

ManyBodyKeywords

specification: how to run MBE *subsystems*

- single

AtomicSpecification

- mapping

{hi: **AtomicSpecification**,

lo: **AtomicSpecification** }

ManyBodyInput

schema_version: 1

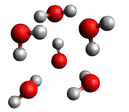
provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:



Molecule

specification: *how* to run MBE single-point

ManyBodySpecification

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyResult

schema_version: 1

input_data: record of starting conditions

ManyBodyInput

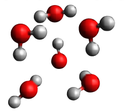
schema_version: 1

provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

success: True

provenance: creator, version, & RT info

`QCManyBody, 0.4.0, psinet, CPU E5-2630`

id: tracker for databases

stdout: program's text output for humans

stderr: anything in error channel

native_files: requested raw program files

extras: free-form storage

molecule: orientation & frame for results

return_result: MBE derivative

properties: key results for MBE

ManyBodyProperties

cluster_properties: key results for subsystems

{cluster id: **AtomicProperties** }

cluster_results: full results for subsystems

{cluster id: **AtomicResult** }

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyProperties (abr.)

calcinfo_nfr: M

return_energy: E

return_gradient: G

***bsse*_corrected_interaction_*egh*_through_*m*_body:**
- defined as E^m_{IE}
- defined for all *bsse* as **bsse_type** allows
- defined for all *egh* as **driver** allows
- def. for all *m* as **max_nbody/levels** allows

***bsse*_corrected_total_*egh*_through_*m*_body:**
- defined as E^m

***bsse*_corrected_*m*_body_contribution_to_*egh*:**
- defined as $E^{(m)}$

ManyBodyResult

schema_version: 1

input_data: record of starting conditions

ManyBodyInput

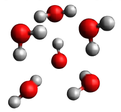
schema_version: 1

provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

success: True

provenance: creator, version, & RT info

`QCManyBody, 0.4.0, psinet, CPU E5-2630`

id: tracker for databases

stdout: program's text output for humans

stderr: anything in error channel

native_files: requested raw program files

extras: free-form storage

molecule: orientation & frame for results

return_result: MBE derivative

E G H

properties: key results for MBE

ManyBodyProperties

cluster_properties: key results for subsystems

{cluster id: **AtomicProperties** }

cluster_results: full results for subsystems

{cluster id: **AtomicResult** }

QCSchema MANYBODY

driving the many-body expansion

$$E^M = E^1 + E^{(2)} + E^{(3)} + \cdots + E^{(M)},$$
$$= \sum_{I \in \mathcal{P}_1} E^I + \sum_{IJ \in \mathcal{P}_2} E^{(IJ)} + \sum_{IJK \in \mathcal{P}_3} E^{(IJK)} + \cdots,$$

- **RECENT** added in 2024
- Counterpoise, VMFC, and no-CP arbitrary-order manybody expansion (**MBE**) extracted from Psi4 program
- **SUITABLE** for intermolecular fragments, no bond-breaking

