Composing and decomposing quantum chemistry software: Adventures with Psi4 and QCArchive

LORI A. BURNS

ELECTRONIC STRUCTURE IN OPEN SCIENCE SYMPOSIUM

AMERICAN PHYSICAL SOCIETY MARCH MEETING, LAS VEGAS, NEVADA

7 MARCH 2023

MOLSSI

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

QC PROGRAMS

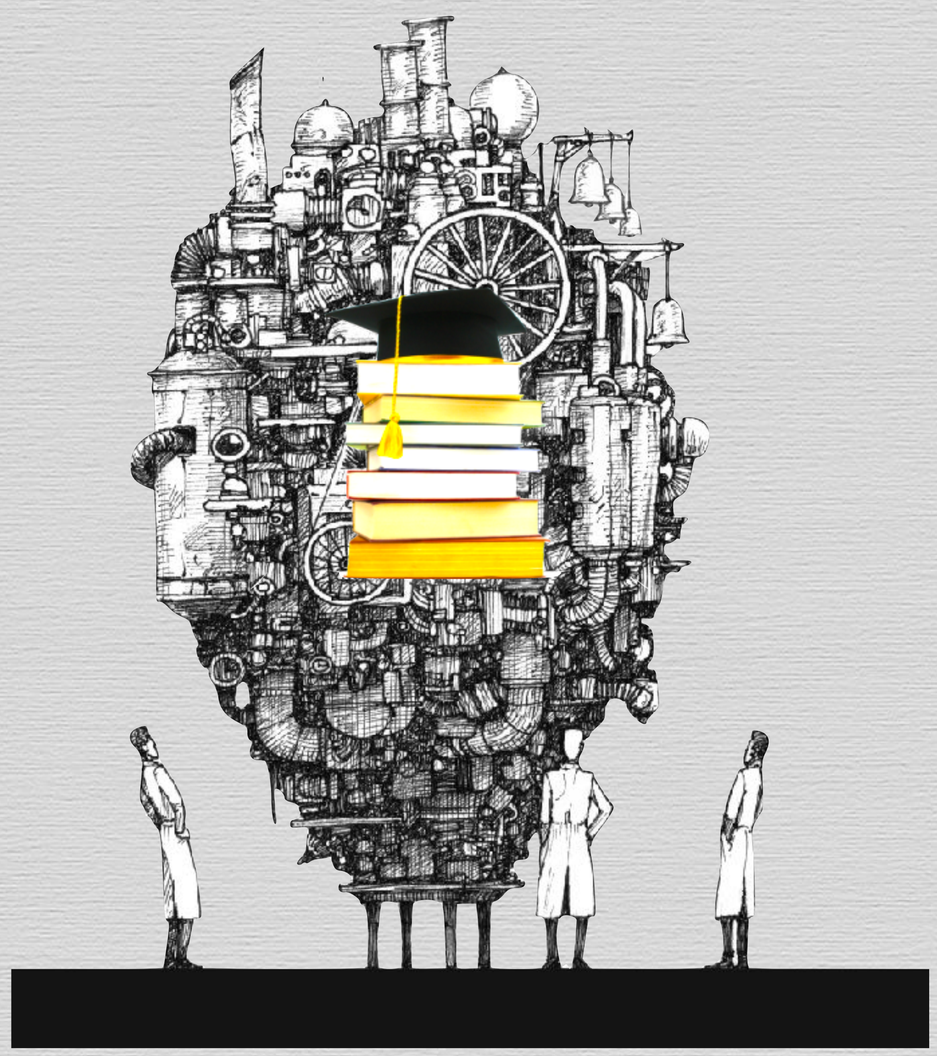
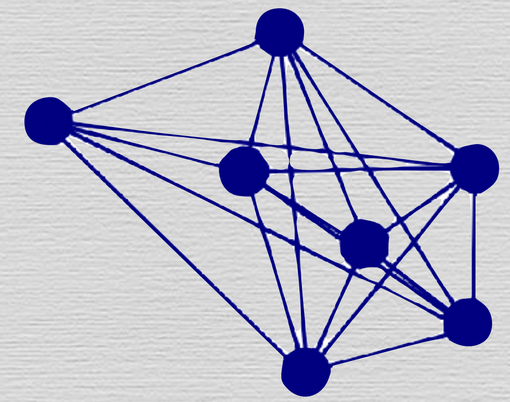


image from https://patientsafe.wordpress.com/2016/05/29/the-healthcare-safety-machine-is-broken/

REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

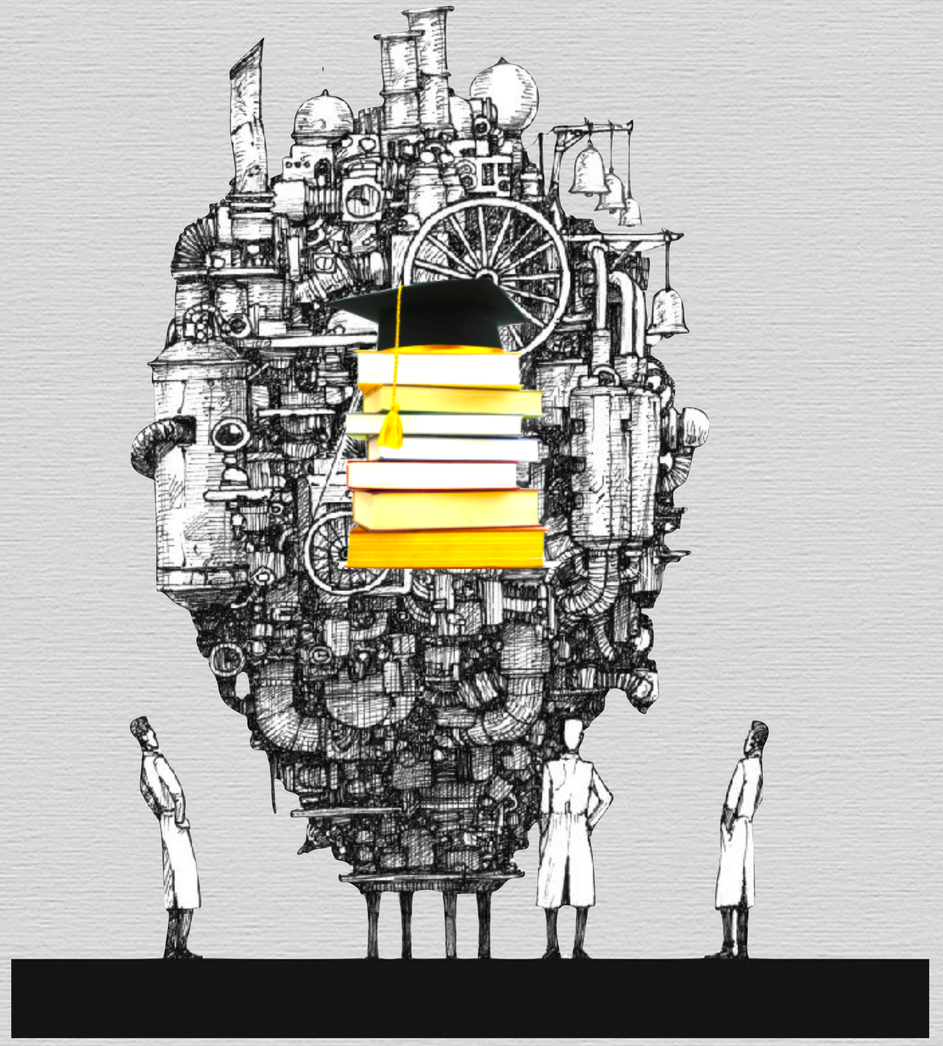
STRONG COUPLING



LOW COHESION

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

QC PROGRAMS

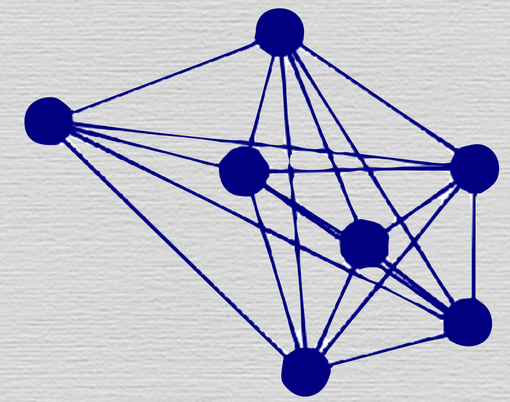


Slide LHS taken from Managing Complexity and Modular Code Development by Radovan Bast

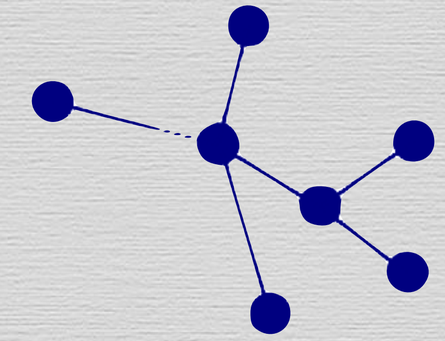
REFACTORING AN ECOSYSTEM

prefer loose coupling and high cohesion

STRONG COUPLING



LOOSE COUPLING



- easier to understand
- easier to compose

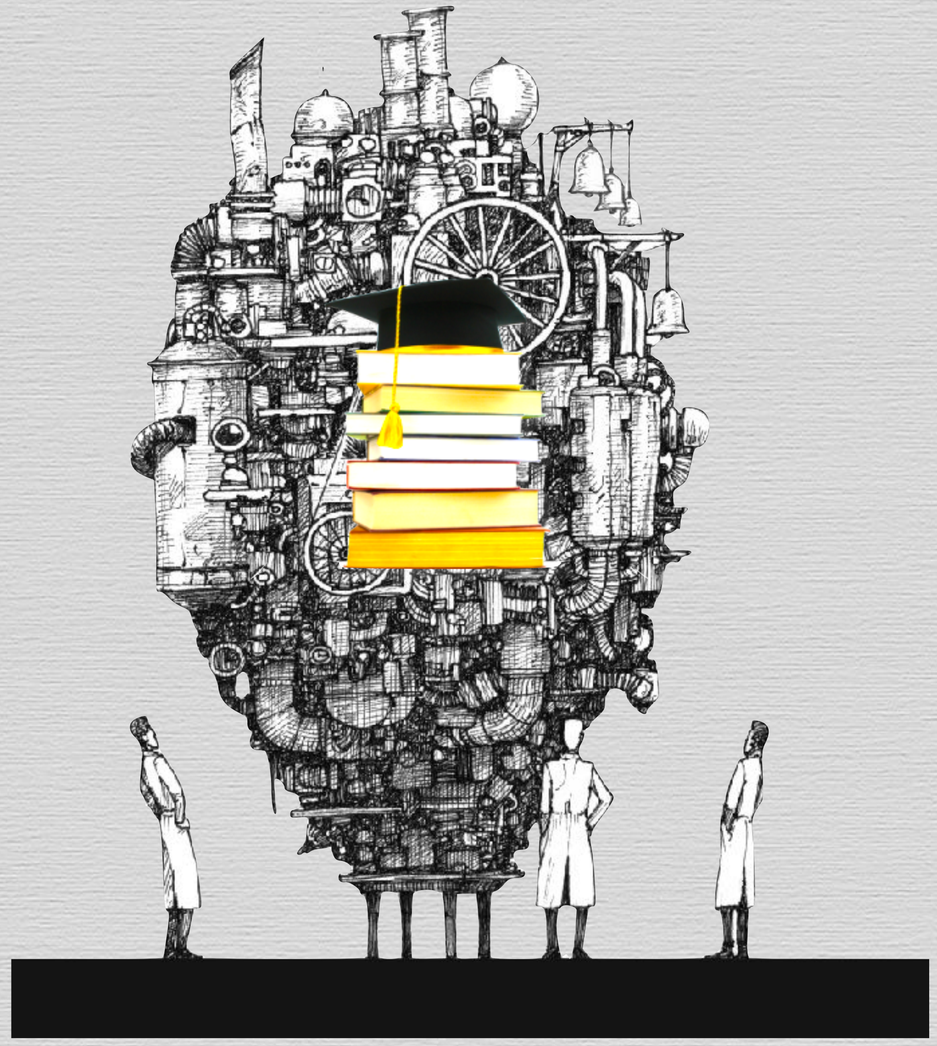
LOW COHESION

- 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 difference 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 & multi-ref. PT2 • arb.-order CC & CI
active-space CC & CR-CC
ADC methods • GW
linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & fragment (EFP) potentials
relativistic contrib. • solvation corrections
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
fast Fourier transform

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

QC PROGRAMS

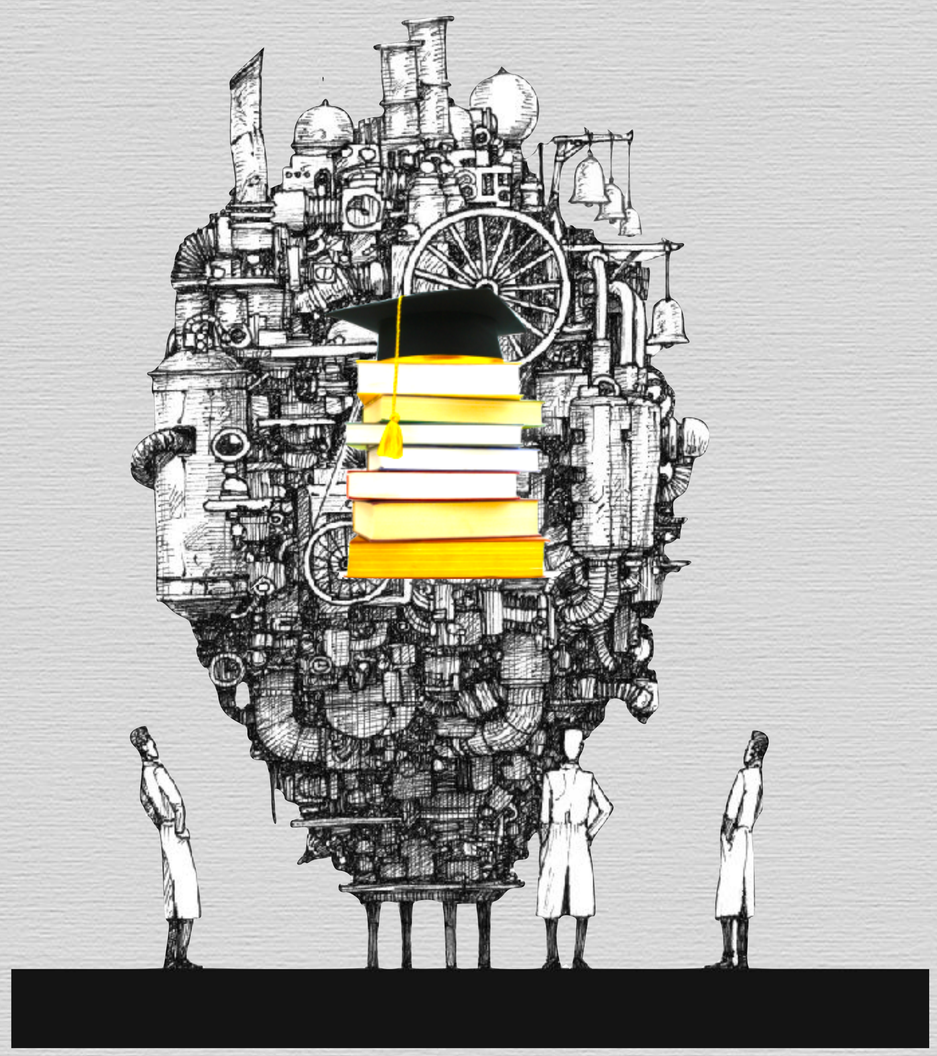


figure from Burns, Oliveira, Pritchard, Slipchenko

QC PROGRAMS

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

processors of whole calcs
finite difference 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 & multi-ref. PT2 • arb.-order CC & CI
active-space CC & CR-CC
ADC methods • GW
linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & fragment (EFP) potentials
relativistic contrib. • solvation corrections
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
fast Fourier transform

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

minimal desired monolithification

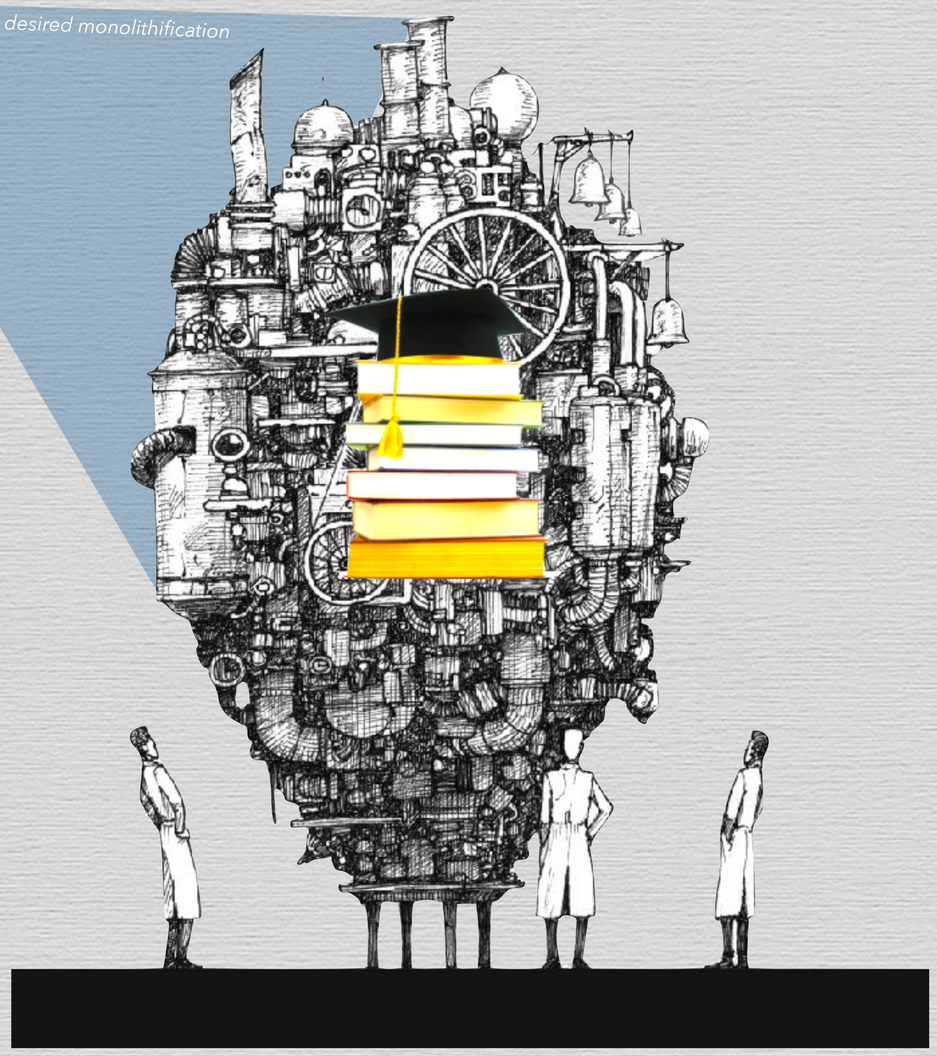


figure from Burns, Oliveira, Pritchard, Slipchenko

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

processors of whole calcs
finite difference 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 & multi-ref. PT2 • arb.-order CC & CI
active-space CC & CR-CC
ADC methods • GW
linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & fragment (EFP) potentials
relativistic contrib. • solvation corrections
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
fast Fourier transform

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

maximal observed monolithification

minimal desired monolithification

QC PROGRAMS

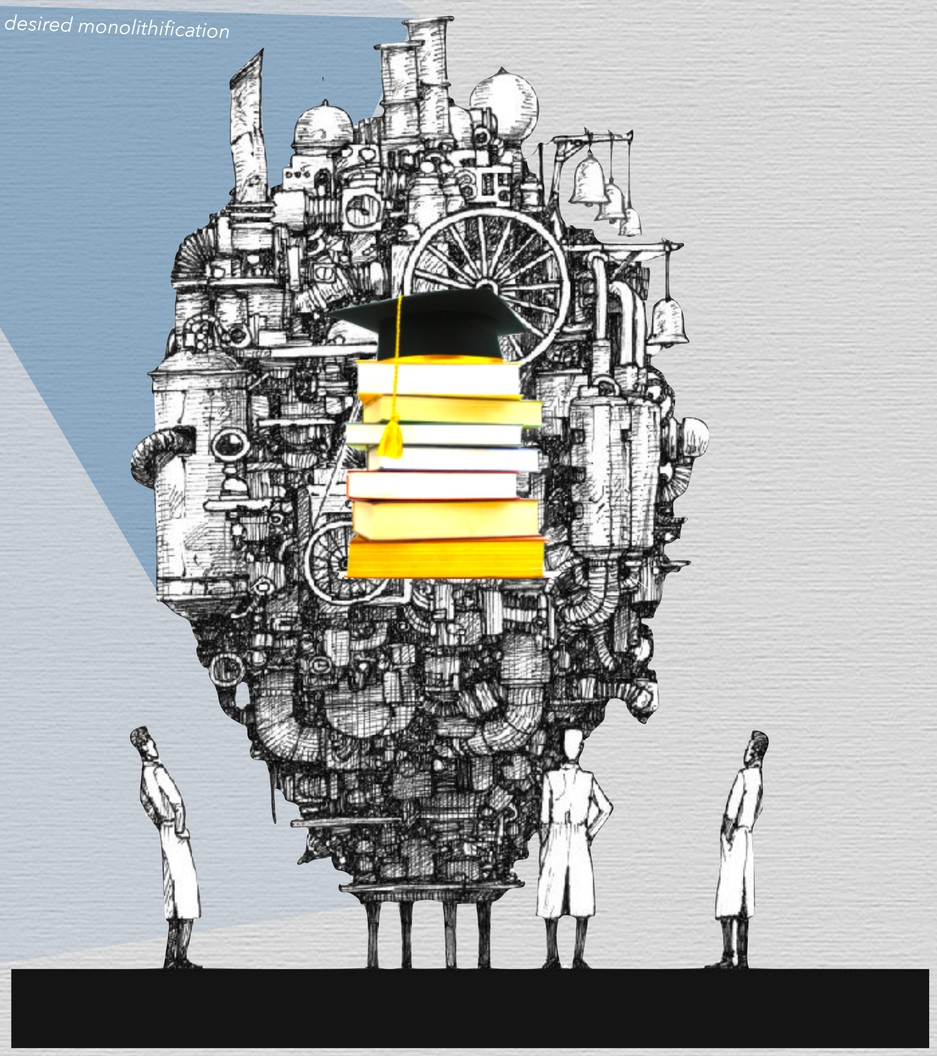


figure from Burns, Oliveira, Pritchard, Slipchenko

internal geo. optimizer → **OPTKING**
external **CRYSTALATTE**

internal **MBD** driver
internal **FINDIF** driver

branched **GDMA**

still **OFFENDING**
now **EXCULPATED**
always **MODULAR**

DFTD3 interface embedded

CHEMPS2 interface embedded
internal ADC methods → **ADCC**
external **MRCC**

LIBEFP vendored
DKH embedded
external **PCMSOLVER**, **CPPE**

internal **BASIS SET LIBRARY**
PHYS CONST copied → **QCELEMENTAL**

LIBINT vendored & branched
internal grid code → **GAU2GRID**
internal DFT functionals → **LIBXC**

internal **PSIO** binary I/O
external **LAPACK**

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

processors of whole calcs
finite difference 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 & multi-ref. PT2 • arb.-order CC & CI
active-space CC & CR-CC
ADC methods • GW
linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & fragment (EFP) potentials
relativistic contrib. • solvation corrections
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
fast Fourier transform

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

maximal observed monolithification

minimal desired monolithification

QC PROGRAMS

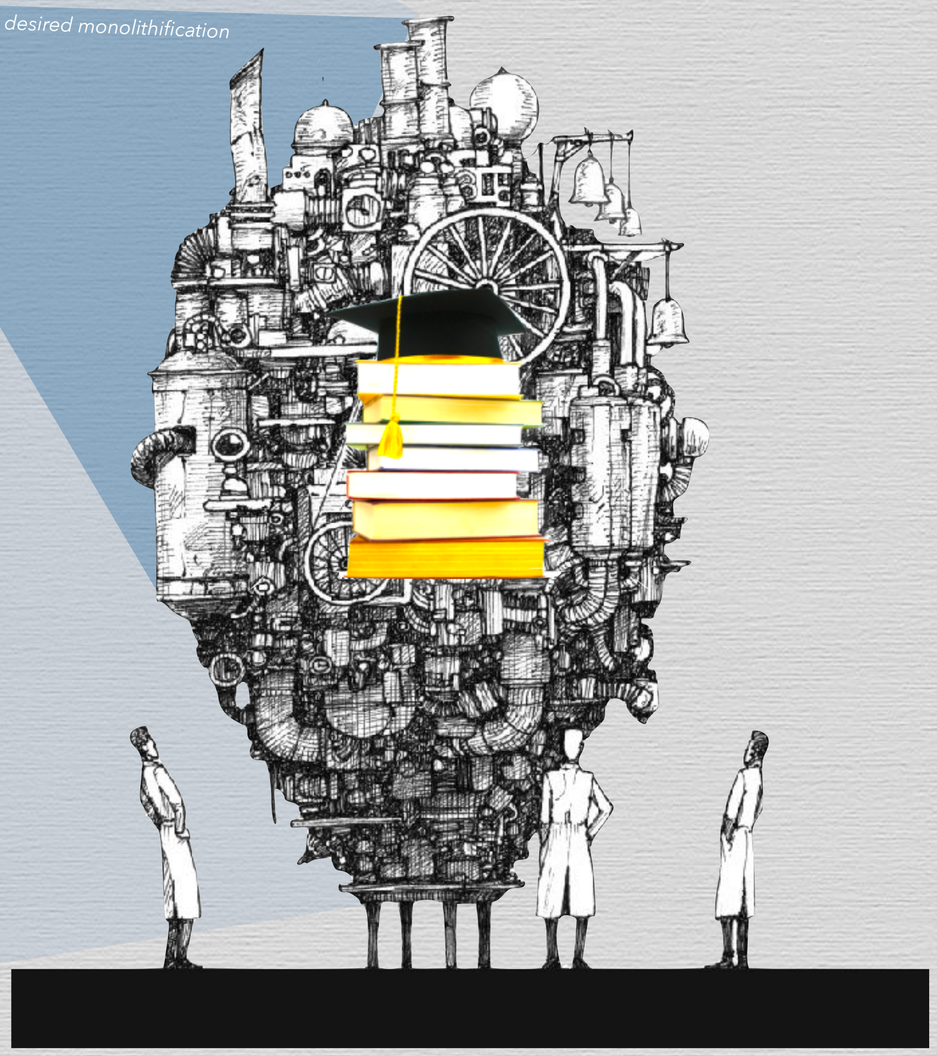
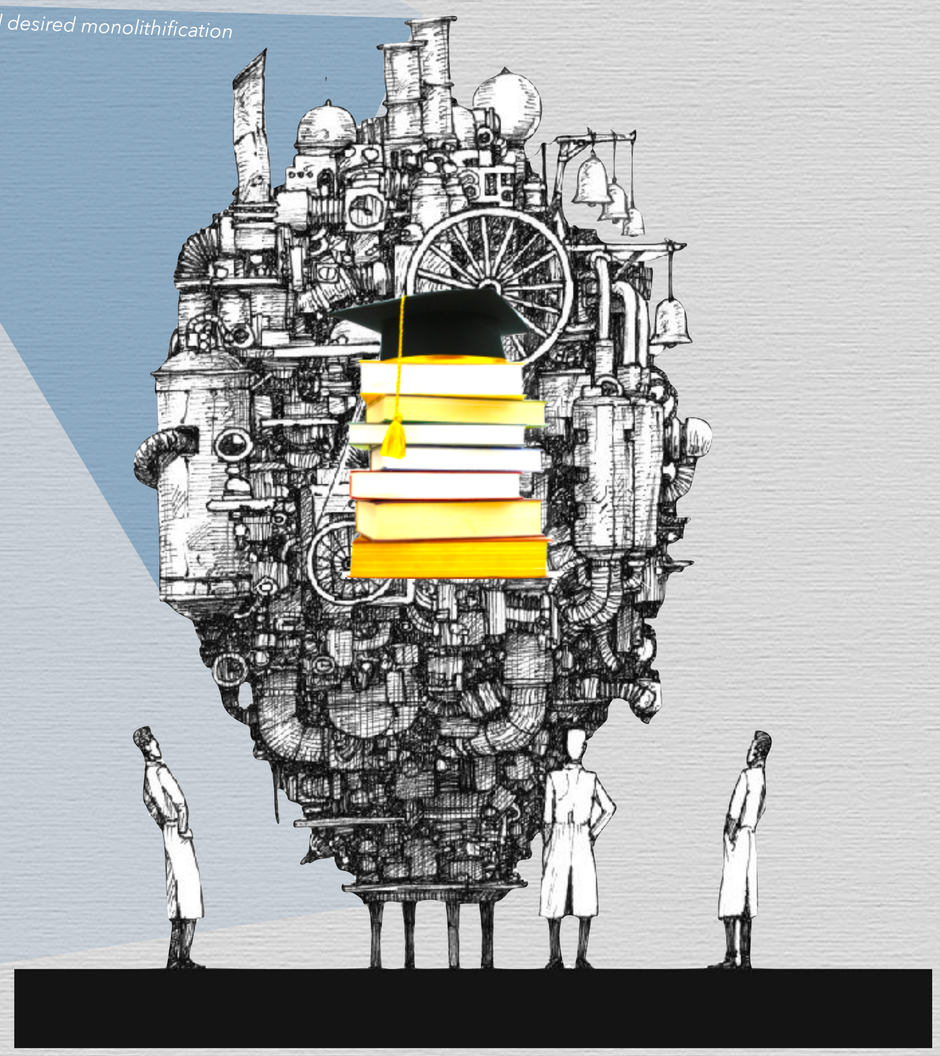


figure from Burns, Oliveira, Pritchard, Slipchenko

QC PROGRAMS



maximal observed monolithification

minimal desired monolithification

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

processors of whole calcs
finite difference 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 & multi-ref. PT2 • arb.-order CC & CI
active-space CC & CR-CC
ADC methods • GW
linear-response TDDFT

SCF-time (part of wfn), API comm.
eff. core (ECP) & fragment (EFP) potentials
relativistic contrib. • solvation corrections
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
fast Fourier transform

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

internal geo. optimizer → **OPTKING**
external **CRYSTALATTE**

internal **MBD** driver ★
internal **FINDIF** driver

branched **GDMA**

DFTD3 interface embedded ★

CHEMPS2 interface embedded
internal ADC methods → **ADCC**
external **MRCC**

LIBEFP vendored
DKH embedded
external **PCMSOLVER**, **CPPE**

internal **BASIS SET LIBRARY**
PHYS CONST copied → **QCELEMENTAL**

LIBINT vendored & branched ★
internal grid code → **GAU2GRID**
internal DFT functionals → **LIBXC**

internal **PSIO** binary I/O
external **LAPACK**

Psi4

still **OFFENDING**
now **EXCULPATED**
always **MODULAR**

figure from Burns, Oliveira, Pritchard, Slipchenko

(0) Psi4 SINGLE-METHOD

simple and customary

```
psi4.energy("ccsd(t)/cc-pvtz", molecule=  )
```

Summary capabilities of Psi4. "✓" runs analytically. "::" runs derivative with internal finite difference. Double underline "✓" selector (e.g., CC_TYPE) unspecified.

	REFERENCE →	Restricted (RHF)										Unrestricted (UHF)									
name ↓ →		energy()						gradient()[8]				energy()						gradient()[8]			
	type[6] ↓ →	CV		DF		CD		CV		DF		CD		CV		DF		CD		CV		DF		CD	
	FREEZE_CORE[7] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
hf	SCF_TYPE	✓		✓		✓		✓		✓		::		✓		✓		✓		✓		✓		::	
mp2	MP2_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::
mp2.5	MP_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::
mp3	MP_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::
mp4(sdq)	MP_TYPE	✓	✓																						
mp4, mpn[5]	MP_TYPE	✓	✓																						
zapt2, zaptn[9]	MP_TYPE																								
cisd, cin[5]	CI_TYPE	✓	✓																						
qcisd	CI_TYPE	✓	✓																						
qcisd(t)	CI_TYPE	✓	✓																						
fci	CI_TYPE	✓	✓																						
remp2	CC_TYPE	✓	✓	✓	✓	✓	✓							✓	✓	✓	✓	✓	✓						
lccd	CC_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::
lccsd, cepa(0)	CC_TYPE	✓	✓																						
cepa(1)	CC_TYPE	✓	✓																						
cepa(3)	CC_TYPE	✓	✓																						
acpf	CC_TYPE	✓	✓																						
aqcc	CC_TYPE	✓	✓																						
ccd	CC_TYPE			✓	✓	✓	✓			✓	✓	::	::			✓	✓	✓	✓			✓	✓	::	::
hccd	CC_TYPE	✓	✓											✓	✓										
cc2	CC_TYPE	✓	✓					✓	::					✓	✓					::	::				
ccsd	CC_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::
ccsd(t)	CC_TYPE	✓	✓	✓	✓	✓	✓	✓	::	✓	✓	::	::	✓	✓					✓	::				

(0) Psi4 SINGLE-METHOD

simple and customary

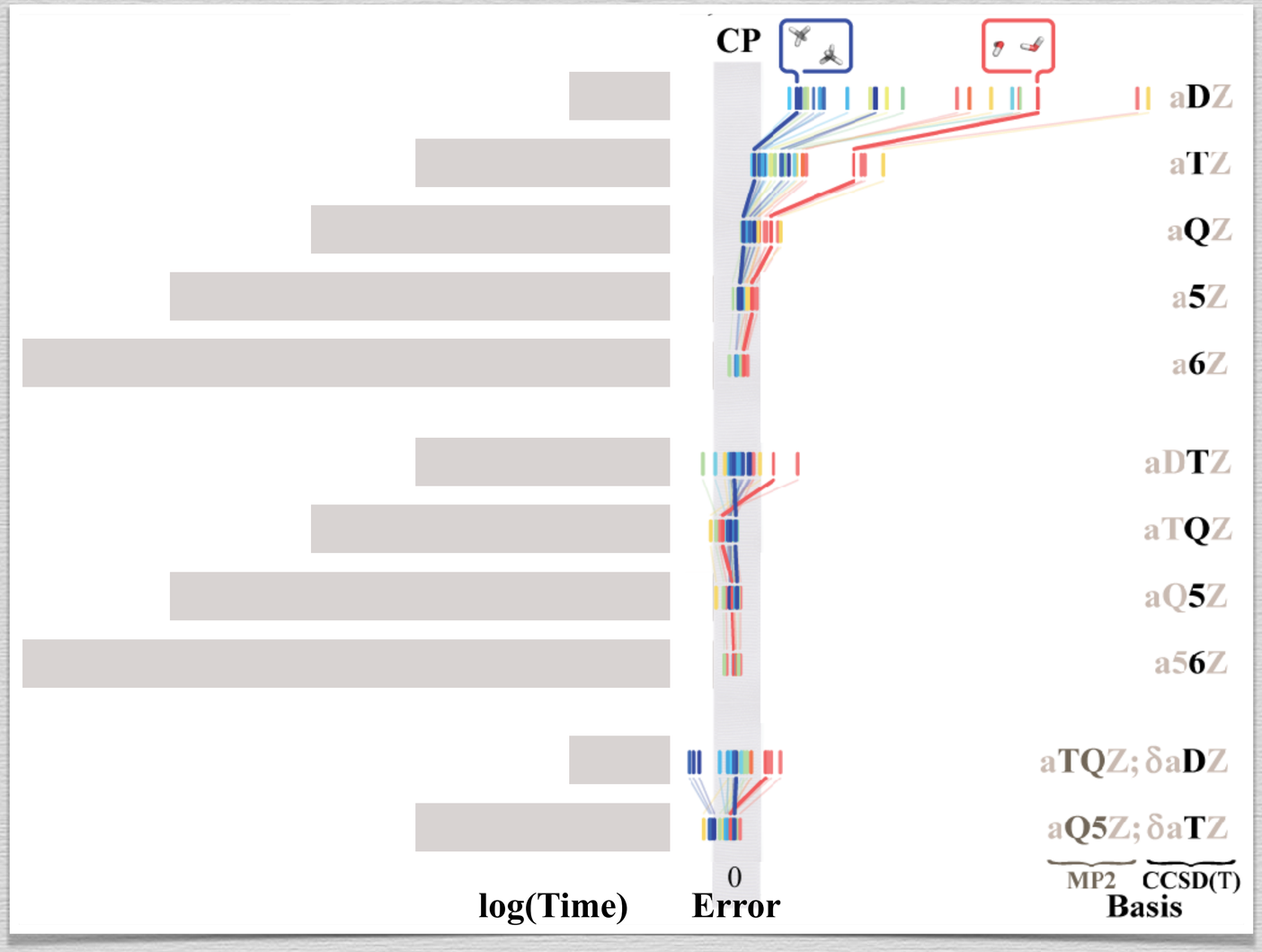
```
psi4.energy("ccsd(t)/cc-pvtz", molecule=  )
```

Local Compute

- **AVAILABLE** with Psi4 since 2010.
- **SPECIFICATION** through single file.
- **RATE-LIMITED** by self since single calc.
- **RETRIEVAL** from filesystem if named systematically.
- **FLEXIBILITY** to choose any QC program for single more expensive method.

COMPOSITE & BSSE APPROXIMATIONS

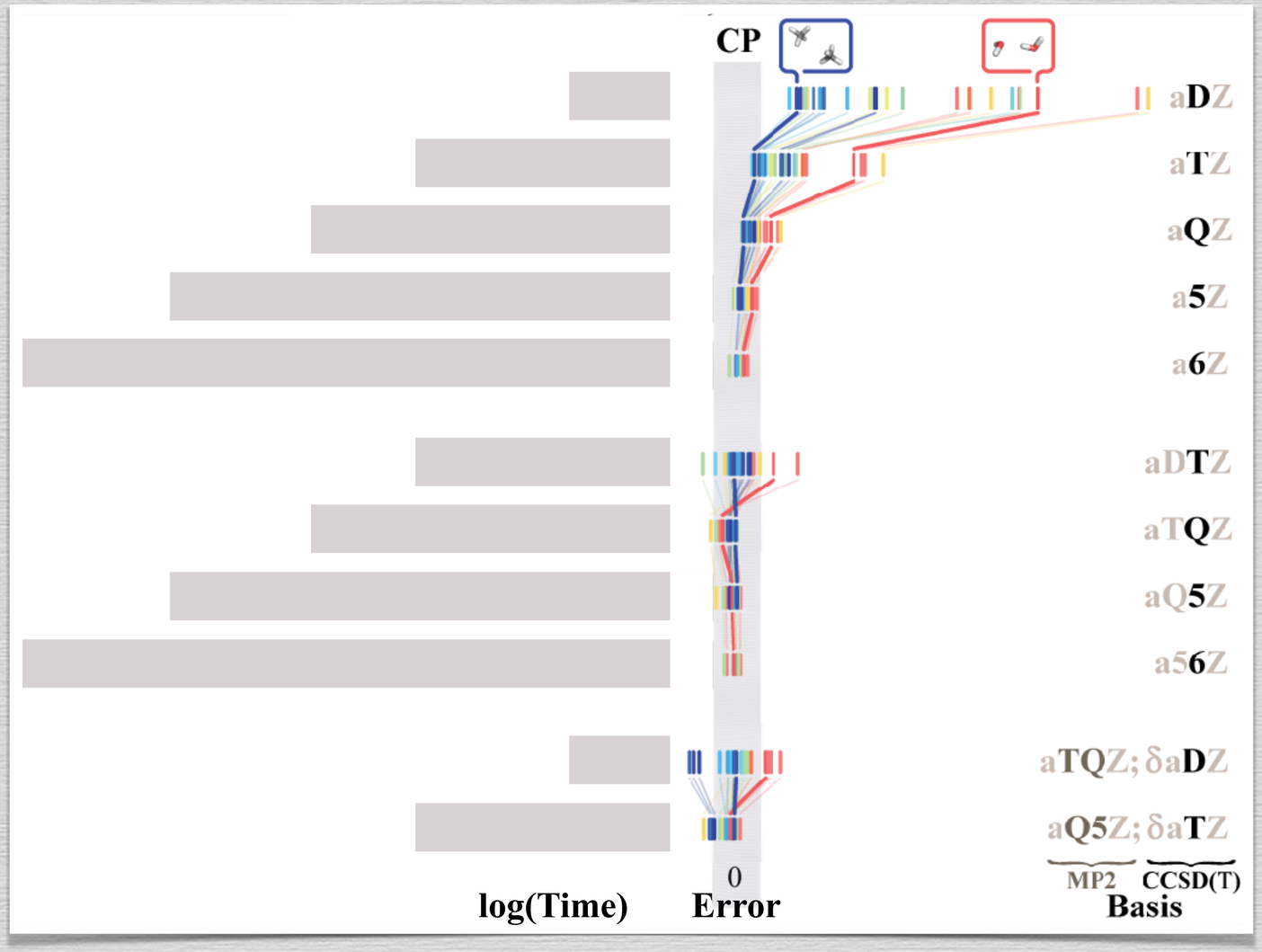
complete basis set (CBS) approximations, delta corrections, focal point methods




```
psi4.energy("ccsd(t)/cc-pvtz")
```

COMPOSITE & BSSE APPROXIMATIONS

complete basis set (CBS) approximations, delta corrections, focal point methods



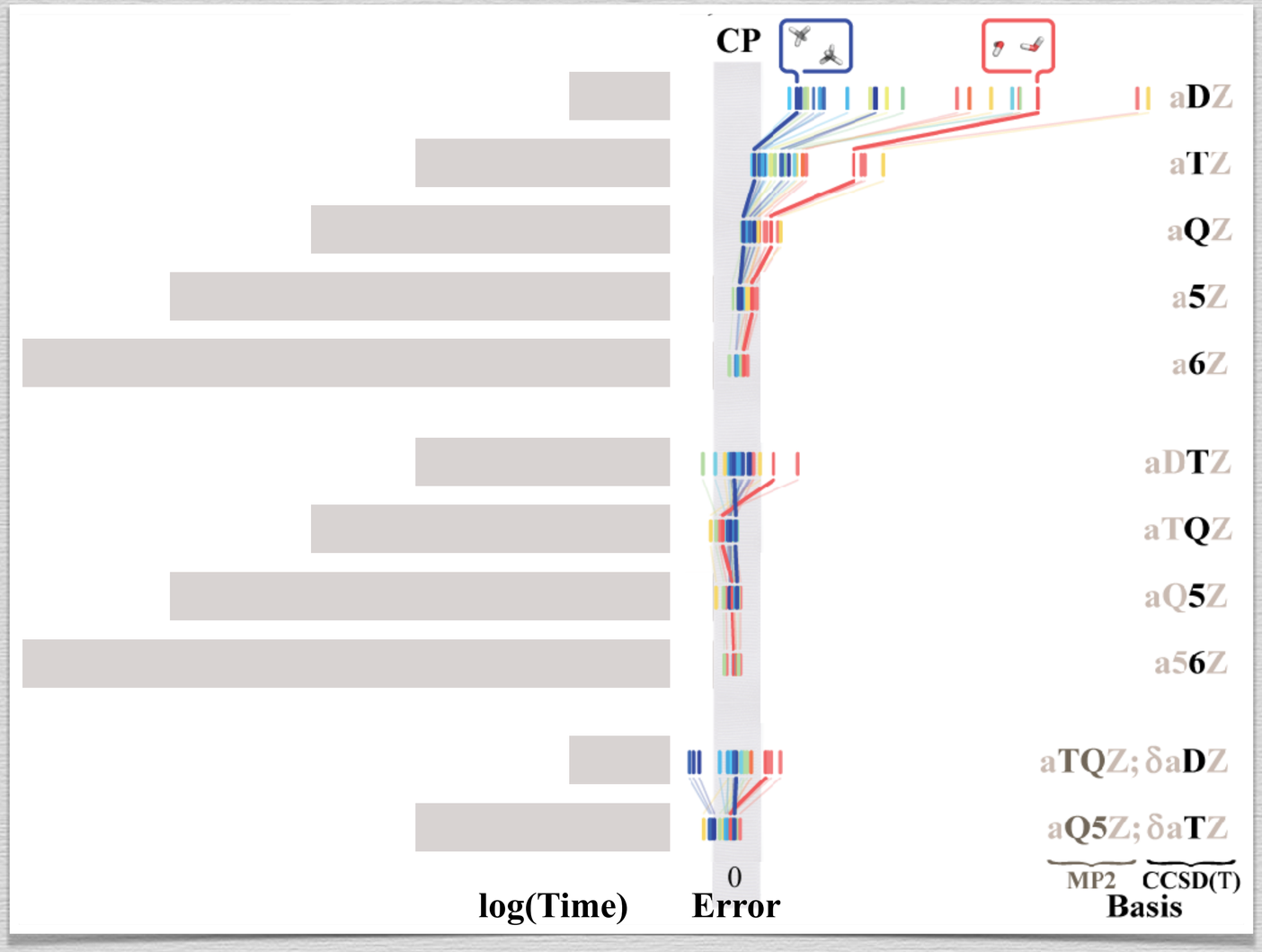

```
psi4.energy("ccsd(t)/cc-pvtz")
```

```
psi4.energy("ccsd(t)/cc-pv[dt]z")
```

$$\text{AB}\zeta: \quad E_{\text{total}}^{\text{post-HF/TQ}\zeta} = E_{\text{total}}^{\text{HF/Q}\zeta} + \frac{\ell_{\max}^{\text{pow}}}{\ell_{\max}^{\text{pow}} - (\ell_{\max}-1)^{\text{pow}}} E_{\text{corr}}^{\text{post-HF/Q}\zeta} - \frac{(\ell_{\max}-1)^{\text{pow}}}{\ell_{\max}^{\text{pow}} - (\ell_{\max}-1)^{\text{pow}}} E_{\text{corr}}^{\text{post-HF/T}\zeta}. \qquad (7)$$

COMPOSITE & BSSE APPROXIMATIONS

complete basis set (CBS) approximations, delta corrections, focal point methods




```
psi4.energy("ccsd(t)/cc-pvtz")
```

```
psi4.energy("ccsd(t)/cc-pv[dt]z")
```

$$\mathrm{AB}\zeta: \quad E_{\mathrm{total}}^{\text{post-HF/TQ}\zeta} = E_{\mathrm{total}}^{\mathrm{HF/Q}\zeta} + \frac{\ell_{\max}^{\mathrm{pow}}}{\ell_{\max}^{\mathrm{pow}} - (\ell_{\max}-1)^{\mathrm{pow}}} E_{\mathrm{corr}}^{\text{post-HF/Q}\zeta} - \frac{(\ell_{\max}-1)^{\mathrm{pow}}}{\ell_{\max}^{\mathrm{pow}} - (\ell_{\max}-1)^{\mathrm{pow}}} E_{\mathrm{corr}}^{\text{post-HF/T}\zeta}. \qquad (7)$$

```
psi4.energy("mp2/cc-pv[dt]z +
             D:ccsd(t)/cc-pvdz")
```

$$[\mathrm{A}\zeta;\delta{:}\mathrm{B}\zeta]: \quad E_{\mathrm{total}}^{\text{post-MP2/[Q5}\zeta;\delta{:}\mathrm{T}\zeta]}$$

$$= E_{\mathrm{total}}^{\mathrm{MP2/Q5}\zeta} + \delta_{\mathrm{MP2/T}\zeta}^{\text{post-MP2/T}\zeta}, \qquad (8)$$

$$= E_{\mathrm{total}}^{\mathrm{HF/5}\zeta} + E_{\mathrm{corr}}^{\mathrm{MP2/Q5}\zeta} + E_{\mathrm{corr}}^{\text{post-MP2/T}\zeta} - E_{\mathrm{corr}}^{\mathrm{MP2/T}\zeta}, \qquad (9)$$

(1) Psi4 COMPOSITE & BSSE BY HAND

flexible but error-prone

```
psi4.energy("mp2/cc-pvtz", molecule= )
psi4.energy("mp2/cc-pvtz", molecule= )
psi4.energy("mp2/cc-pvtz", molecule= )
```

```
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
```

Local Compute

- **AVAILABLE** with Psi4 since 2010.
- **SPECIFICATION** through multiple files of slight variation. Procedure manual, so high risk of user error.
- **RATE-LIMITED** by single longest calc since can run parallel in queue.
- **RETRIEVAL** from filesystem if named systematically.
- **FLEXIBILITY** to mix QC programs to access more or more efficient methods since separate input files.