ManyBodyResult

schema_version: 1

input_data: record of starting conditions

ManyBodyInput

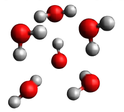
schema_version: 1

provenance: creator name and version info

`QCManyBody, 0.3.0`

id: tracker for databases

extras: free-form storage

molecule:  **Molecule**

specification: *how* to run MBE single-point

ManyBodySpecification

success: True

provenance: creator, version, & RT info

`QCManyBody, 0.4.0, psinet, CPU E5-2630`

id: tracker for databases

stdout: program's text output for humans

stderr: anything in error channel

native_files: requested raw program files

extras: free-form storage

molecule: orientation & frame for results

return_result: MBE derivative

properties: key results for MBE

ManyBodyProperties

cluster_properties: key results for subsystems

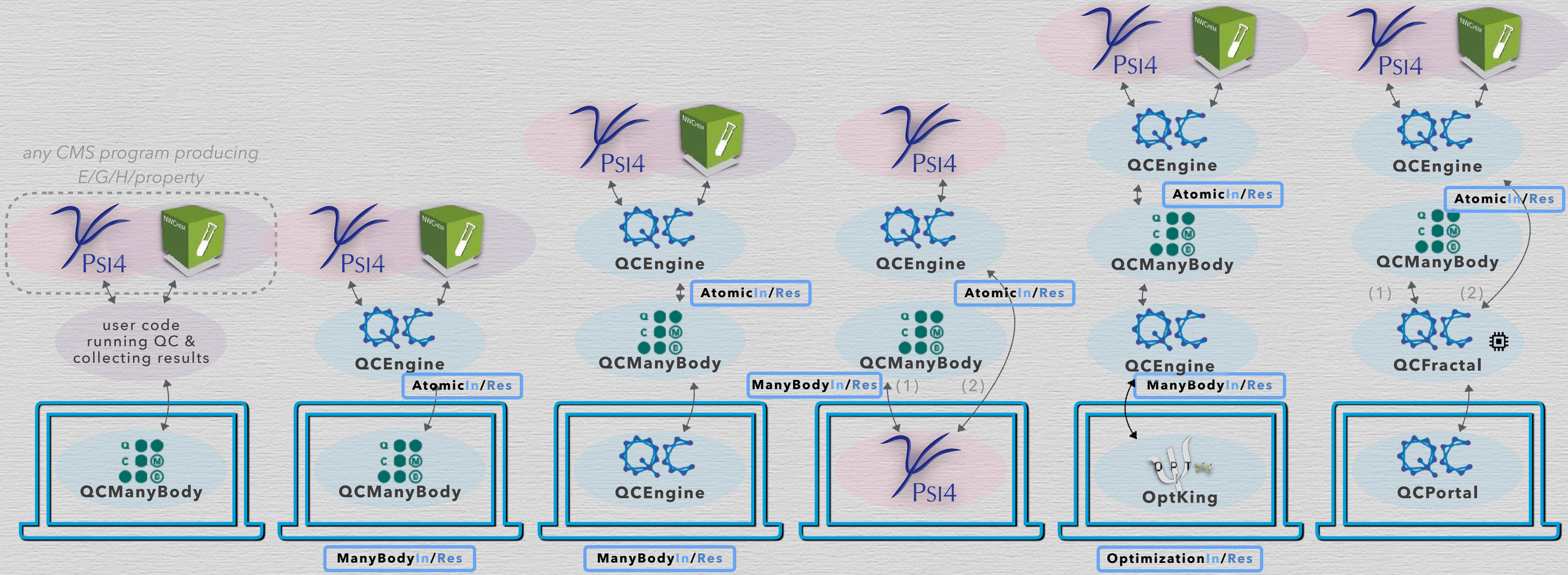
{cluster id: **AtomicProperties** }

cluster_results: full results for subsystems

{cluster id: **AtomicResult** }

INTEGRATING QCMANYBODY

multiple entry points for flexibility



QCManyBody core interface

- user-prov. serial/parallel
- user runs CMS backends

QCManyBody high-level interface

- serial
- multi CMS backends

QCEngine procedure

- serial
- multi CMS backends

PSI4 distributed driver

- serial or parallel
- only Psi4 backend

Optimizer backend

- serial
- multi CMS backends

QCFractal service

- parallel
- multi CMS backends

CRYSTALATTE + QCARCHIVE

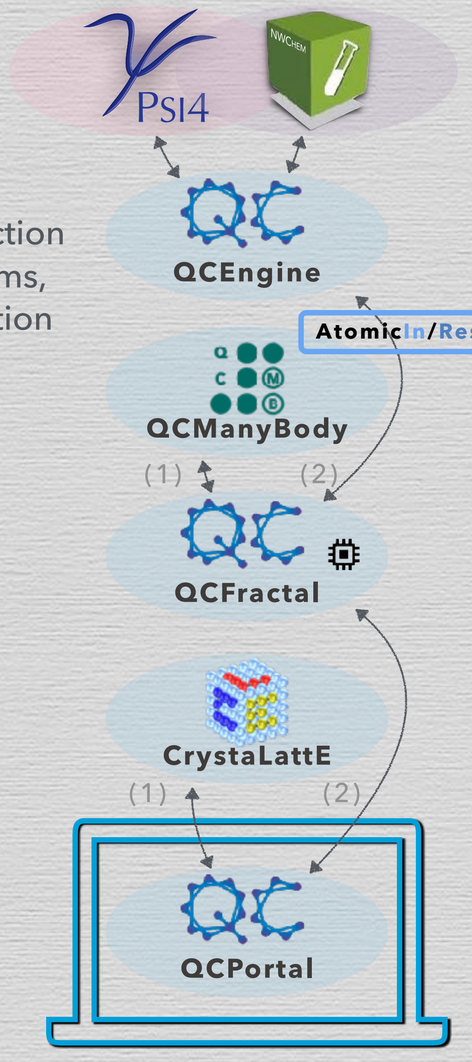
fast lattice energies

Philip Nelson
GaTech

Ben Pritchard
MolSSI

$$\mathcal{C}_{N-\mathrm{mer}} = \mathcal{R}_{N-\mathrm{mer}} \times \frac{\Delta E^{(N)}_{I\ldots N}}{N},$$

- **BEN** has added QCFractal service to work with CrystaLattE (in a development version)
- **QCPORTAL** calls CLE, collects all the jobs that previously would have been sent to PSI4, forms a collection of molecules at each interaction order (dimers, trimers, etc.), calls QCManyBody to get the subsystems, run those in parallel, allowing the database to de-duplicate, calls QCManyBody to assemble interaction energies for each dimer, etc., the performs the final analysis summary to form the lattice energy.
- **FINALLY**, multiple QC backends for CLE and automated parallelization



QCFractal frontend to CrystaLattE

- uses QCMB core intf
- parallel
- multi CMS backends

Burns, Sherrill, & Pritchard, J. Chem. Phys. 161, 152101 (2024).

Philip Nelson
GaTech

Ben Pritchard
MolSSI

CRYSTALATTE + QCARCHIVE

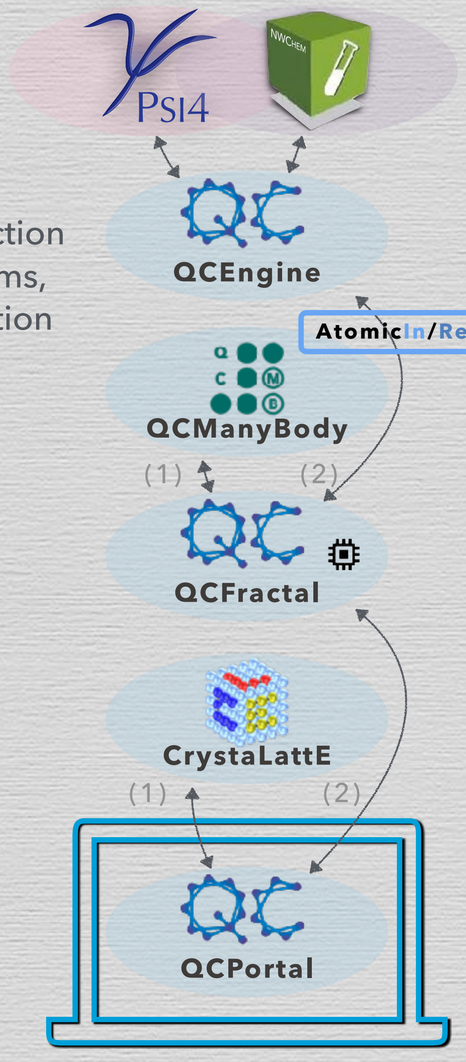
fast lattice energies

$$\mathcal{C}_{N\text{-mer}} = \mathcal{R}_{N\text{-mer}} \times \frac{\Delta E^{(N)}_{I\ldots N}}{N},$$

- **BEN** has added QCFractal service to work with CrystaLattE (in a development version)
- **QCPORTAL** calls CLE, collects all the jobs that previously would have been sent to PSI4, forms a collection of molecules at each interaction order (dimers, trimers, etc.), calls QCManyBody to get the subsystems, run those in parallel, allowing the database to de-duplicate, calls QCManyBody to assemble interaction energies for each dimer, etc., the performs the final analysis summary to form the lattice energy.
- **FINALLY**, multiple QC backends for CLE and automated parallelization

```python
from qcportal import PortalClient
from qcportal.singlepoint import QCSpecification
from qcportal.external import crystalatte as cle

c = PortalClient.from_file('molssi_testing', '/home/ben/work/qcarchive/qcportal_config.yaml')
#> PortalClient(server_name='MolSSI QCFractal MB Testing Server', ...)

cle.create_datasets(c, "Philip Reproduce 1", "cif/acetic_acid.cif",
    [120.0, 20.0, 20.0],          # CLE monomer, dimer, trimer cutoffs,
    QCSpecification(              # how to run single-points
        program="psi4",
        driver="energy",
        method="hf",
        basis="cc-pvtz",
        keywords={"d_convergence"},
        protocols={"stdout": False}),
    ['cp'],                       # BSSE correction,
    {'return_total_data': True},  # keywords for QCManyBody
)

ds1 = c.get_dataset("manybody", "Philip Reproduce 1: dimers")
ds2 = c.get_dataset("manybody", "Philip Reproduce 1: trimers")
ds1.submit()
ds2.submit()

# wait 6h to run 215k calcs on 12 nodes
```

Burns, Sherrill, & Pritchard, J. Chem. Phys. 161, 152101 (2024).