(1) Psi4 COMPOSITE & BSSE BY HAND

flexible but error-prone

```
psi4.energy("mp2/cc-pvtz", molecule= )
psi4.energy("mp2/cc-pvtz", molecule= )
psi4.energy("mp2/cc-pvtz", molecule= )
```

```
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
psi4.energy("ccsd(t)/cc-pvdz", molecule= )
```

Local Compute

- **AVAILABLE** with Psi4 since 2010.
- **SPECIFICATION** through multiple files of slight variation. Procedure manual, so high risk of user error.
- **RATE-LIMITED** by single longest calc since can run parallel in queue.
- **RETRIEVAL** from filesystem if named systematically.
- **FLEXIBILITY** to mix QC programs to access more or more efficient methods since separate input files.

PSI4 RECURSIVE DRIVER

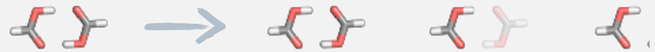
`def energy(method):` `def gradient(method):` `def hessian(method):`

PSI4 RECURSIVE DRIVER

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

`def manybody ():`

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



RUN `for frag in fragments:`

`energy`
`gradient(method,frag)`
`hessian`

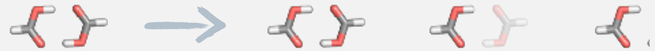
Assemble n-body & interaction results from fragments.

PSI4 RECURSIVE DRIVER

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

`def manybody ():`

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



RUN `for frag in fragments:`

```
energy
gradient(method,frag)
hessian
```

Assemble n-body & interaction results from fragments.

`def findif ():`

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

RUN `for disp in displacements:`

```
energy
gradient(method,disp)
hessian
```

Assemble derivative results from displacements.

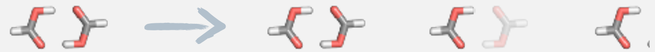
E · *G* ⋮⋮ *H*

PSI4 RECURSIVE DRIVER

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

`def manybody ():`

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



RUN `for frag in fragments:`

```
energy
gradient(method,frag)
hessian
```

Assemble n-body & interaction results from fragments.

`def findif ():`

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

RUN `for disp in displacements:`

```
energy
gradient(method,disp)
hessian
```

Assemble derivative results from displacements.

`def composite ():`

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

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

RUN `for mc in modelchems:`

```
energy
gradient(mc,molecule)
hessian
```

Assemble extrapolations & total results from modelchems.

E · *G* ⋮⋮ *H*

PSI4 RECURSIVE DRIVER

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

start
QC calculation
end

decision
yes
no

BSSE in **options**?

composite **method**?

analytic Hessian for **method**?

analytic gradient for **method**?

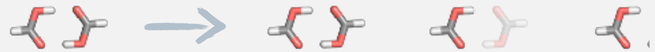
analytic energy for **method**?

`procedures[func][method](molecule)`

E · *G* ⋮⋮ *H*

`def manybody ():`

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



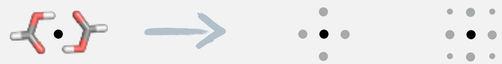
RUN `for frag in fragments:`

`energy`
`gradient(method,frag)`
`hessian`

Assemble n-body & interaction results from fragments.

`def findif ():`

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



RUN `for disp in displacements:`

`energy`
`gradient(method,disp)`
`hessian`

Assemble derivative results from displacements.

`def composite ():`

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

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

RUN `for mc in modelchems:`

`energy`
`gradient(mc,molecule)`
`hessian`

Assemble extrapolations & total results from modelchems.

PSI4 RECURSIVE DRIVER

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

start
QC calculation
end

decision
yes
no

BSSE in **options**?

composite **method**?

analytic Hessian for **method**?

analytic gradient for **method**?

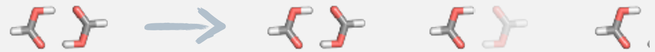
analytic energy for **method**?

`procedures[func][method](molecule)`

E · *G* ⋮⋮ *H*

`def manybody ():`

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



RUN `for frag in fragments:`

`energy`
`gradient(method,frag)`
`hessian`

Assemble n-body & interaction results from fragments.

`def findif ():`

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

RUN `for disp in displacements:`

`energy`
`gradient(method,disp)`
`hessian`

Assemble derivative results from displacements.

`def composite ():`

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

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

RUN `for mc in modelchems:`

`energy`
`gradient(mc,molecule)`
`hessian`

Assemble extrapolations & total results from modelchems.

PSI4 RECURSIVE DRIVER

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

start
QC calculation
end

decision
yes
no

BSSE in **options**?

composite **method**?

analytic Hessian for **method**?

analytic gradient for **method**?

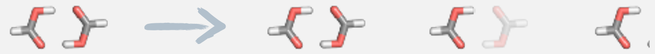
analytic energy for **method**?

```
procedures[func][method](molecule)
```

E · G ⋮⋮ H

def manybody ():

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



RUN `for frag in fragments:`

```
energy
gradient(method,frag)
hessian
```

Assemble n-body & interaction results from fragments.

def findif ():

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

RUN `for disp in displacements:`

```
energy
gradient(method,disp)
hessian
```

Assemble derivative results from displacements.

def composite ():

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

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

RUN `for mc in modelchems:`

```
energy
gradient(mc,molecule)
hessian
```

Assemble extrapolations & total results from modelchems.

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

Psi4 RECURSIVE DRIVER

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

start
QC calculation
end

decision
yes
no

BSSE in **options**?

composite **method**?

analytic Hessian for **method**?

analytic gradient for **method**?

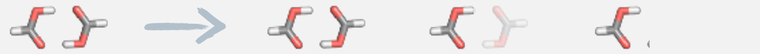
analytic energy for **method**?

```
procedures[func][method](molecule)
```

E · G ⋮⋮ H

```
def manybody ():
```

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



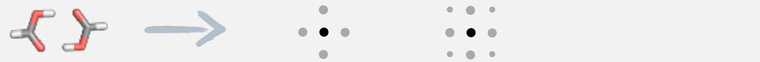
RUN `for frag in fragments:`

```
energy
gradient(method,frag)
hessian
```

Assemble n-body & interaction results from fragments.

```
def findif ():
```

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



RUN `for disp in displacements:`

```
energy
gradient(method,disp)
hessian
```

Assemble derivative results from displacements.

```
def composite ():
```

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

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

RUN `for mc in model chems:`

```
energy
gradient(mc,molecule)
hessian
```

Assemble extrapolations & total results from model chems.

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

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

(2) Psi4 RECURSIVE DRIVER, INTERNAL

capable but complicated

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

Local Compute

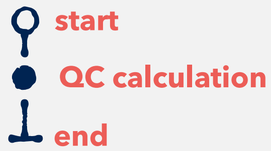
- **AVAILABLE** with Psi4 in current form since 2016.
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by sum of all calcs since run sequentially.
- **RETRIEVAL** from filesystem difficult since many calcs in aggregated outfile.
- **FLEXIBILITY** limited to Psi4 only.

PSI4 DISTRIBUTED DRIVER

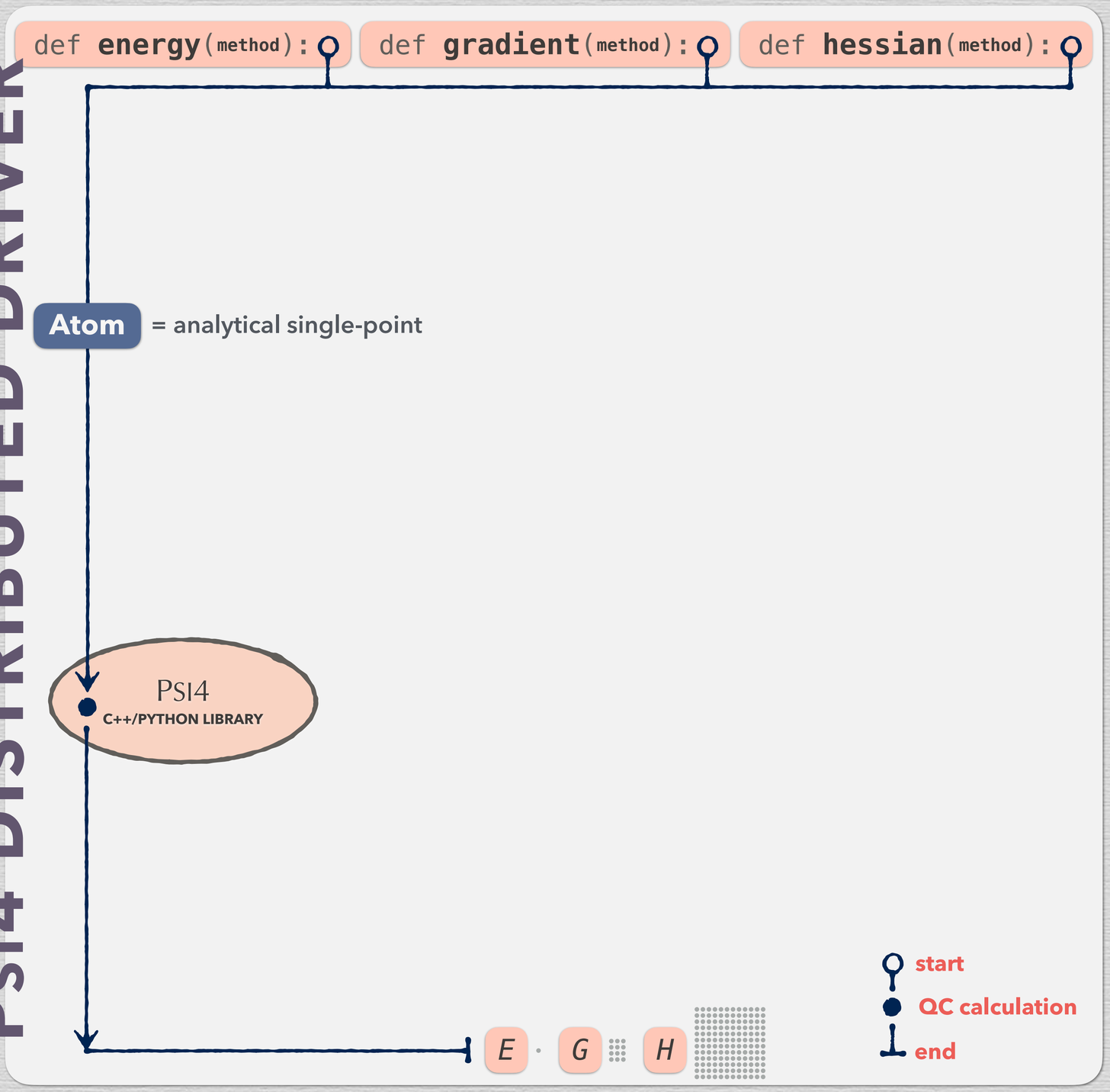
```
def energy(method):
```

```
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

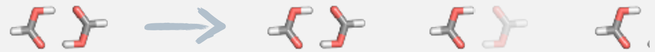
PSI4
C++/PYTHON LIBRARY

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

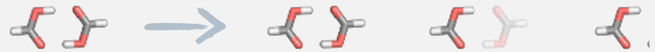
PSI4
C++/PYTHON LIBRARY

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

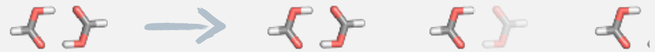
PSI4
C++/PYTHON LIBRARY

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

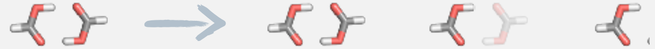
PSI4
C++/PYTHON LIBRARY

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

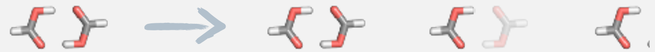
PSI4
C++/PYTHON LIBRARY

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

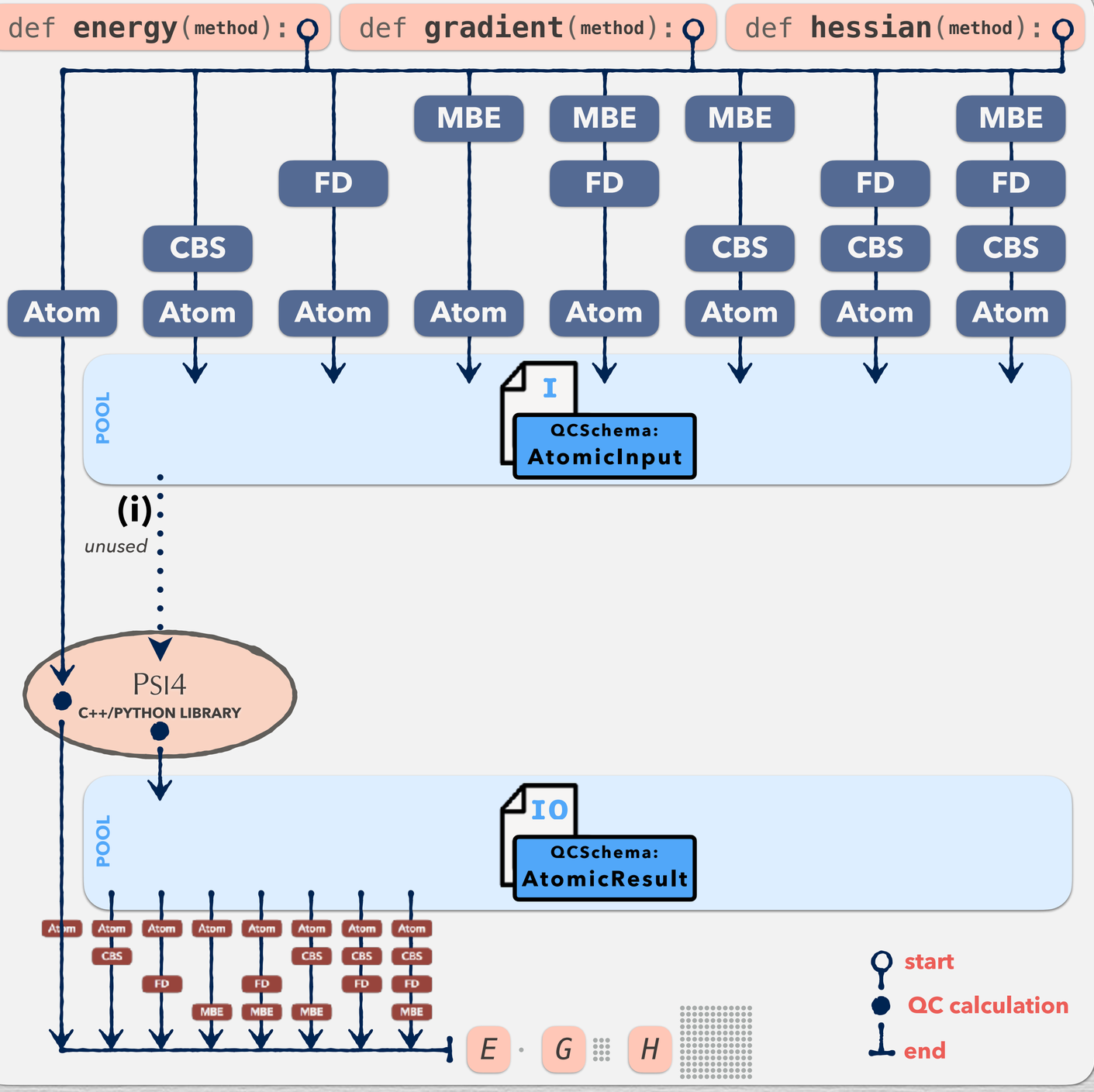
ASM Assemble extrapolations & total results from modelchems.

`class AtomicComputer ():`

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

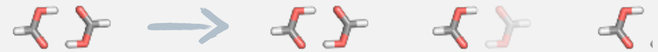
ASM Return analytic energy, gradient, or Hessian.

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

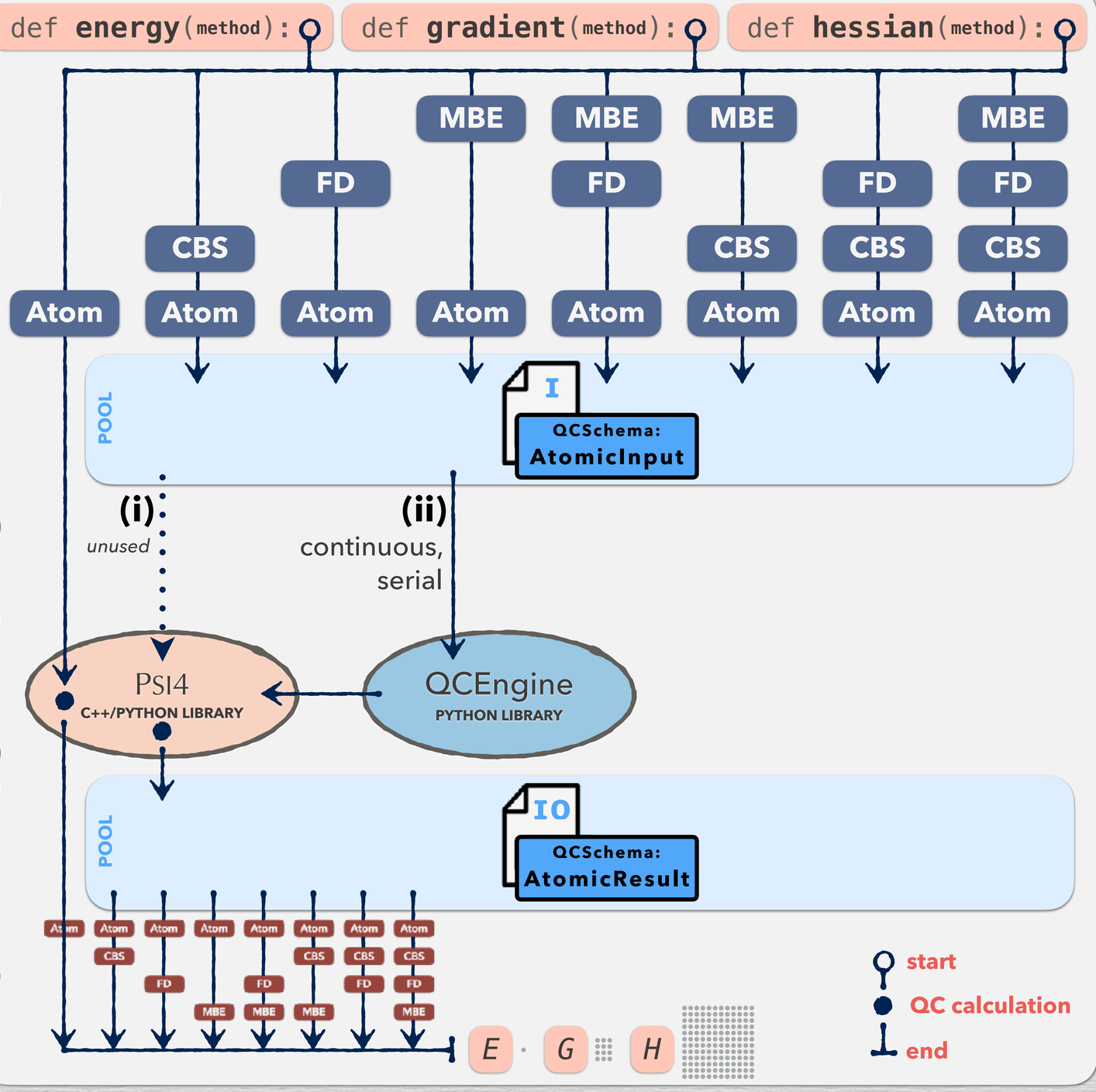
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

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

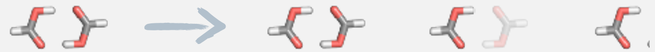
ASM Return analytic energy, gradient, or Hessian.

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

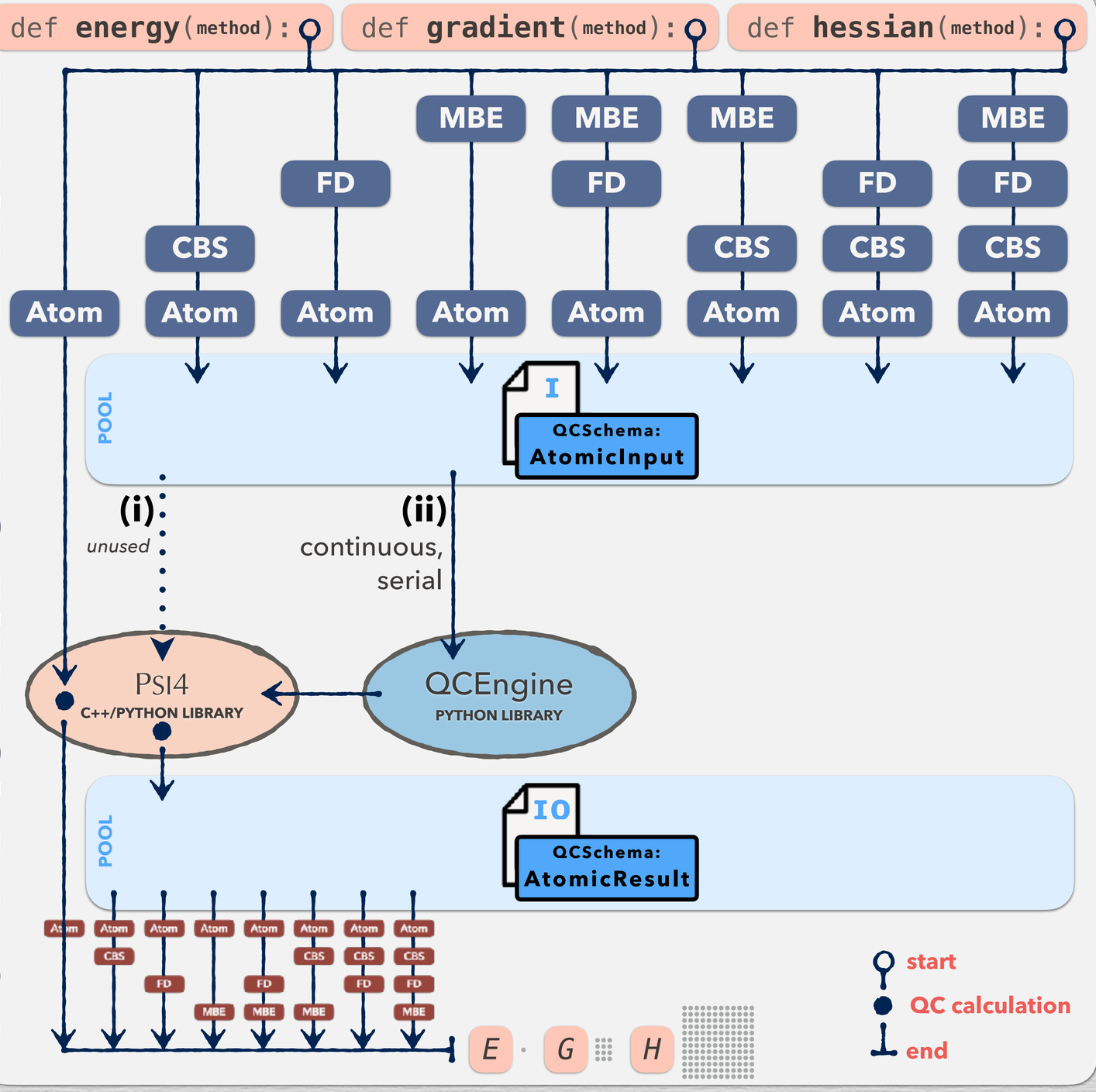
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

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

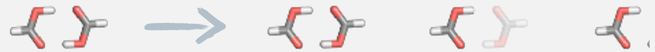
ASM Return analytic energy, gradient, or Hessian.

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

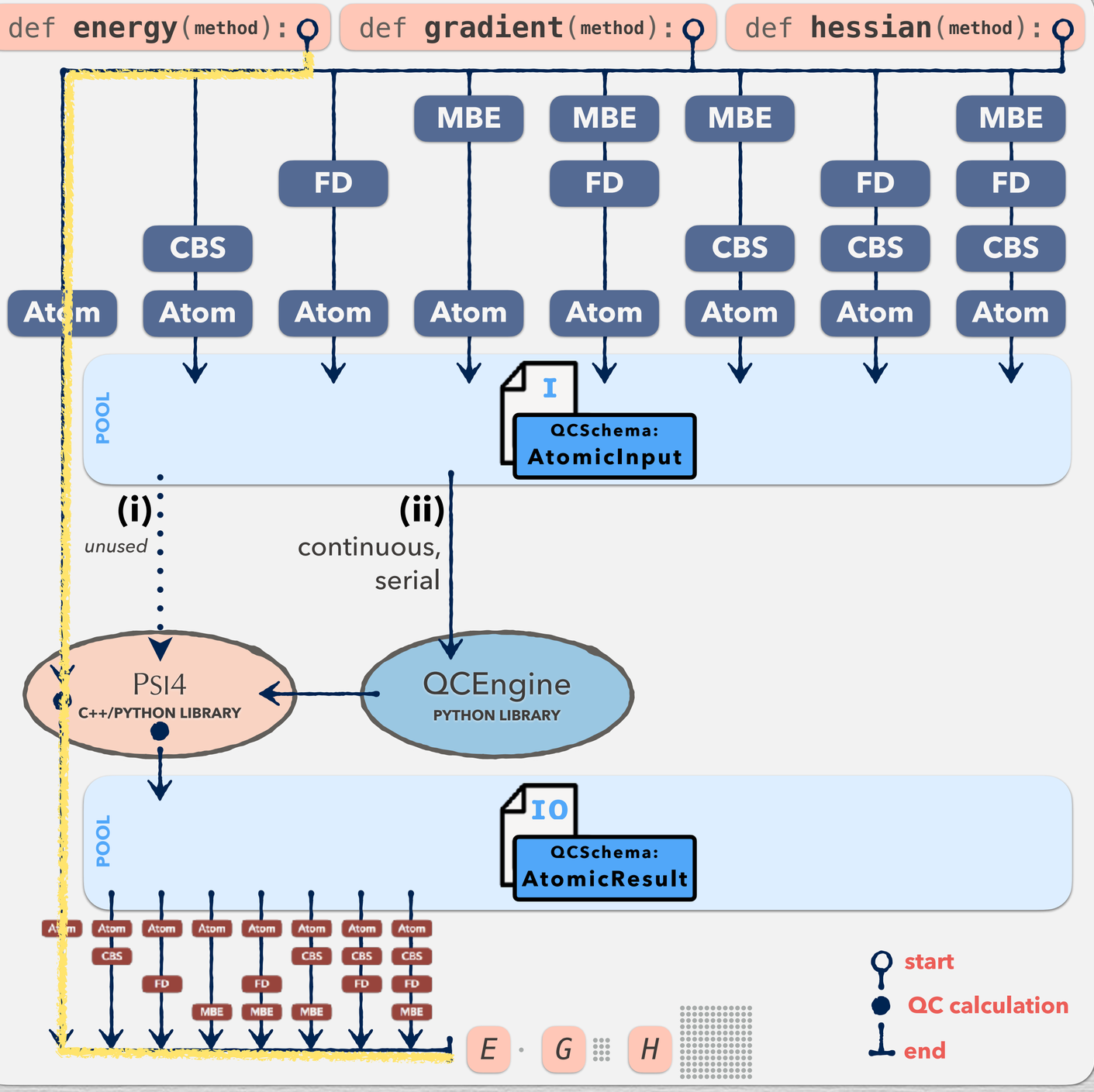
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

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

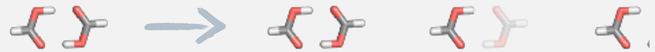
ASM Return analytic energy, gradient, or Hessian.

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.

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.

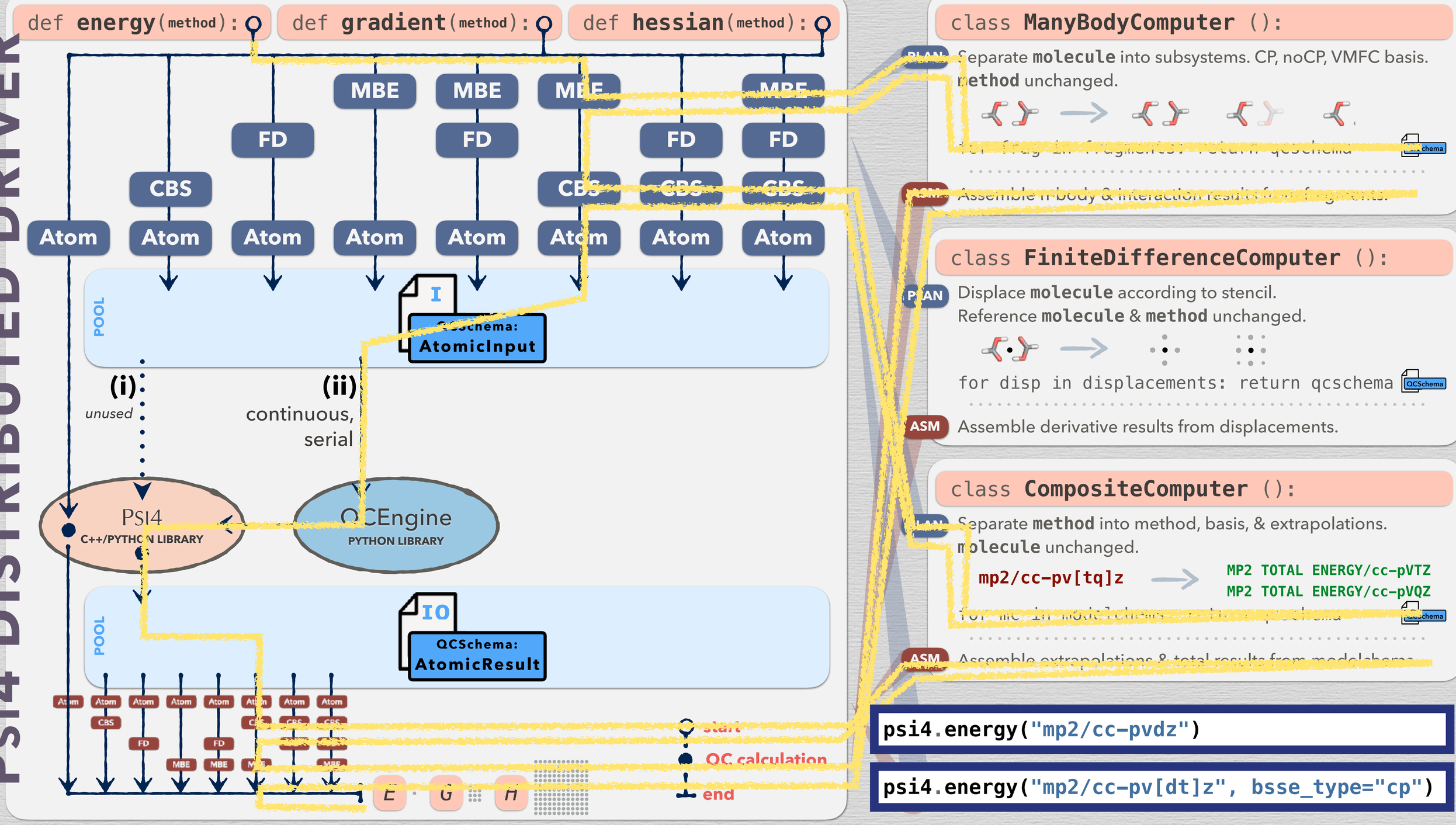
```
mp2/cc-pv[tq]z  →  MP2 TOTAL ENERGY/cc-pVTZ
                   MP2 TOTAL ENERGY/cc-pVQZ
for mc in modelchems: return qcschema
```

QCSchema

ASM Assemble extrapolations & total results from modelchems.

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

PSI4 DISTRIBUTED DRIVER




```
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.

```
class FiniteDifferenceComputer ():
```

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


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

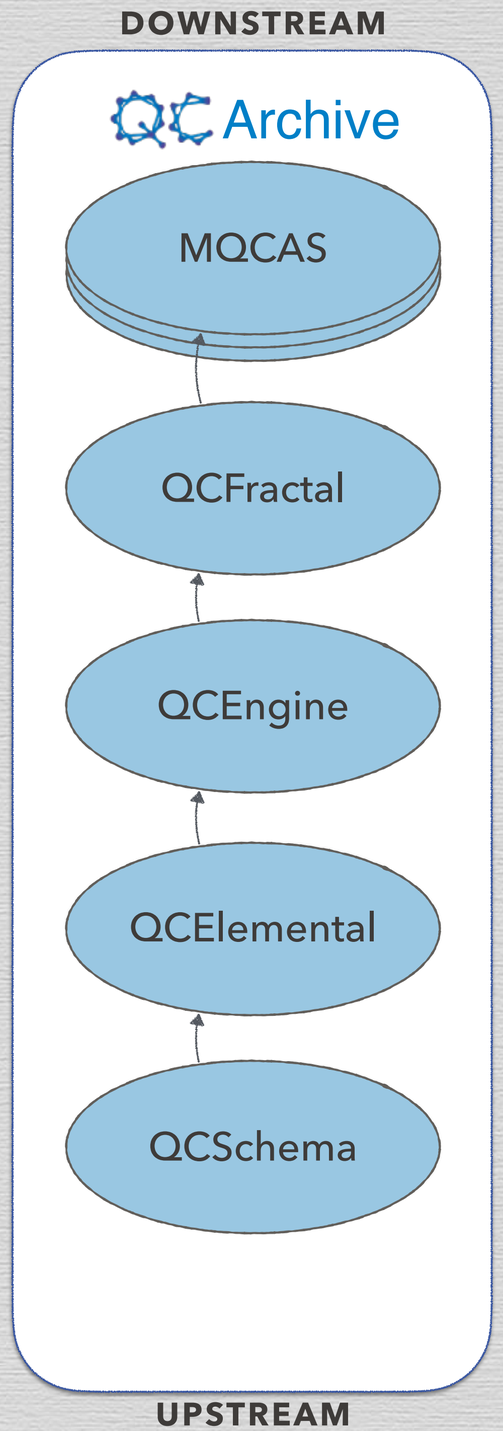
ASM Assemble extrapolations & total results from modelchems.

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

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

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

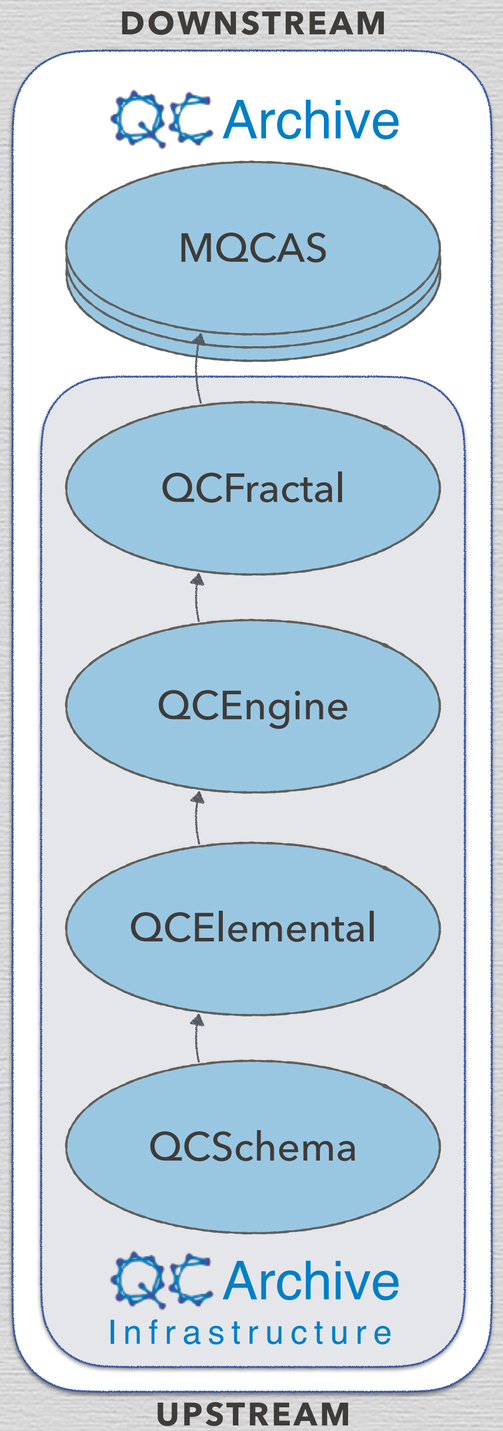
- Python QCSchema runner
- hardware compute configuration (e.g., memory, nodes)
- DSL input syntax for QC programs

- 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/



- 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 elemental

conda install qcelemental -c conda-forge

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

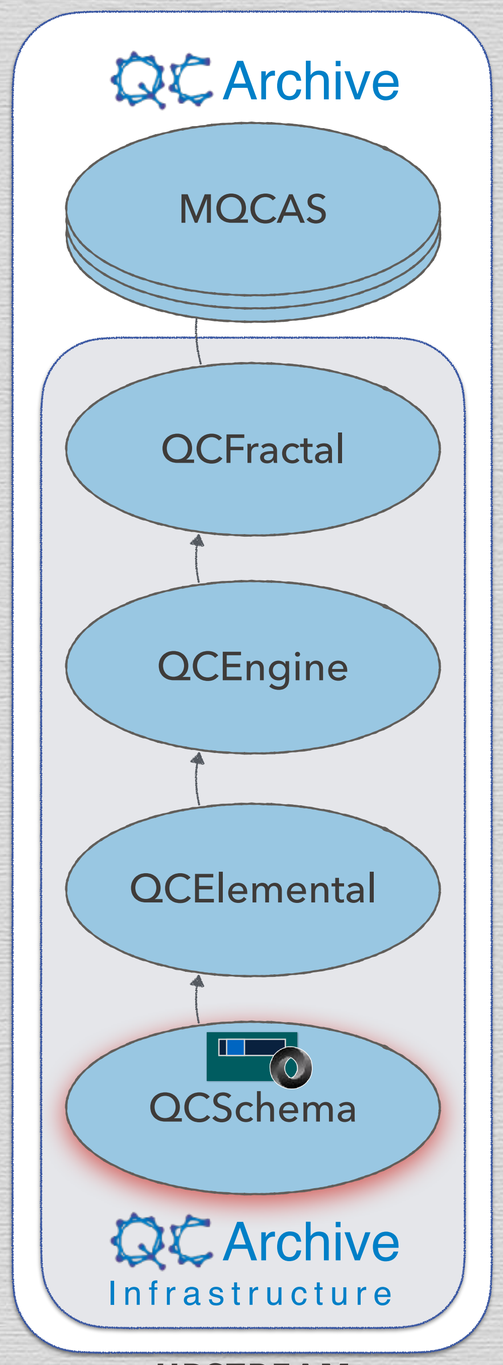
github.com/MolSSI/QCSchema

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

QCArchive STACK

qcarchive.molssi.org/

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 elemental

conda install qcelemental -c conda-forge

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

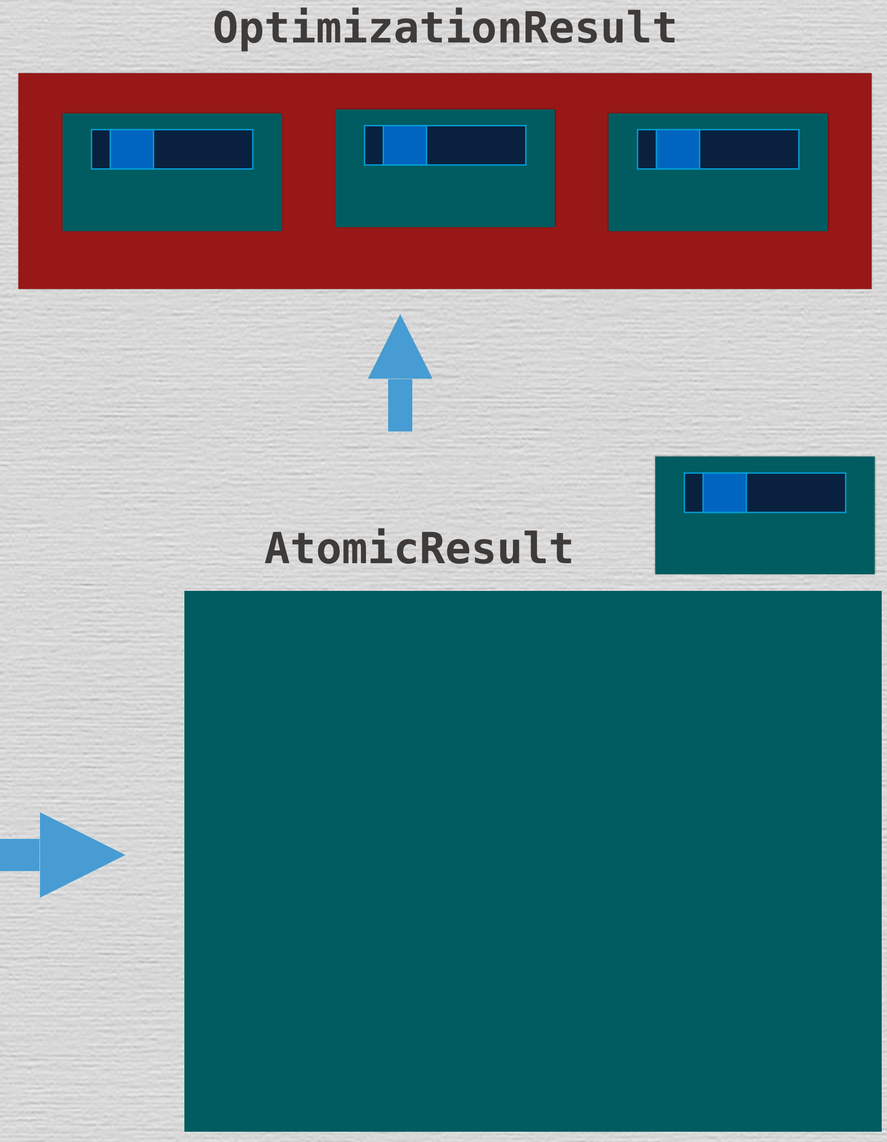
github.com/MolSSI/QCSchema

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

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** development stage since 2017. Most changes for QCA. More community participation desired but little momentum.
- **SCHEMA** defined in Python with Pydantic in QCElemental module, then exported to JSON.



QUANTUM CHEMISTRY SCHEMA

primary schemas defined thus far

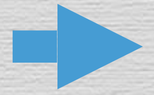
- **MOLECULE** Cartesian, atomic units (AU) based specification.
- **ATOMICINPUT** job directive for analytic single-point – energy, gradient, H
  or property – as defined by **DRIVER**.
- **ATOMICRESULT** append to **ATOMICINPUT** with output, simple scalar and ar
  results, and wavefunction results.
- **PROVENANCE** who wrote the data – program, version, module, compute
  information.
- **BASISSET** similar to BSE. allows **WAVEFUNCTION** data in CCA ordering.
- **OPTIMIZATIONINPUT & OPTIMIZATIONRESULT** job directive for abstract gen
  optimizer call generic QC program for gradient computation.

Molecule

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

```
0 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

primary schemas defined thus far

- **MOLECULE** Cartesian, atomic units (AU) based specification.
- **ATOMICINPUT** job directive for analytic single-point – energy, gradient, Hessian, or property – as defined by **DRIVER**.
- **ATOMICRESULT** append to **ATOMICINPUT** with output, simple scalar and array results, and wavefunction results.
- **PROVENANCE** who wrote the data – program, version, module, compute information.
- **BASISSET** similar to BSE. allows **WAVEFUNCTION** data in CCA ordering.
- **OPTIMIZATIONINPUT & OPTIMIZATIONRESULT** job directive for abstract generic optimizer call generic QC program for gradient computation.

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

primary schemas defined thus far

- **MOLECULE** Cartesian, atomic units (AU) based specification.
- **ATOMICINPUT** job directive for analytic single-point – energy, gradient, Hessian, or property – as defined by **DRIVER**.
- **ATOMICRESULT** append to **ATOMICINPUT** with output, simple scalar and array results, and wavefunction results.
- **PROVENANCE** who wrote the data – program, version, module, compute information.
- **BASISSET** similar to BSE. allows **WAVEFUNCTION** data in CCA ordering.
- **OPTIMIZATIONINPUT & OPTIMIZATIONRESULT** job directive for abstract generic optimizer call generic QC program for gradient computation.

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

primary schemas defined thus far

- **MOLECULE** Cartesian, atomic units (AU) based specification.
- **ATOMICINPUT** job directive for analytic single-point – energy, gradient, Hessian, or property – as defined by **DRIVER**.
- **ATOMICRESULT** append to **ATOMICINPUT** with output, simple scalar and array results, and wavefunction results.
- **PROVENANCE** who wrote the data – program, version, module, compute information.
- **BASISSET** similar to BSE. allows **WAVEFUNCTION** data in CCA ordering.
- **OPTIMIZATIONINPUT & OPTIMIZATIONRESULT** job directive for abstract generic optimizer call generic QC program for gradient computation.

- orbitals_a
- orbitals_b
- density_a
- density_b
- fock_a
- fock_b
- eigenvalues_a
- eigenvalues_b
- occupations_a
- occupations_b

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

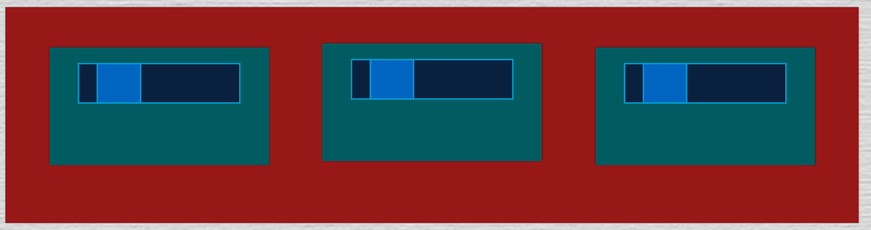
```
{
    ... AtomicInput ...
    "provenance": {
        "creator": "My QM Program",
        "version": "1.1rc1",
    },
    "properties": {
        "calcinfo_nalpha": 5,
        "scf_total_energy": -5.433191881443323,
        "nuclear_repulsion_energy": 2.11670883436,
        "scf_iterations": 8.0,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

QUANTUM CHEMISTRY SCHEMA

primary schemas defined thus far

- **MOLECULE** Cartesian, atomic units (AU) based specification.
- **ATOMICINPUT** job directive for analytic single-point – energy, gradient, Hessian, or property – as defined by **DRIVER**.
- **ATOMICRESULT** append to **ATOMICINPUT** with output, simple scalar and array results, and wavefunction results.
- **PROVENANCE** who wrote the data – program, version, module, compute information.
- **BASISSET** similar to BSE. allows **WAVEFUNCTION** data in CCA ordering.
- **OPTIMIZATIONINPUT & OPTIMIZATIONRESULT** job directive for abstract generic optimizer call generic QC program for gradient computation.