QCFractal frontend to CrystaLattE

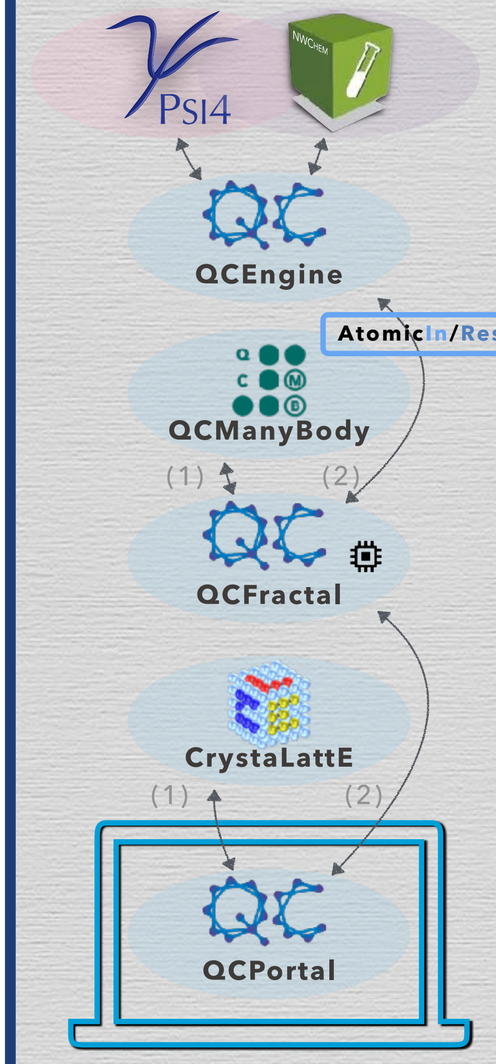
- uses QCMB core intf
- parallel
- multi CMS backends

CRYSTALATTE + QCARCHIVE

fast lattice energies

```
    ['cp'],                          # BSSE correction,
    {'return_total_data': True},     # keywords for QCManyBody
)

ds1 = c.get_dataset("manybody", "Philip Reproduce 1: dimers")
ds2 = c.get_dataset("manybody", "Philip Reproduce 1: trimers")
ds1.submit()
ds2.submit()

# wait 6h to run 215k calcs on 12 nodes

ds1.print_status()
#>   specification    complete
#> ---------------  ----------
#>         default         314

ds2.print_status()
#>   specification    complete
#> ---------------  ----------
#>         default       30546

results = cle.analyze_datasets(c, 'Philip Reproduce 1', 'default')
#> Found datasets:
#>   [dimers   ] Philip Reproduce 1: dimers
#>   [trimers  ] Philip Reproduce 1: trimers

s = cle.results_summary_str(results)
#> N-mer Name        Non-Additive    Num.          N-mer           Partial     Calculation  Minimum
#>                      MB Energy    Rep.    Contribution           Crystal        Priority  Monomer
#>                       (kJ/mol)                 (kJ/mol)   Lattice Energy    (Arb. Units)  Separations
#>                                                                   (kJ/mol)                (A)
#> --------------  --------------  ------  --------------  -----------------  --------------  --------------------
#> 2mer-0+5          -30.46320108       2     -3.0463e+01          -30.4632     0.0105637    1.651
#> 2mer-0+2            2.89236795       2      2.8924e+00          -27.5708     0.00159929   2.646
#> 2mer-0+1            0.02910461       2      2.9105e-02          -27.5417     0.00146094   2.707
#> ...
#> 3mer-0+128+463      0.00000598       3      5.9819e-06          -32.8002     6.439e-15    11.783 19.965 19.971
#> 3mer-0+285+549      0.00001801       3      1.8006e-05          -32.8002     6.439e-15    14.453 19.965 19.971
#> 3mer-0+20+552      -0.00043743       3     -4.3743e-04          -32.8006     6.439e-15    6.898 19.966 19.971
#>
#> Crystal Lattice Energy (kJ/mol)   = -32.80060084
#> Crystal Lattice Energy (kcal/mol) = -7.83953175
```

Burns, Sherrill, & Pritchard, J. Chem. Phys. 161, 152101 (2024).



QCFractal frontend to CrystaLattE

- uses QCMB core intf
- parallel
- multi CMS backends

Psi4

Jerome Gonthier
Berkeley → QC Ware

Rob Parrish
Stanford → QC Ware

Holger Kruse
Czech Acad. Sci.