OptimizationResult



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": "..."
}
```

QCSchema DEFINES DATA LAYOUTS

and text advice, nothing more

- **LAYOUT** JSON Schema defines a data layout and some minimal type and count checking.
- **CONVENTIONS** JSON Schema can't check conventions you rely upon like AU units, CCA ordering, center-of-mass positioning

DOMAIN-SPECIFIC ASCII

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

GAMESS Input

DATA LAYOUT TRANSLATION

QCSchema: AtomicInput

```
{
  'molecule': {
    'symbols': [              ],
    'geometry': [                  ],
    'molecular_multiplicity':  },
  'driver': '       ',
  'model': {
    'method': '     ',
    'basis': '     '},
  'keywords': {



                      },
}
```

QCSchema DEFINES DATA LAYOUTS

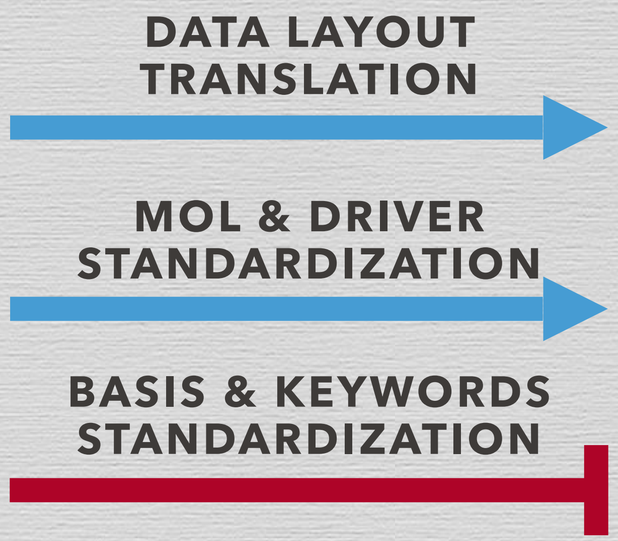
and text advice, nothing more

- **LAYOUT** JSON Schema defines a data layout and some minimal type and count checking.
- **CONVENTIONS** JSON Schema can't check conventions you rely upon like AU units, CCA ordering, center-of-mass positioning

DOMAIN-SPECIFIC ASCII

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

GAMESS Input



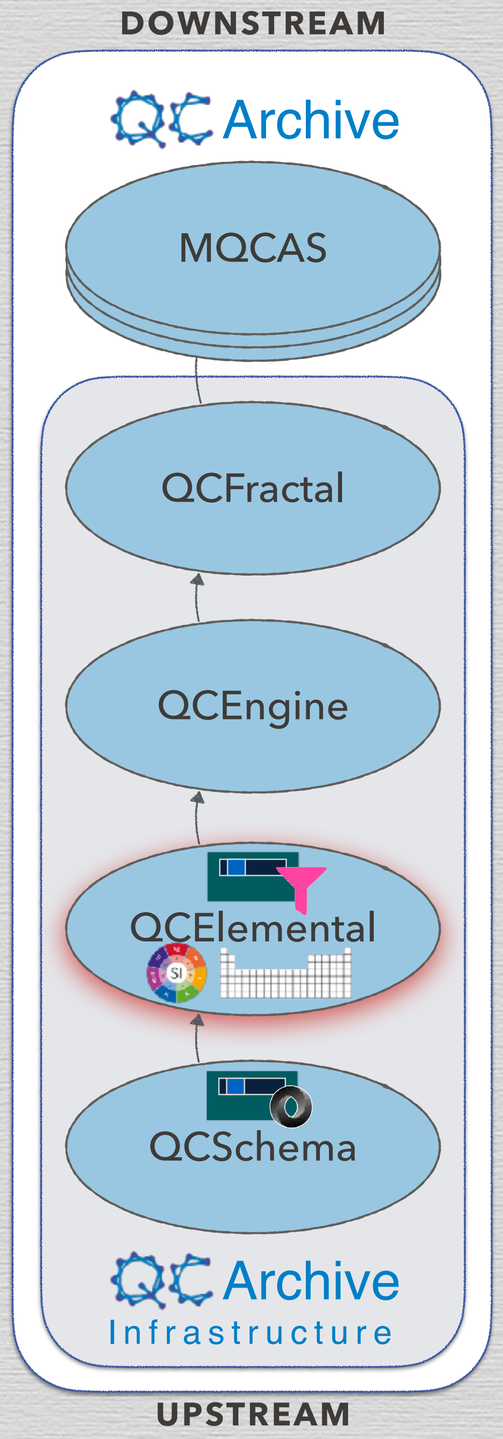
QCSchema: AtomicInput

```
{
  'molecule': {
    'symbols': ['N', 'H', 'H'],
    'geometry': [0.000, ..., 1.014],
    'molecular_multiplicity': 2},
  'driver': 'energy',
  'model': {
    'method': 'ccsd',
    'basis': 'accd'},
  'keywords': {
    'contrl__ispher': 1,
    'contrl__scftyp': 'rohf',
    'ccinp__ncore': 0},
}
```

GAMESS QCEngine Input

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

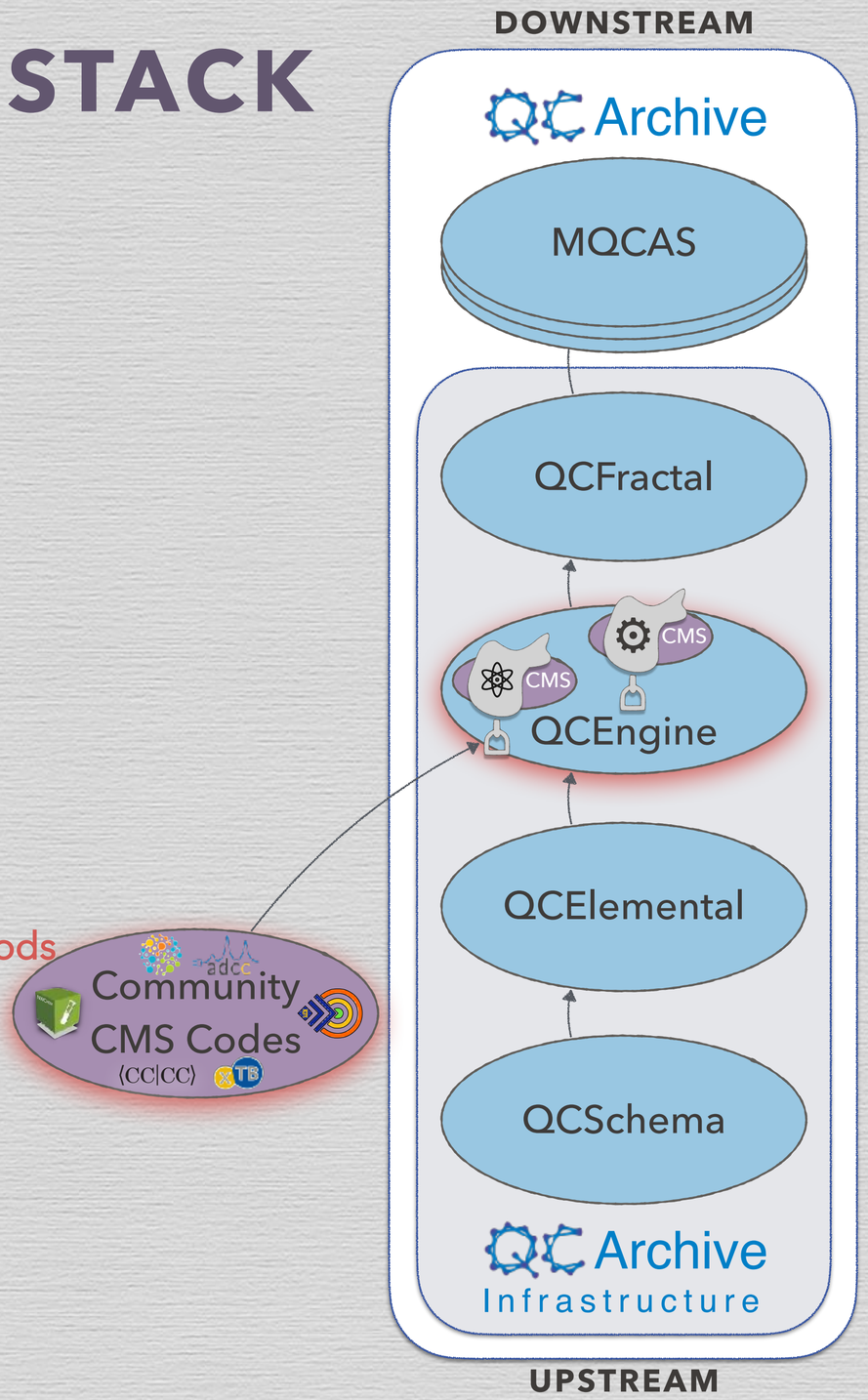
NIST CODATA, covalent and vdW radii, Periodic Table

- **STRUCTURED DATA** processed into light Python API directly from NIST structured data, if available.
- **PRECISION** original significant figures value always available
- **VERSIONED** store& upgrade: when reproducibility demands old versions, accessing both helps transitions.
- **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

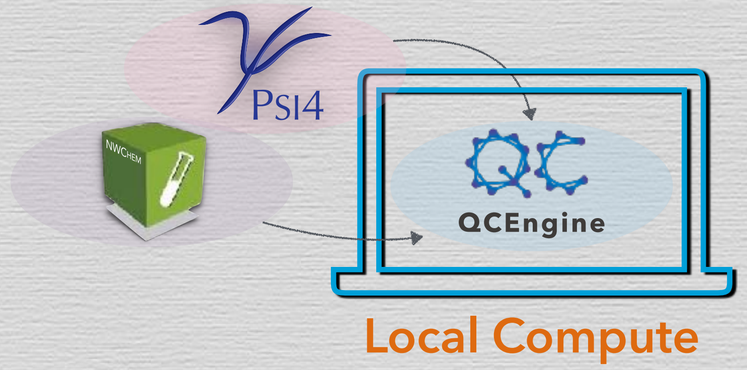
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

- 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

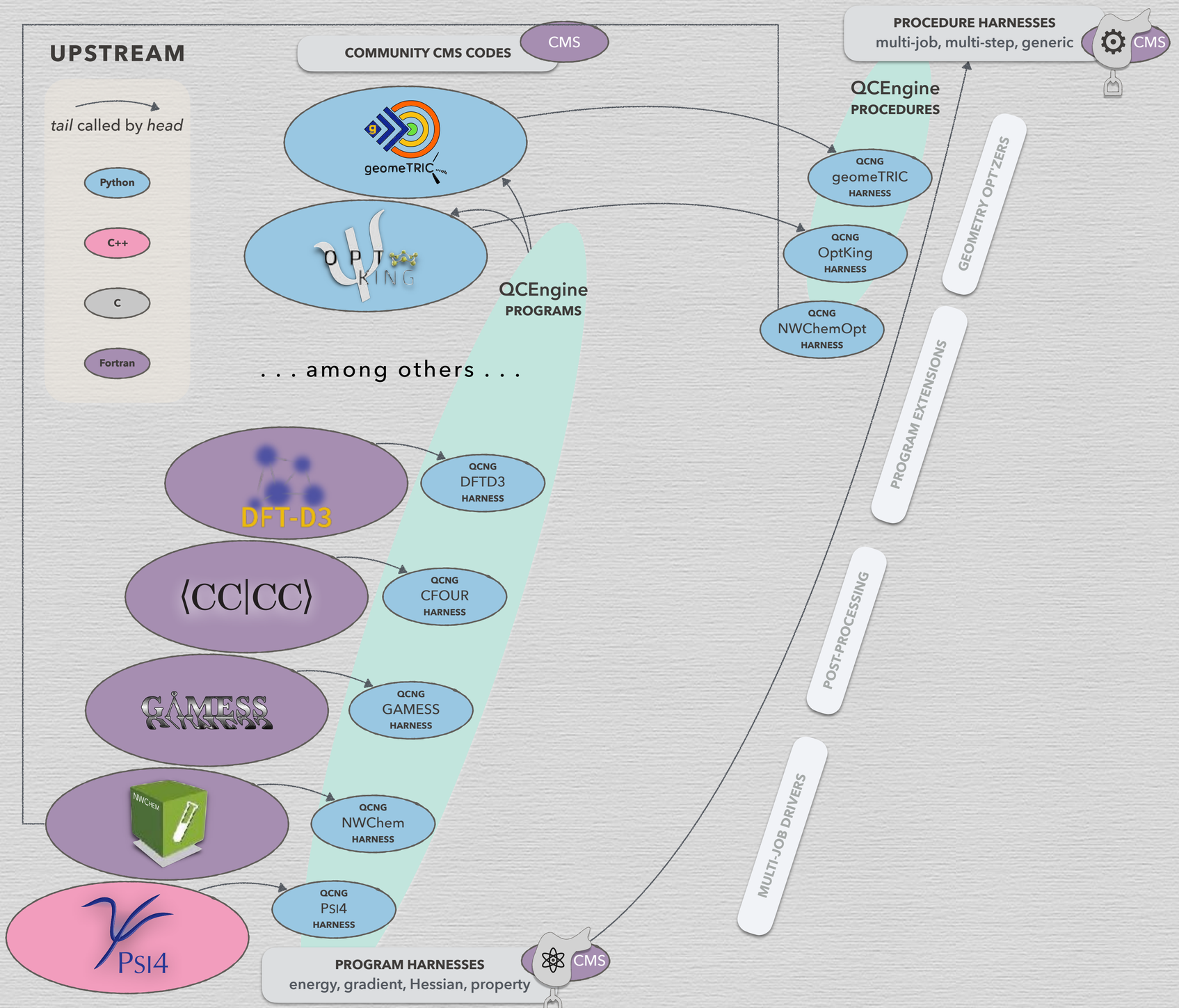
```
                                  cfour
qcengine.compute(atomicinput, "molpro", task_config({"nnodes": 4, "memory": 10})
                                  psi4
                                  dftd4
```

QCENGINE: AVAILABLE PROGRAM HARNESSES

github.com/MolSSI/QCEngine

CMS Program	QCEngine Data Interchange Mode			
	API	XML / JSON / BIN	TXT	Indep. QCSk
Quantum Chemistry	adcc	CFOUR	CFOUR	
			GAMESS	
		Molpro		
		MRChem		
			NWChem	
	Psi4			Psi4
		Q-Chem	Q-Chem	
		Qcore		
	TeraChem		TeraChem	TeraChem
			Turbomole	
Semi-Empirical			MOPAC	
	xtb			xtb
Molecular Mechanics	OpenMM			
	RDKit			
Analytical Corrections	DFTD3		DFTD3	DFTD3
	DFTD4			DFTD4
			gCP	
			MP2D	
Machine Learning	TorchANI			

- **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

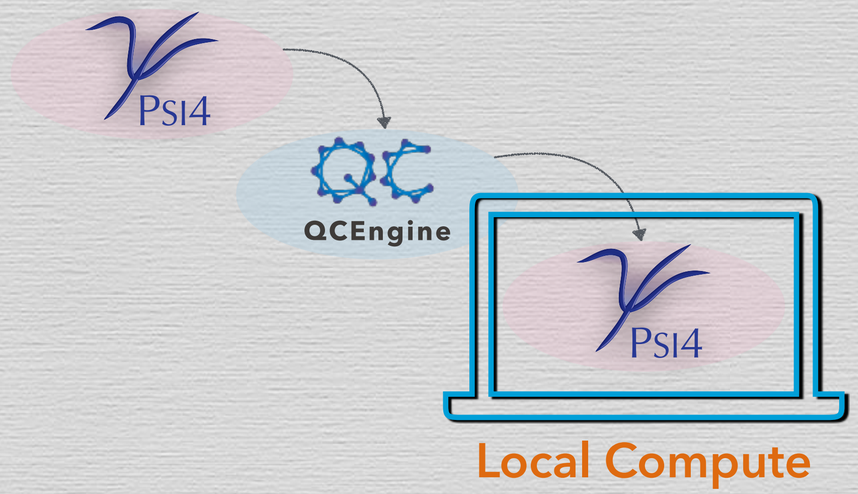
- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also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

DOWNSTREAM

(3) Psi4 DISTRIBUTED DRIVER, INTERNAL

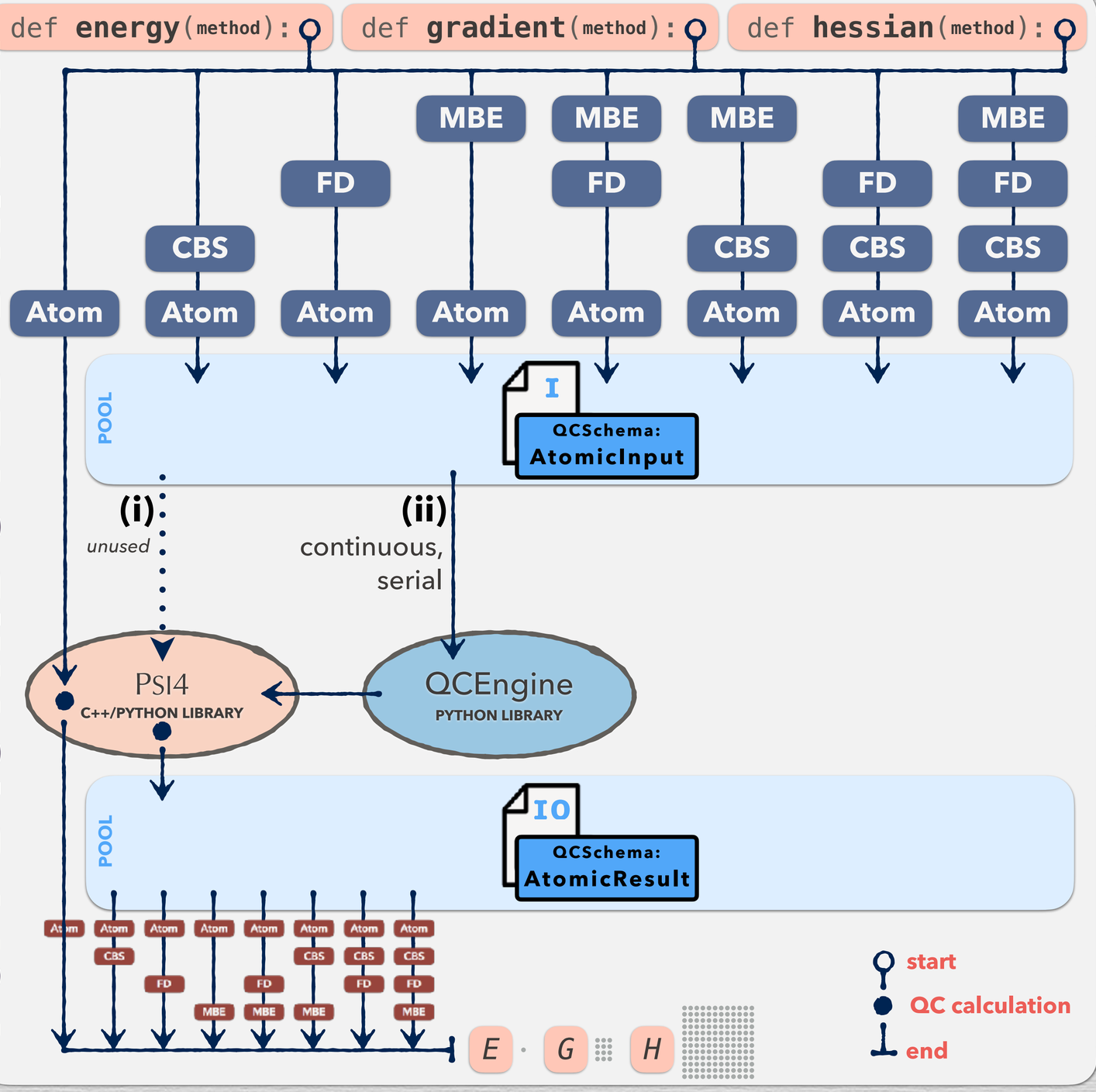
still capable, now orderly and extensible

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



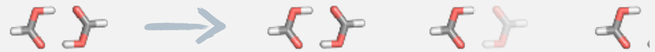
- **AVAILABLE** now with Psi4 & QCEngine since v1.6, May 2022.
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by sum of all calcs since run sequentially.
- **RETRIEVAL** from filesystem difficult since many calcs in aggregated outfile.
- **FLEXIBILITY** limited to Psi4 only.

PSI4 DISTRIBUTED DRIVER



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

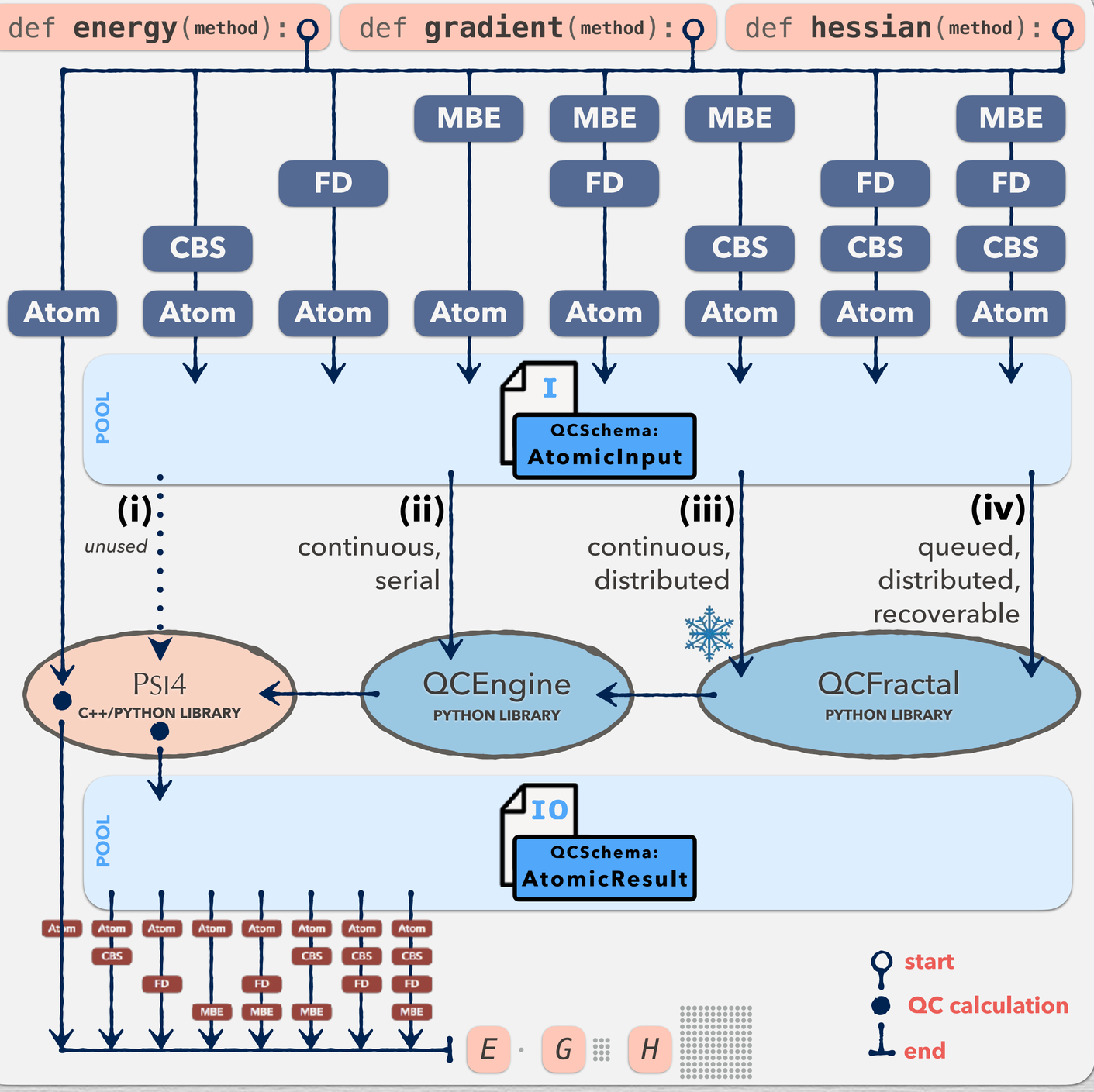
ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

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

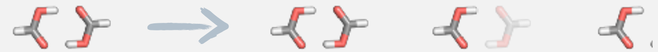
ASM Return analytic energy, gradient, or Hessian.

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 ():

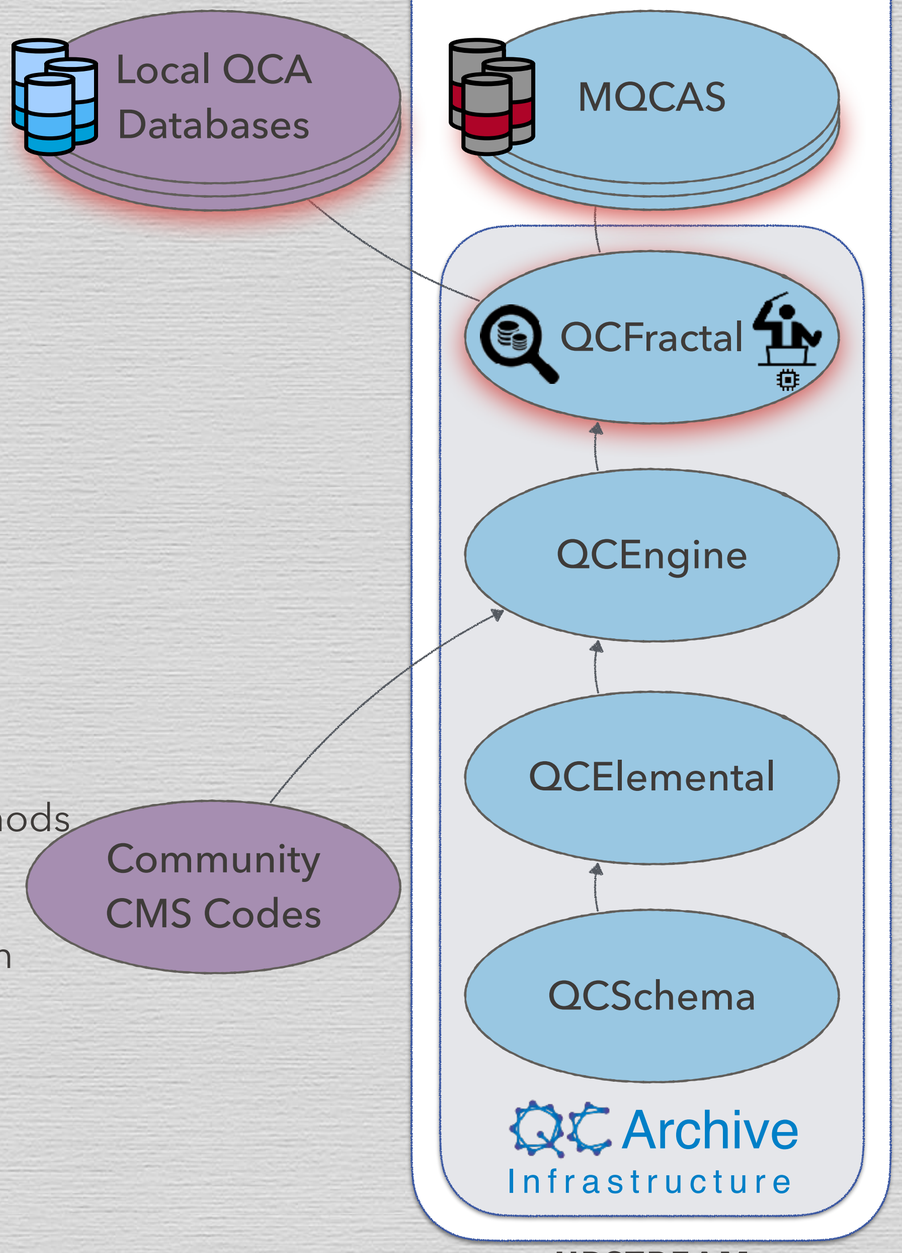
PLAN **molecule** & **method** unchanged. `return qcschema` QCSchema

ASM Return analytic energy, gradient, or Hessian.

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

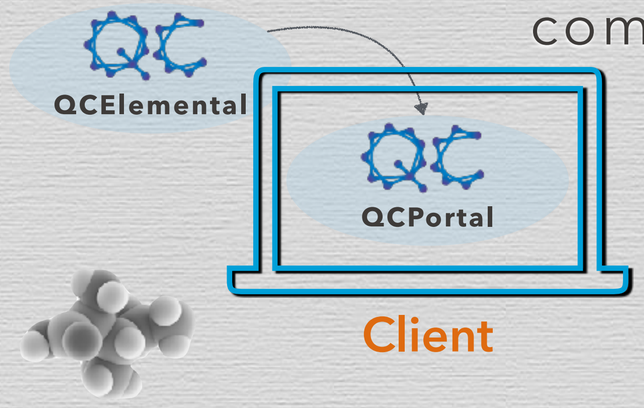
- the difficult part – coded QC methods
- structured output uncommon
- DSL input; API/schema uncommon

UPSTREAM

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org

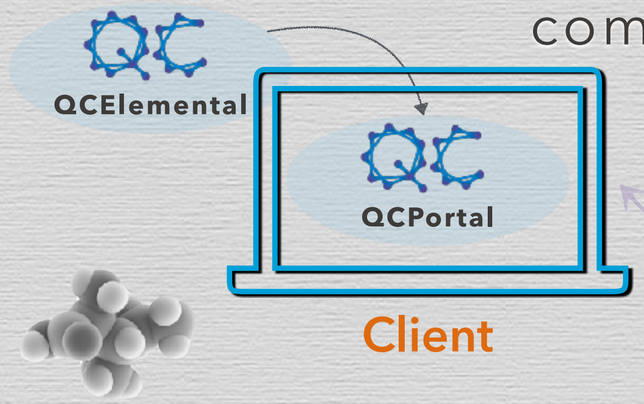


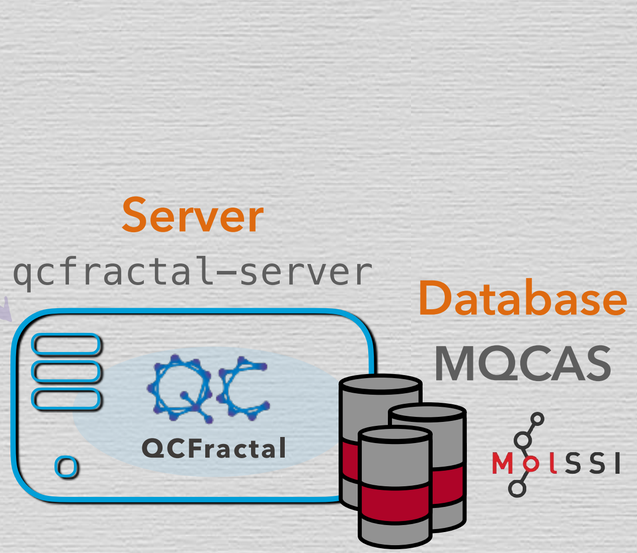
- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- Completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE -** Up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC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

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



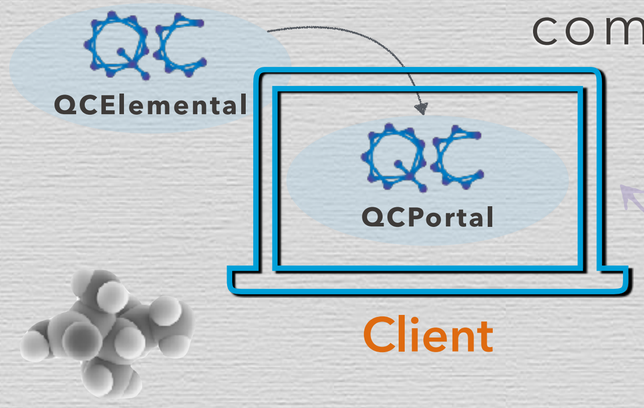


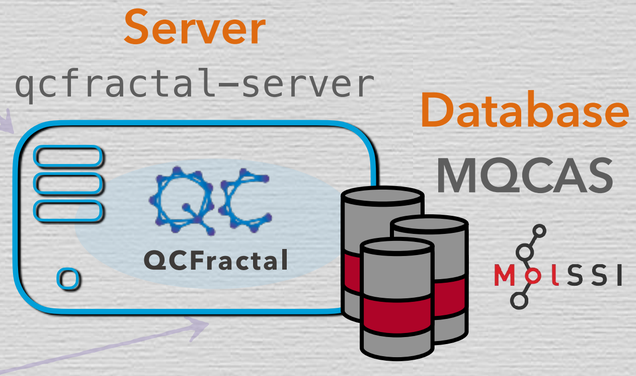
- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- Completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE -** Up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC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

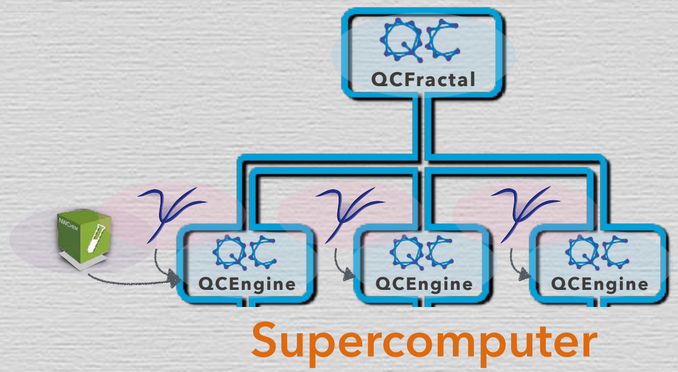
QCArchive OVERVIEW

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

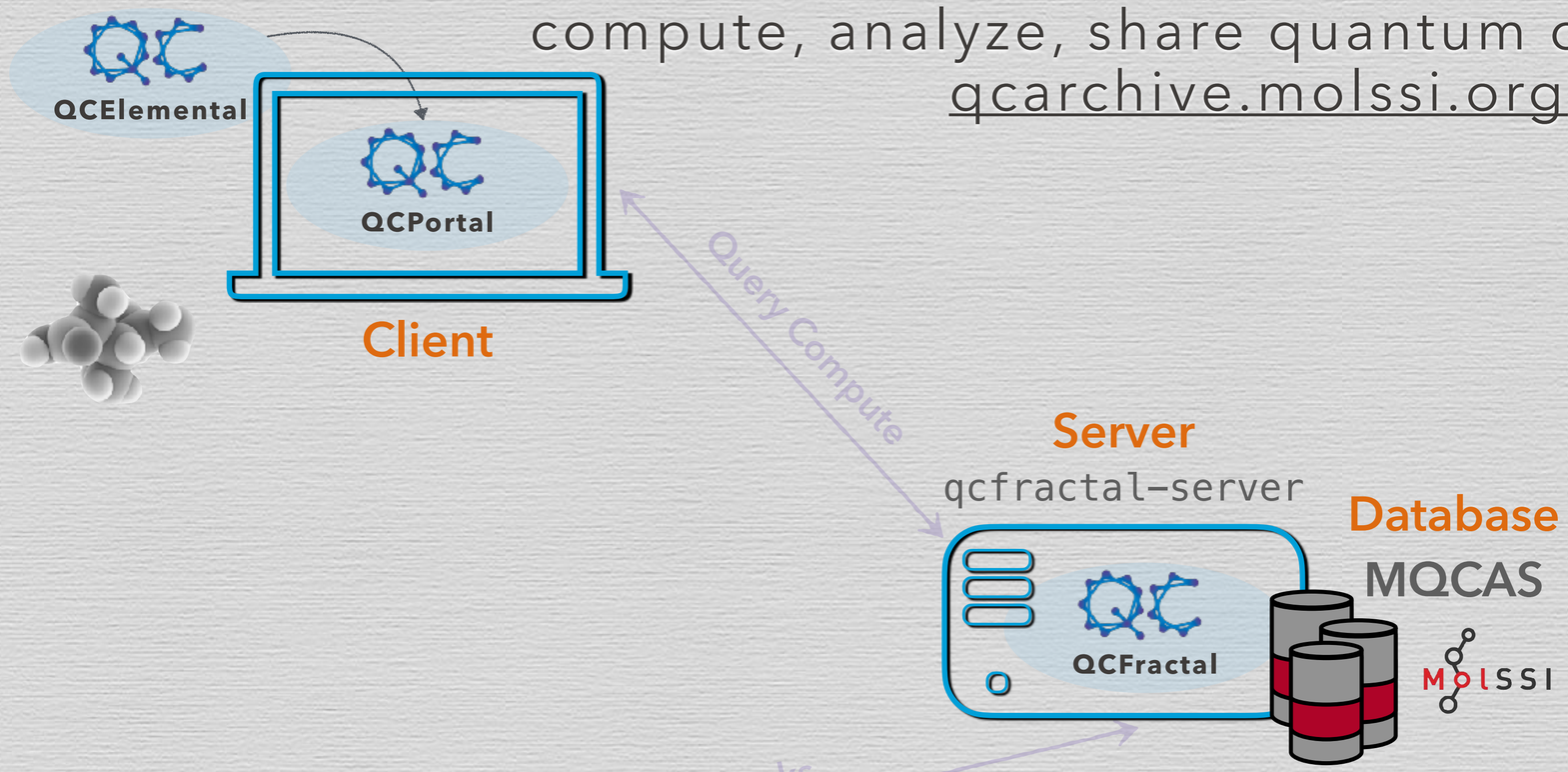
`qcfractal-manager` **Distributed Compute**

worker **Compute Nodes**

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org



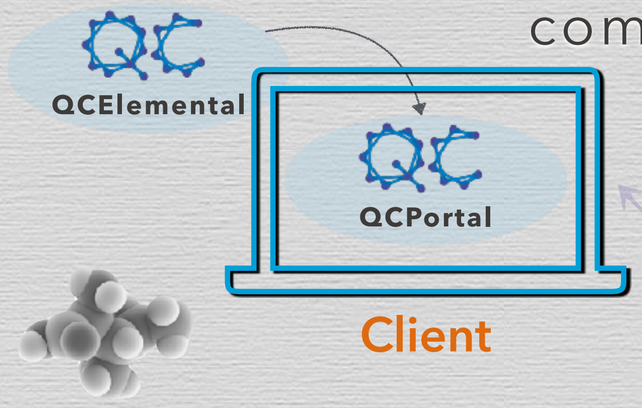
- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- Completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE -** Up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC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

`qcfractal-manager`	Distributed Compute
adapter	Task Management
scheduler	Task Scheduling
worker	Compute Nodes

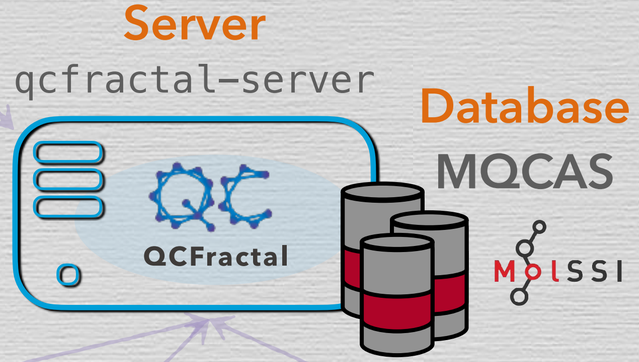
QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org

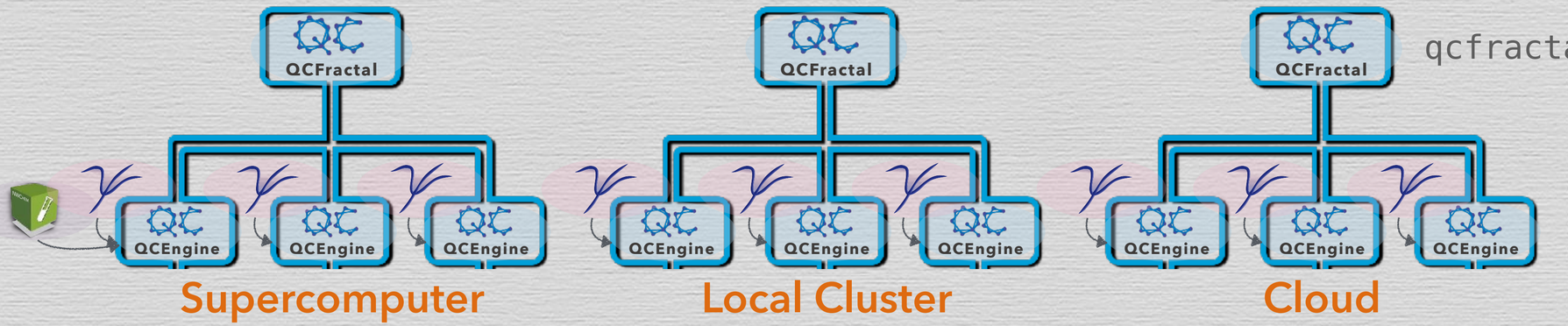


Query Compute



New Tasks

Complete Tasks



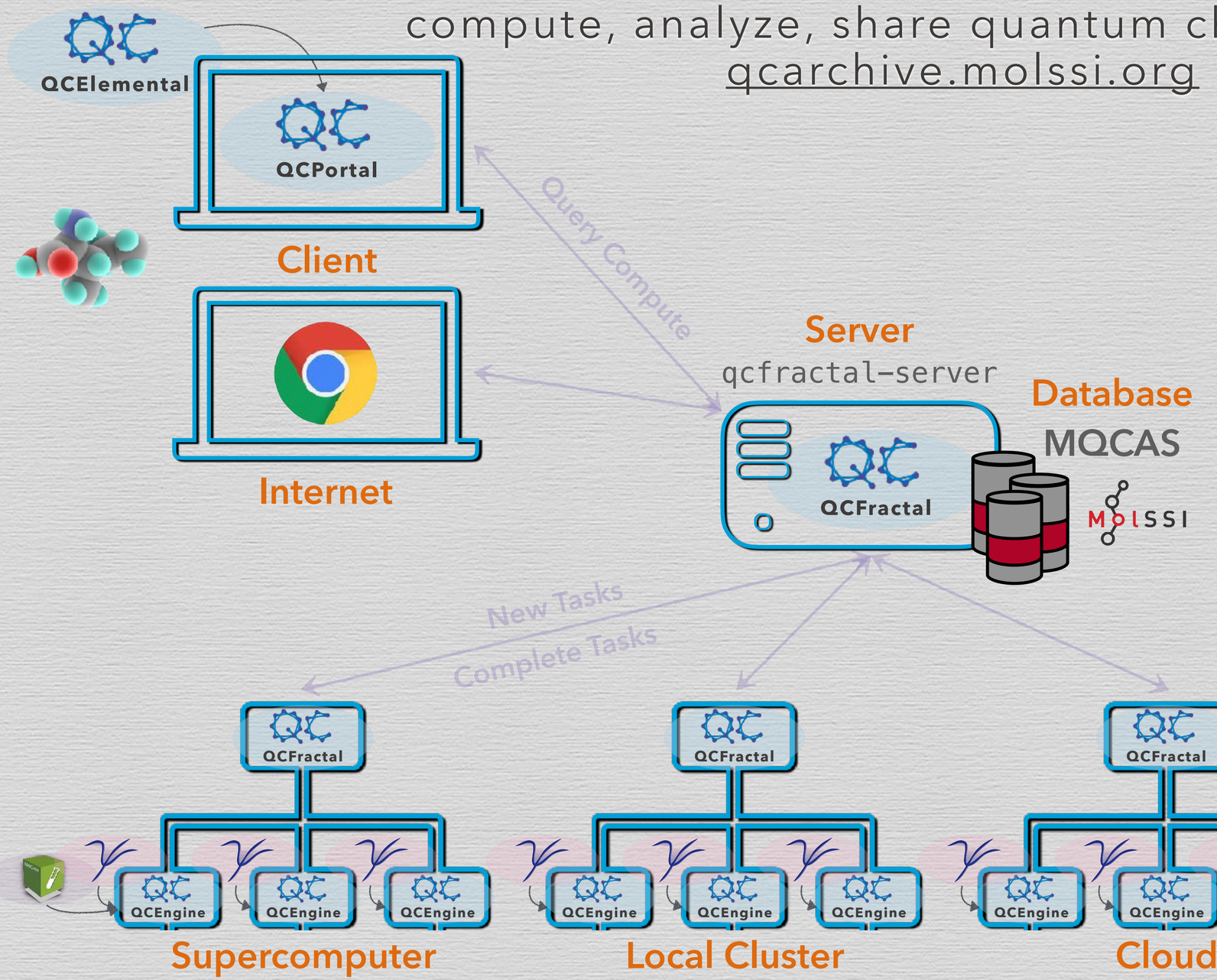
- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- Completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE -** Up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC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

`qcfractal-manager`	Distributed Compute
adapter	Task Management
scheduler	Task Scheduling
worker	Compute Nodes

QCArchive OVERVIEW

compute, analyze, share quantum chemistry data

qcarchive.molssi.org

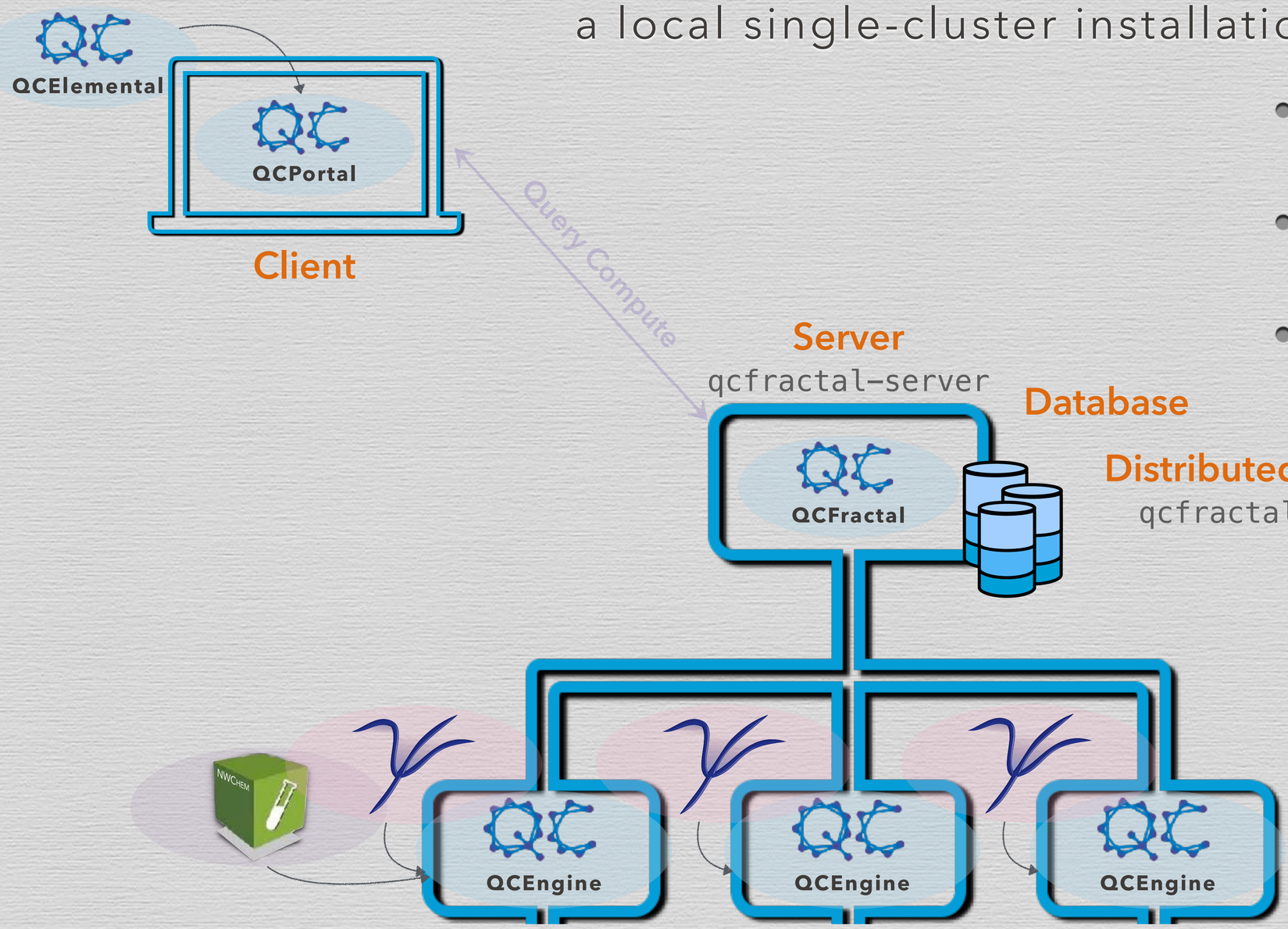


- hybrid **COMPUTE** and **DATA** management tool for quantum chemistry
- **SHARE** and **COLLABORATE** structured data
- **EASE OF USE** for non-specialist
- Completely installable from **CONDA**
- **MULTI-PHYSICAL-SITE** compute
- **SCALE -** Up to 500 tasks/second, 300,000 concurrent tasks
- **ELASTIC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

`qcfractal-manager`	Distributed Compute
adapter	Task Management
scheduler	Task Scheduling
worker	Compute Nodes

QCARCHIVE, SMALLER

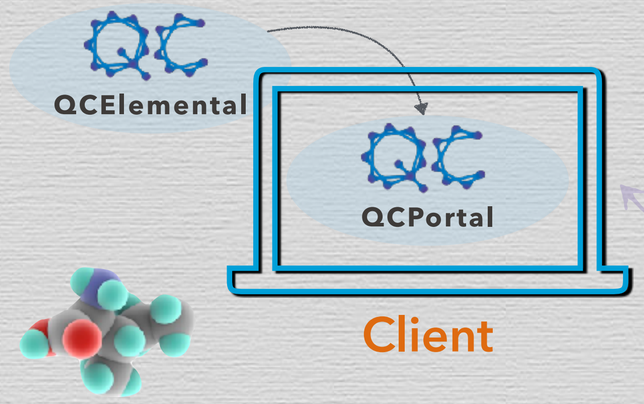
a local single-cluster installation



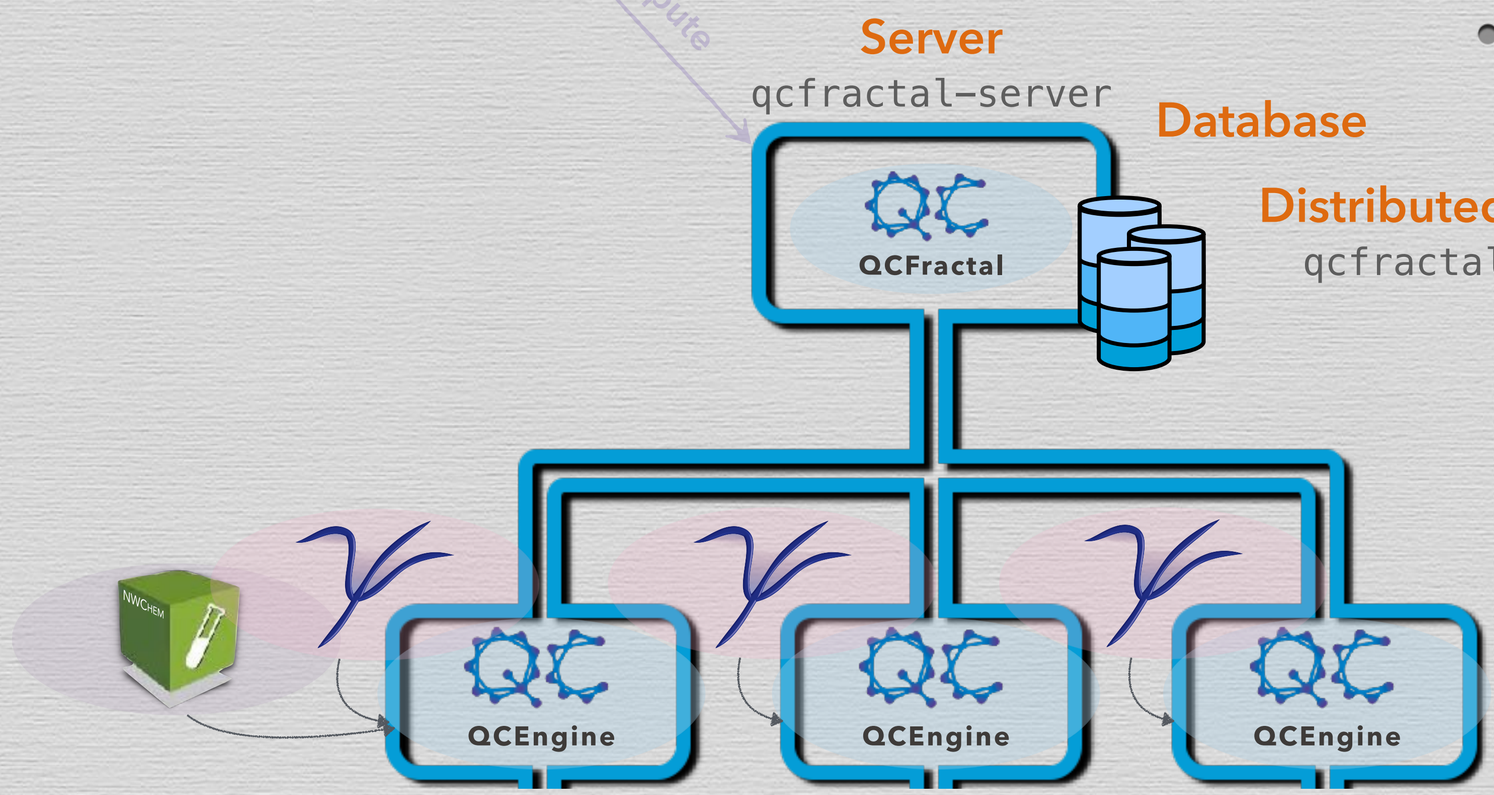
- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal and database for task submission and orchestration
- **LOCAL** database fully powerful as MQCAS but locally controlled

QCARCHIVE, SMALLER

a local single-cluster installation

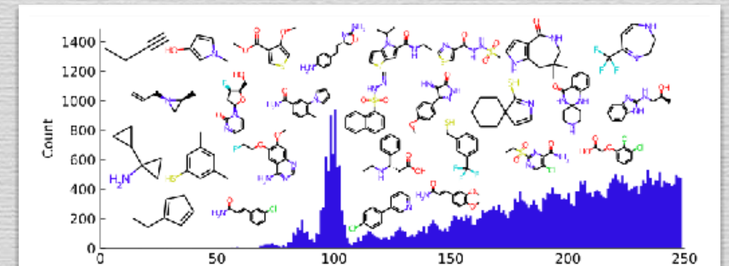


- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal and database for task submission and orchestration
- **LOCAL** database fully powerful as MQCAS but locally controlled



QCARCHIVE FOR THROUGHPUT & REPRODUCIBILITY

HPC message-passing neural network computations preserved on public server



50K DRUG-LIKE MOLECULES

MolSSI QCArchive Server

< 24 Hours

256 Cluster Nodes Running Psi4

ML Model Development

DFT Properties
(Energies, Orbitals, Multipoles, etc.)

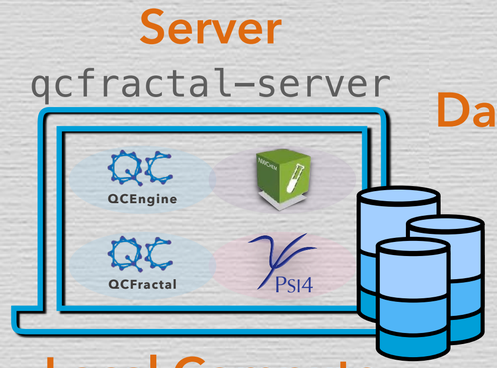
Zach Glick
GaTech

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

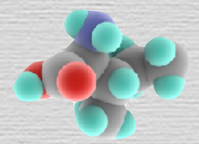
- **QCARCHIVE INFRASTRUCTURE** software stack unchanged
- **COLOCATE** QCFractal, database and QC codes for self-contained setup
- **SNOWFLAKE** a still-simpler mode where the database does not persist



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

Distributed Compute

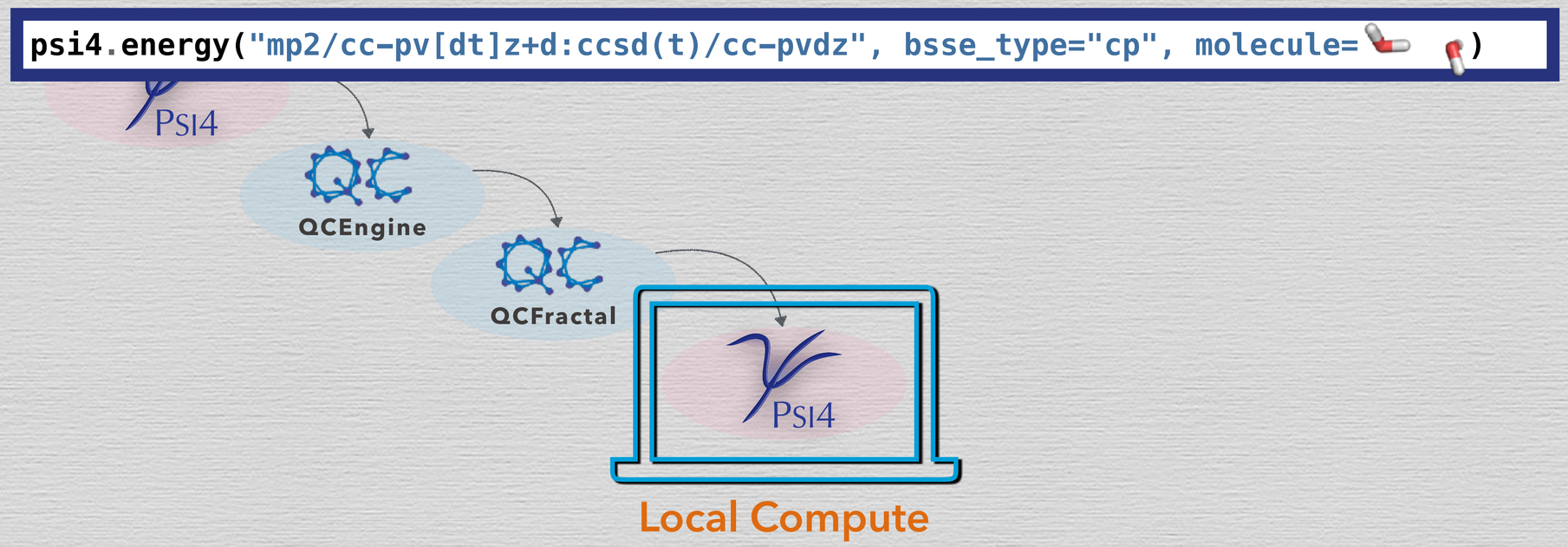
`qcfractal-manager`

Local Compute

worker

(4) Psi4 DISTRIBUTED DRIVER, QCArchive

distributed, searchable, and error-resistant

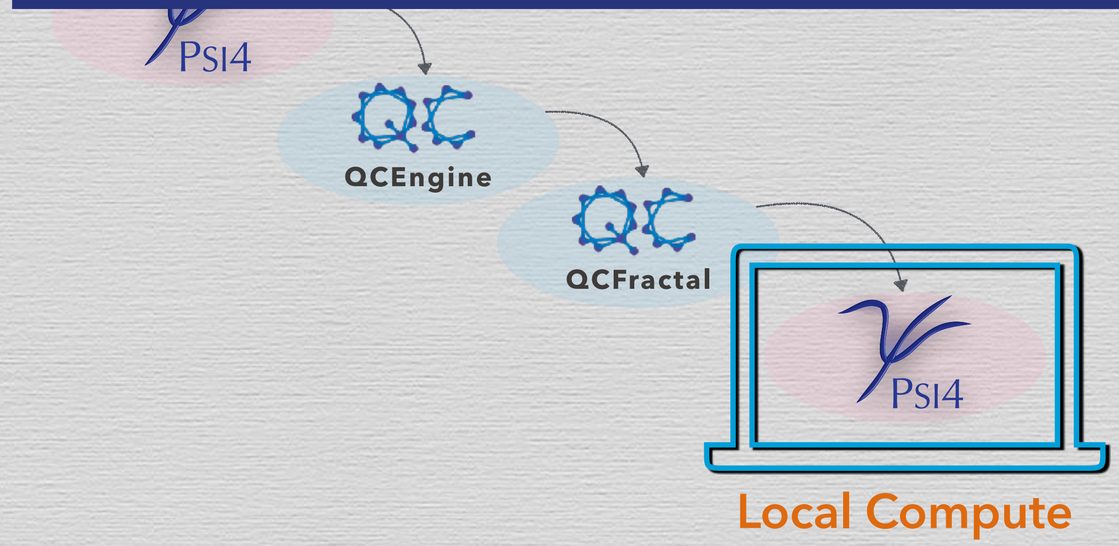


- **AVAILABLE** now with Psi4 & QCEngine & QCFractal since v1.6, May 2022.
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** limited to Psi4 only.

(4) 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=  )
```