Rollin King
Bethel

Alexander Sokolov
Ohio State

David Sherrill
GaTech

Lori Burns
GaTech

Asim Alenaizan
KFUPM

Maximilian Scheurer
Heidelberg → Covestro

Zach Glick
GaTech → Lavo

Jeff Schriber
Iona

Francesco Evangelista
Emory

Eugene DePrince
FSU

Fritz Schaefer
UGA

Justin Turney
UGA

Daniel Crawford
VaTech

Daniel Smith
MolSSI → Abiologics

Konrad Patkowski
Auburn

Ben Pritchard
MolSSI

Jonathon Misiewicz
VaTech

Ed Valeev
VaTech

Andy Simmonett
NIH → QC Ware

Ed Hohenstein
Stanford → QC Ware

Roberto Di Remigio
ENCCS

Ugur Bozkaya
Hacettepe

Peter Kraus
TUB

Susi Lehtola
Helsinki

QCENGINE

ADCC

Maximilian Scheurer
Heidelberg

Michael Herbst
EPFL

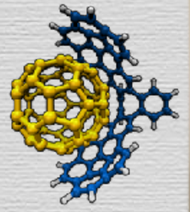
Andreas Dreuw
Heidelberg

CFOUR

John Stanton
UFL

Devin Matthews
SMU

SE / DISP.



Sebastian Ehlert
Bonn

Holger Kruse
Czech Acad. Sci.

Jiří Šponer
Czech Acad. Sci.

GAMESS

Mark Gordon
Iowa State

Nuwan de Silva
W. New England U

ML

Farhad Ramezanghorbani
UFL → Schrödinger

MOLPRO

Sebastian Lee
CalTech

MRCHEM

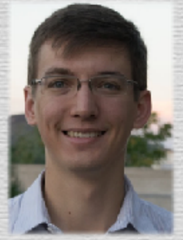
Roberto Di Remigio
Uppsala

NWCHEM

Theresa Windus
Iowa State

Jiyoung Lee
U Texas

Annabelle Lolinco
Iowa State



Logan Ward
Argonne

Adrian Hurtado
Stony Brook

OPENFF / MM

John Chodera
MSKCC

Jeffrey Wagner
UC Irvine

David Dotson
UC Boulder

Joshua Horton
Newcastle

Jamshed Anwar
Lancaster

TERACHEM

Fang Liu
Emory

Heather Kulik
MIT

Colton Hicks
Stanford

Todd Martínez
Stanford

TURBOMOLE

Johannes Steinmetzer
Friedrich Schiller U

HIST./DB/TEST

Nick Petosa
GaTech → Two Sigma

Daniel Nascimento
U Memphis

Dom Sirianni
GaTech

Chaya Stern
MSKCC

ENGINES

Taylor Barnes
MolSSI

OPTIMIZERS

Lee-Ping Wang
UC Davis

Jan Hermann
Free U Berlin

Alexander Heide
UGA

Rollin King
Bethel

PROCEDURES

Asim Alenaizan
GaTech → KFUPM

Peter Kraus
Curtin

Jonathon Misiewicz
Emory

Jeff Schriber
GaTech → Iona

Carlos Borca
Princeton

Philip Nelson
GaTech

QCArchive Acknowledgements

QCArchive

Daniel Smith
MolSSI → Abiologics

Levi Naden
MolSSI

Doaa Altarawy
MolSSI → Alexandria

Matthew Welborn
MolSSI → Entos

Ben Pritchard
MolSSI

MolSSI

Theresa Windus
Iowa State

Daniel Crawford
VaTech

Jessica Nash
MolSSI

Sam Ellis
MolSSI

Susi Lehtola
Helsinki

Open Force Field Consortium

openforcefield.org

psicode.org

molssi.github.io/QCFractal

QCSchema

Daniel Smith
MolSSI → Abiologics

Aaron Virshup
Arzeda

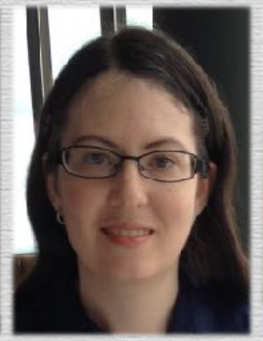
Bert de Jong
LBNL

Geoff Hutchison
U Pittsburgh

Marcus Hanwell
Kitware → Voltron

Eric Berquist
Sandia

Ben Pritchard
MolSSI



Lori Burns
GaTech

Matthew Welborn
MolSSI → Entos

Colton Hicks
Stanford