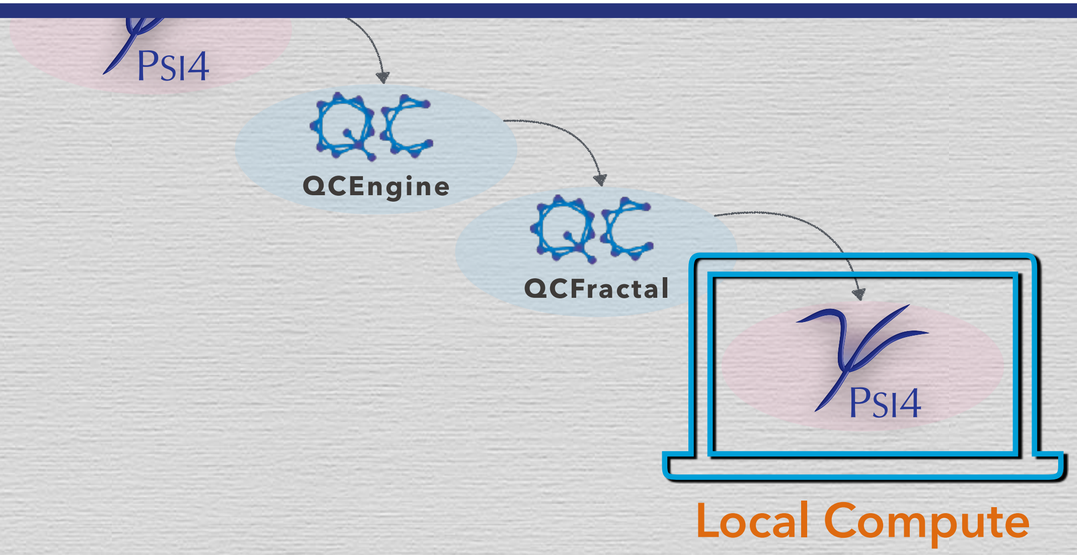

```
import psi4
from qcfractal import FractalServer

server = FractalServer()
client = server.client()

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

qcsk = plan.get_results(client)
print(qcsk.return_result)  # CP IE
```

- **AVAILABLE** now with Psi4 & QCEngine & QCFractal since v1.6, May
- **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.

(4) 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 qcfractal import FractalServer

server = FractalServer()
client = server.client()

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

qcsk = plan.get_results(client)
print(qcsk.return_result)  # CP IE
```

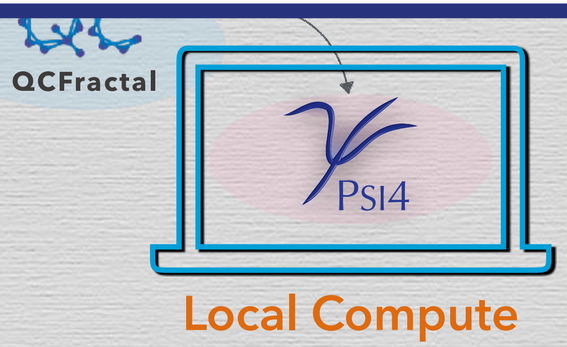
```
plan = psi4.energy("CCSD(T)/cc-pVDZ",
                   return_plan=True,
                   molecule=  )
plan.compute(client)  # free! calcs in database

qcsk = plan.get_results(client)
print(qcsk.return_result)  # CCSD(T) energy
```

- **AVAILABLE** now with Psi4 & QCEngine & QCFractal since v1.6, May
- **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.

(4) 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= )
psi4.gradient("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule= )
psi4.hessian("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", bsse_type="cp", molecule= )
```

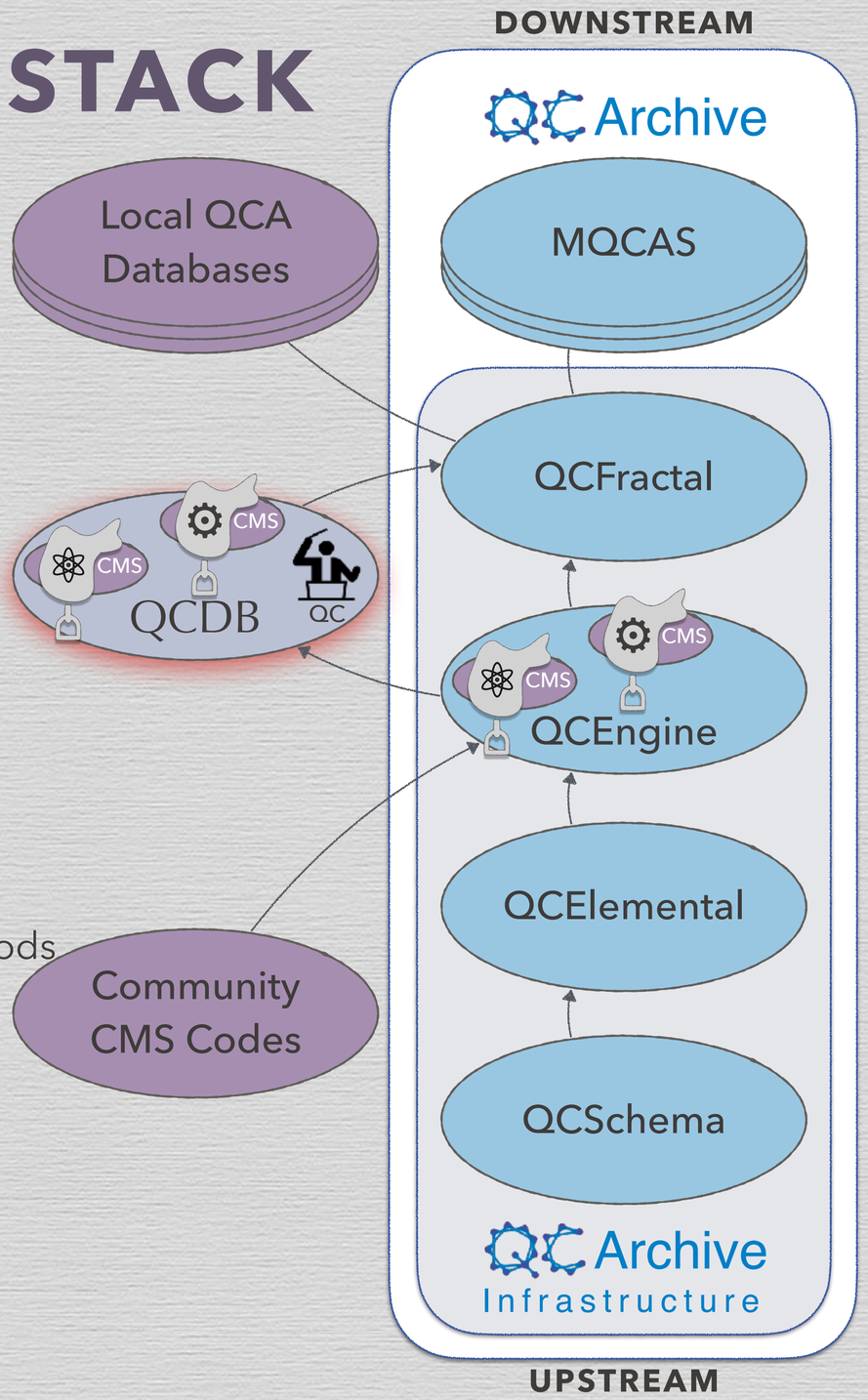



```
import psi4
server = FractalServer()
client = server.client()

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

qcsk = plan.get_results(client)
print(qcsk.return_result)  # CP IE
```

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

qcsk = plan.get_results(client)
print(qcsk.return_result)  # CCSD(T) energy
```

- **AVAILABLE** now with Psi4 & QCEngine & QCFractal since v1.6, May
- **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.

QCArchive STACK



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

- DSL or unified input syntax
- multi-QCprogram workflows
- standardization layers

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

- database, universally queryable
- 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

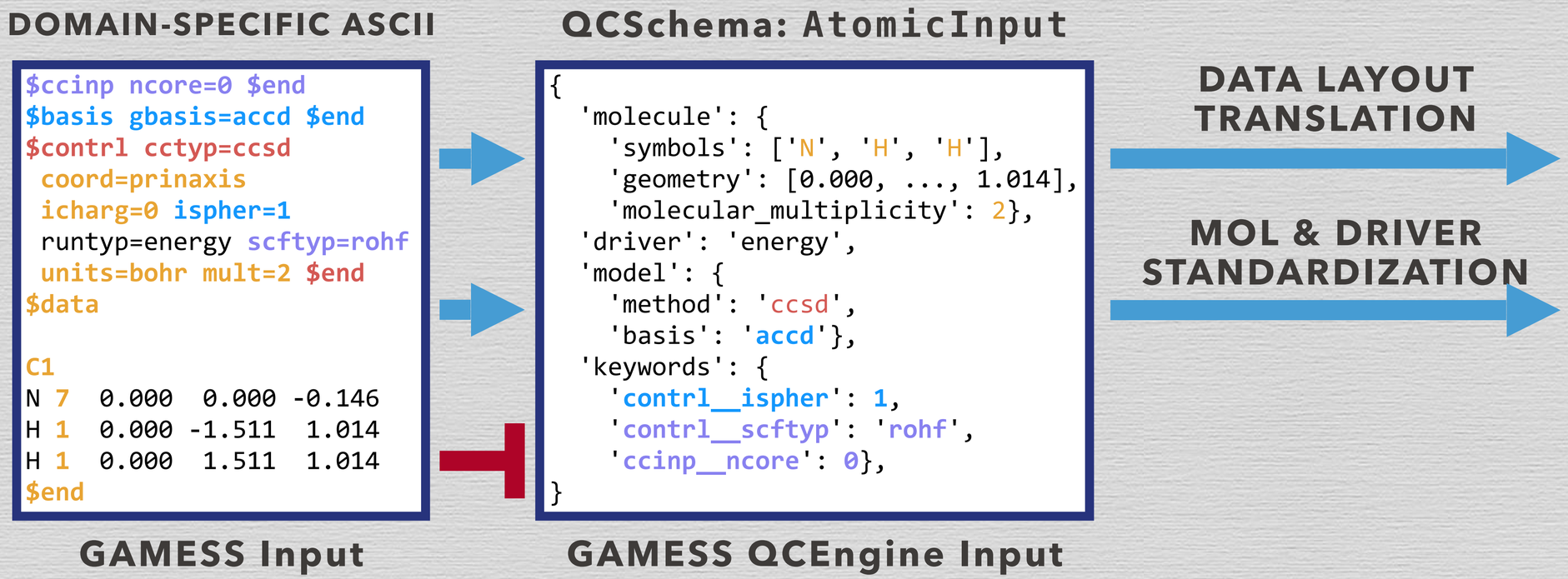
- Python QCSchema implementations & validation
- NIST periodic table & physical constants
- molecule parsing, validation, export

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

QCSchema DEFINES DATA LAYOUTS

QCDB unifies data contents

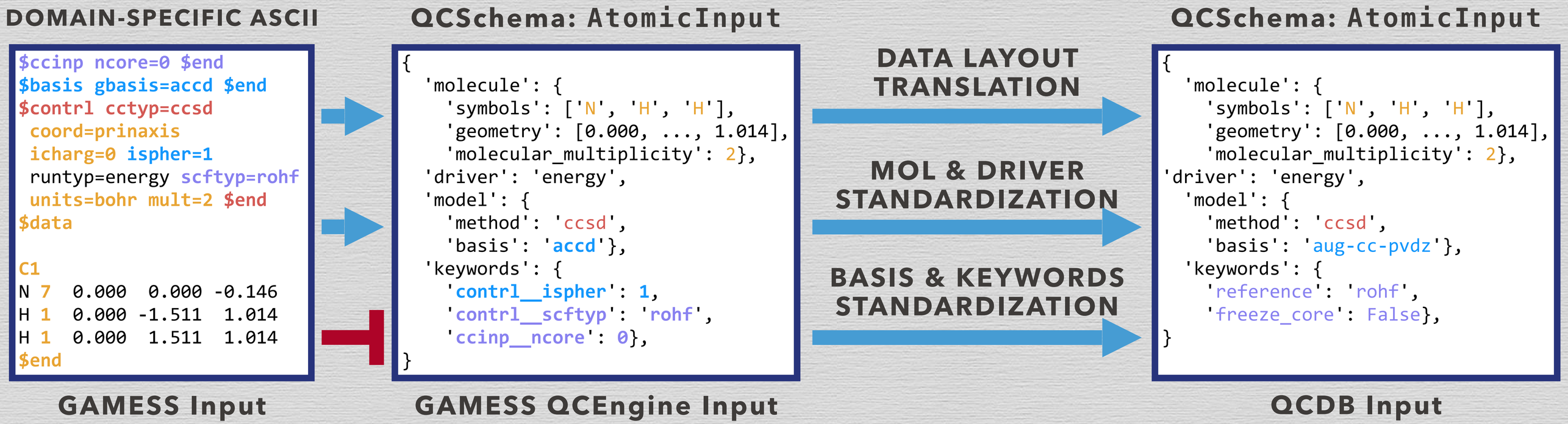
- **LAYOUT** JSON Schema defines a data layout and some minimal type and count checking.
- **CONVENTIONS** JSON Schema can't check conventions you rely upon like AU units, CCA ordering, center-of-mass positioning

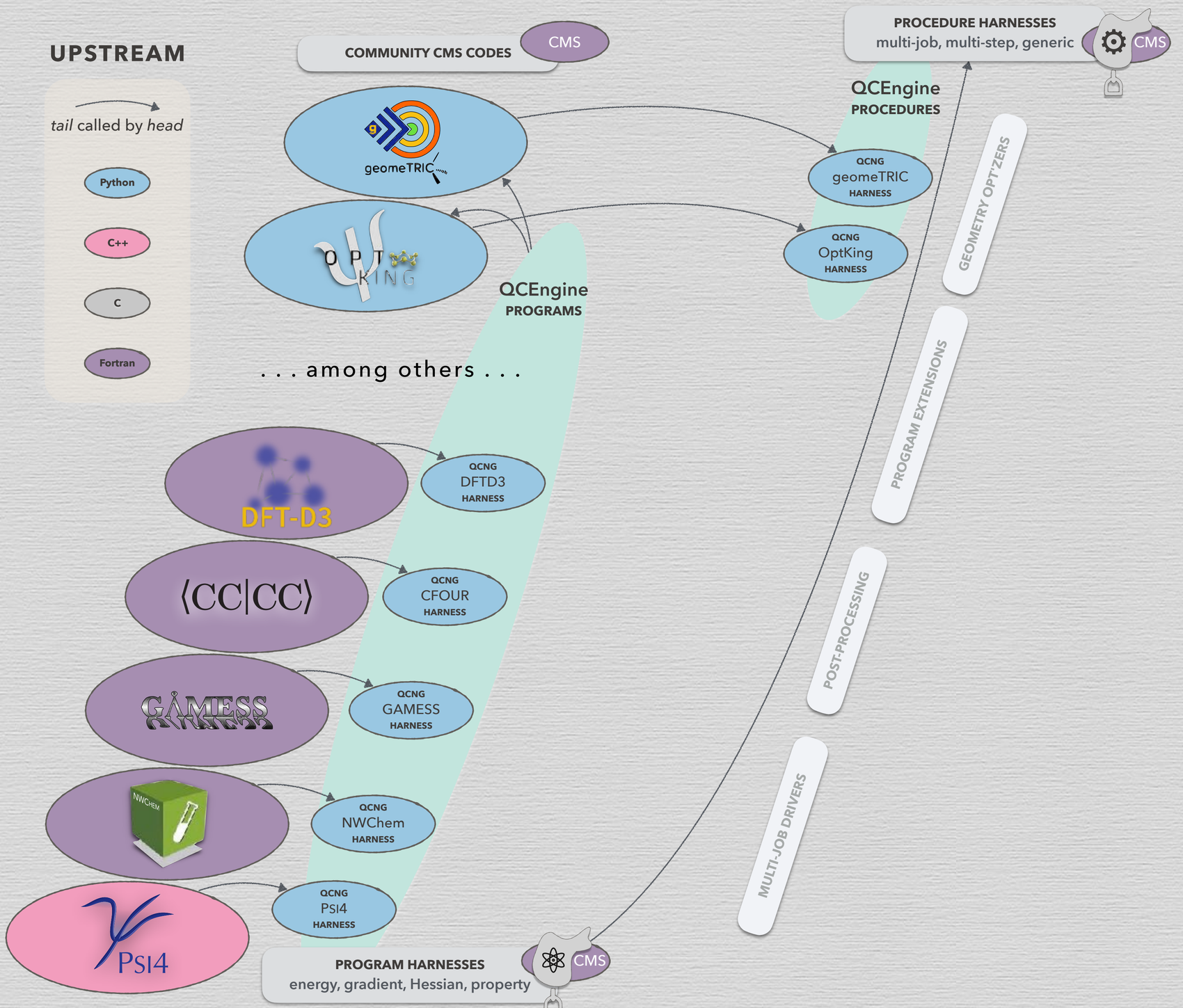


QCSchema DEFINES DATA LAYOUTS

QCDB unifies data contents

- **LAYOUT** JSON Schema defines a data layout and some minimal type and count checking.
- **CONVENTIONS** JSON Schema can't check conventions you rely upon like AU units, CCA ordering, center-of-mass positioning
- **QCDB UNIFIED CONTROL & KEYWORD TRANSLATION** allows users to focus on scientific choices, not DSL.
- **LOWER BARRIERS** to using a variety of codes, many with unique features.





QCENGINE: INTERNAL MAP

github.com/MolSSI/QCEngine

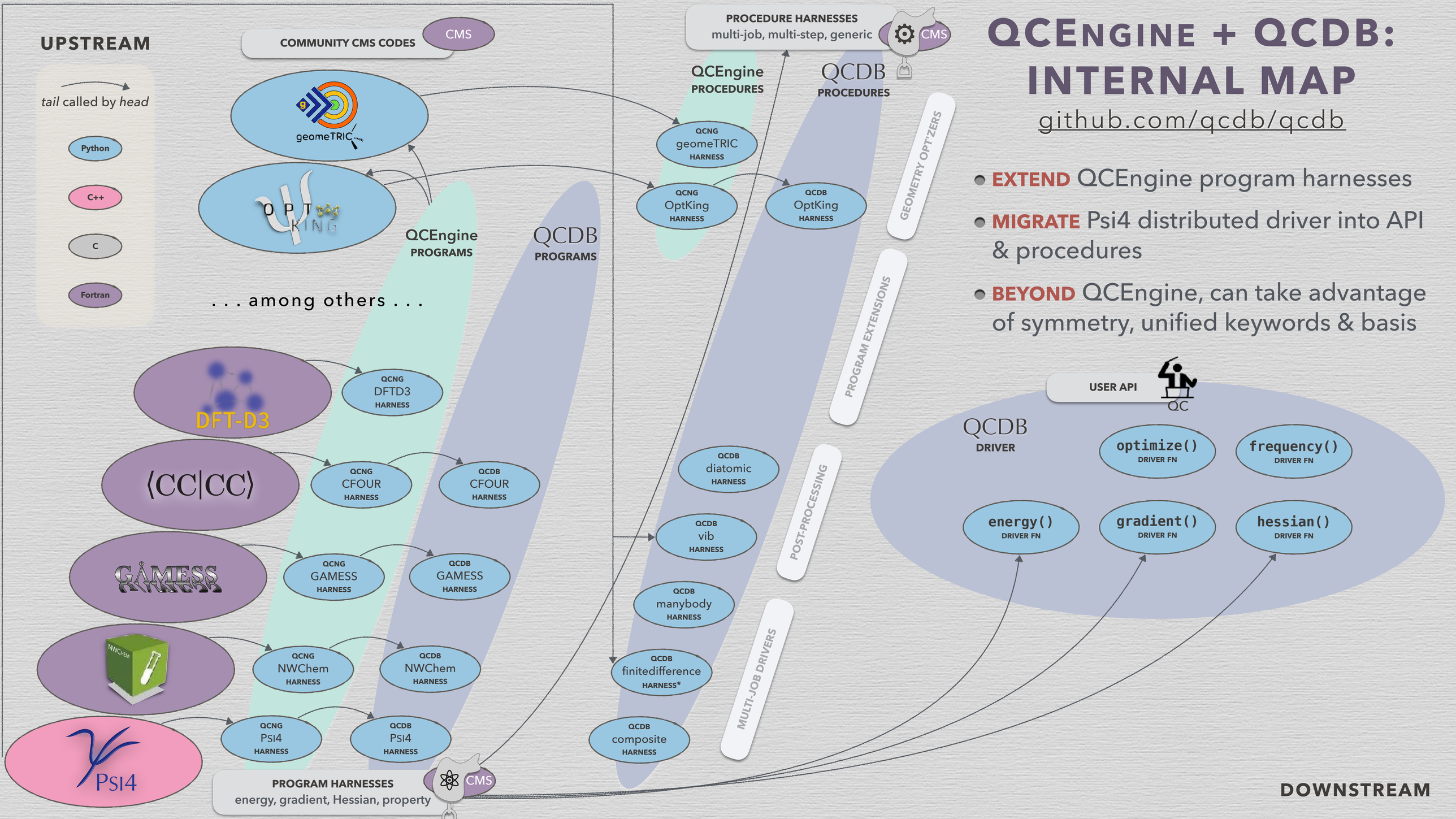
- **SCHEMA** runner
- **PROGRAMS** any analytic single point from CMS code
- **PROCEDURES** anything except analytic single point. So far, mostly optimizers: geometric, optking, pyberny, NWChem (int.).
- **PROGRAM CHECKS** indicate some methods accessible through QCSchema
- **CMS** primarily QM but also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

DOWNSTREAM

QCENGINE + QCDB: INTERNAL MAP

github.com/qcdb/qcdb

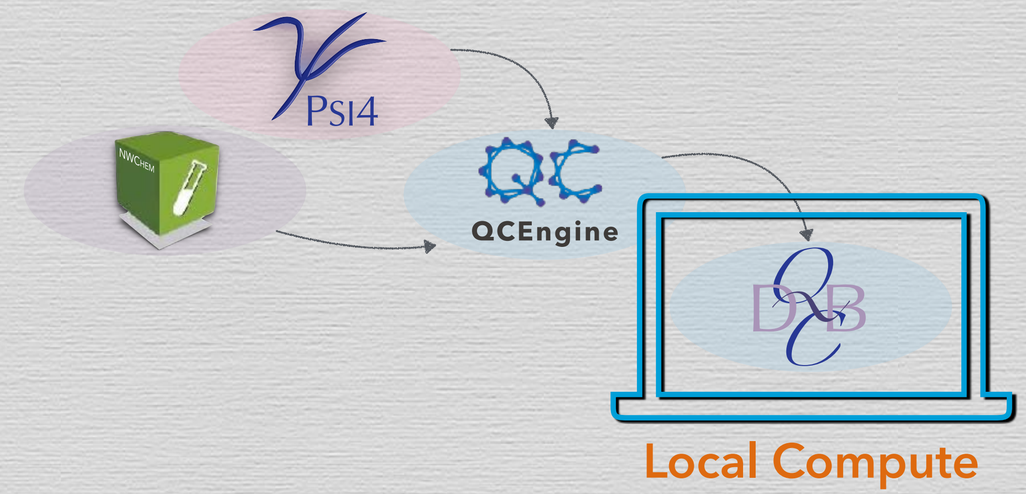
- **EXTEND** QCEngine program harnesses
- **MIGRATE** Psi4 distributed driver into API & procedures
- **BEYOND** QCEngine, can take advantage of symmetry, unified keywords & basis



(5) QCDB DISTRIBUTED DRIVER, INTERNAL

Psi4 procedures generalized to other QC programs

```
qcdb.energy("p4-mp2/cc-pv[dt]z+d:nwc-ccsd(t)/cc-pvdz", bsse_type="cp", molecule=  )
```

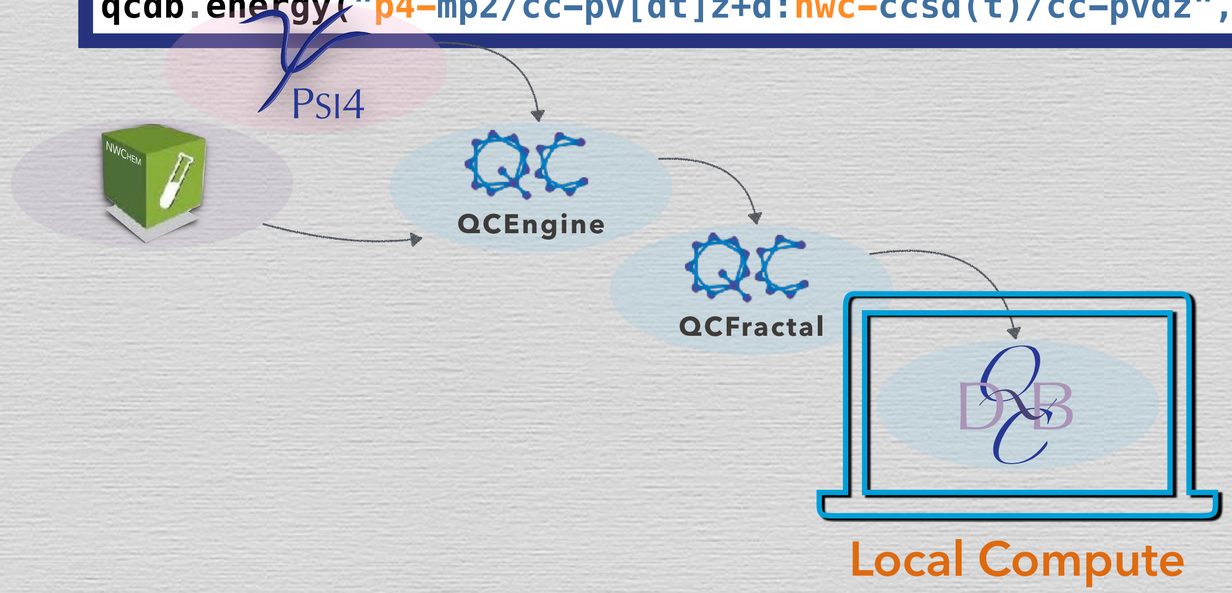


- **AVAILABLE** now with QCDB dev branch & Psi4 & QCEngine & NWChem (or CFOUR or GAMESS).
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by sum of all calcs since run sequentially.
- **RETRIEVAL** from filesystem difficult since many calcs in aggregated outfile.
- **FLEXIBILITY** to mix QC programs to access more or more efficient methods.

(6) QCDB DISTRIBUTED DRIVER, QCARCHIVE

distributed, searchable, error-resistant, *and* generalized

```
qcdb.energy("p4-mp2/cc-pv[dt]z+d:nwc-ccsd(t)/cc-pvdz", bsse_type="cp", molecule=      )
```

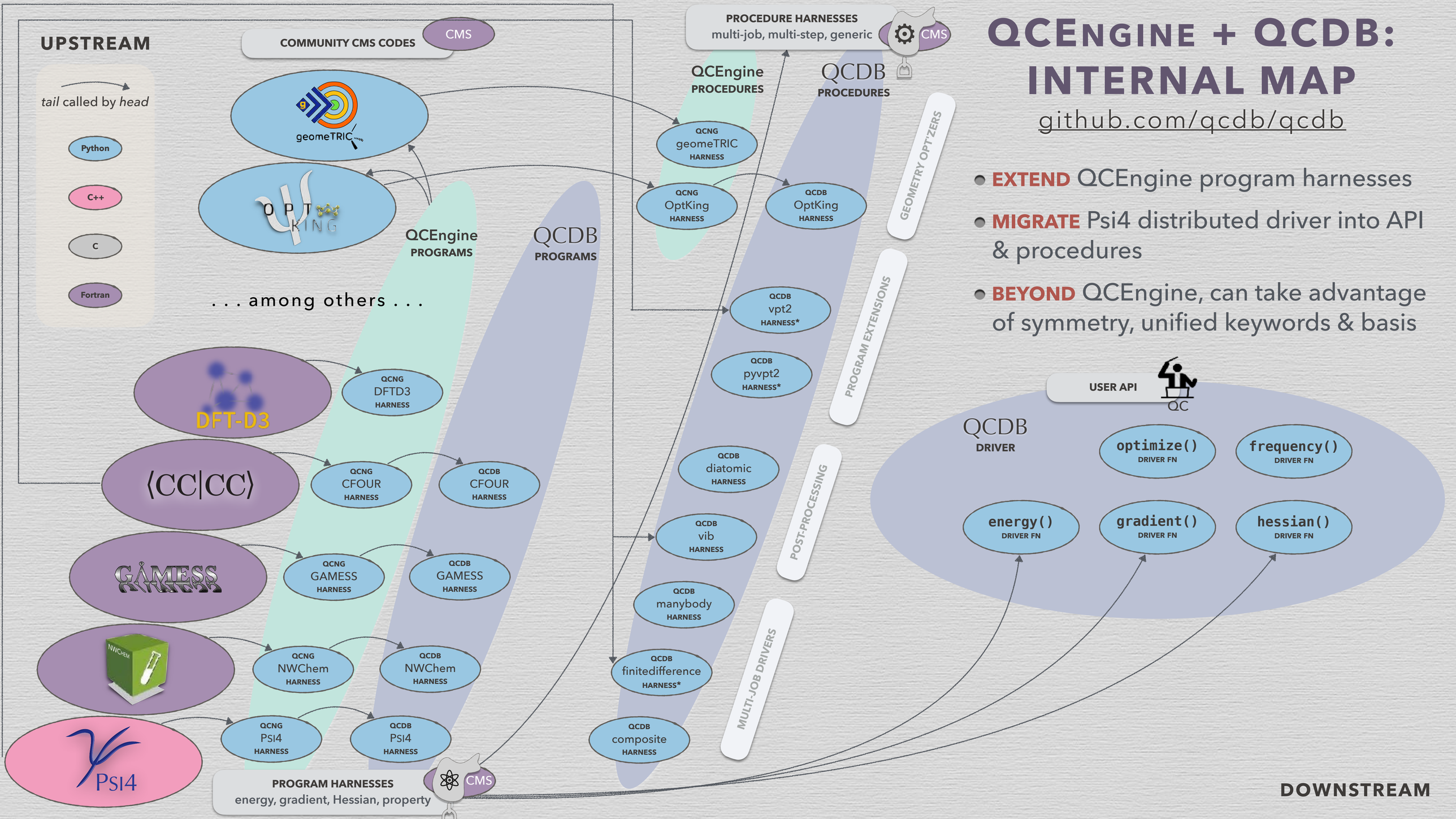


- **AVAILABLE** with QCDB dev branch & Psi4 & QCEngine & QCFractal & NWChem (or CFOUR or GAMESS).
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** to mix QC programs to access more or more efficient methods.

QCENGINE + QCDB: INTERNAL MAP

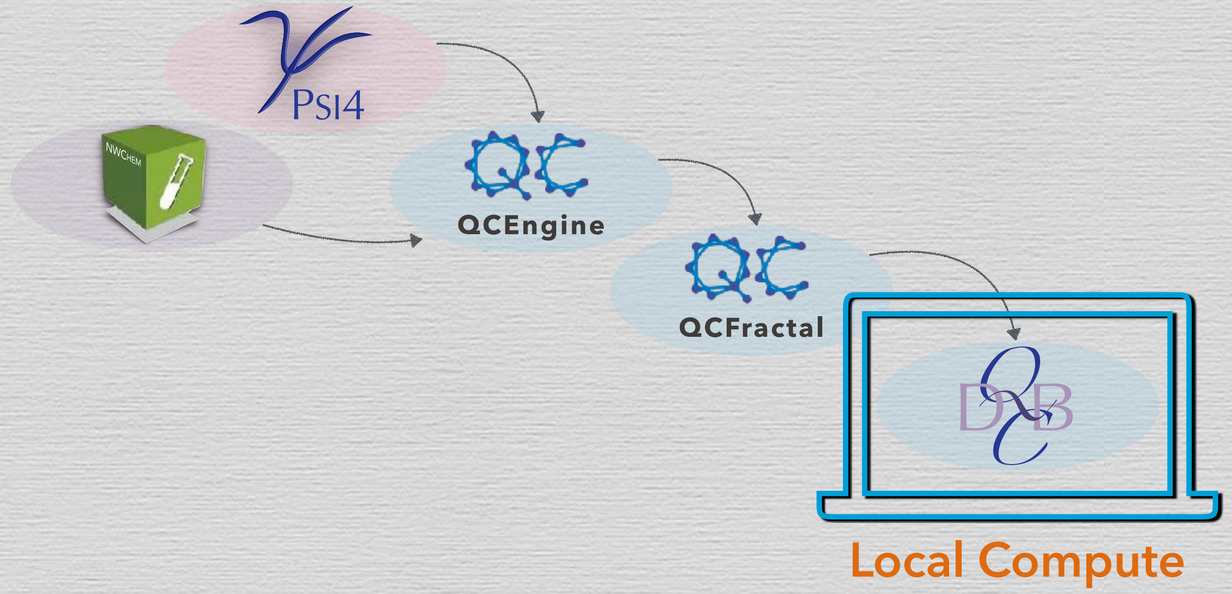
github.com/qcdb/qcdb

- **EXTEND** QCEngine program harnesses
- **MIGRATE** Psi4 distributed driver into API & procedures
- **BEYOND** QCEngine, can take advantage of symmetry, unified keywords & basis



(7) Psi4/QCDB multilevel analysis, QCArchive

distributed, searchable, and error-resistant

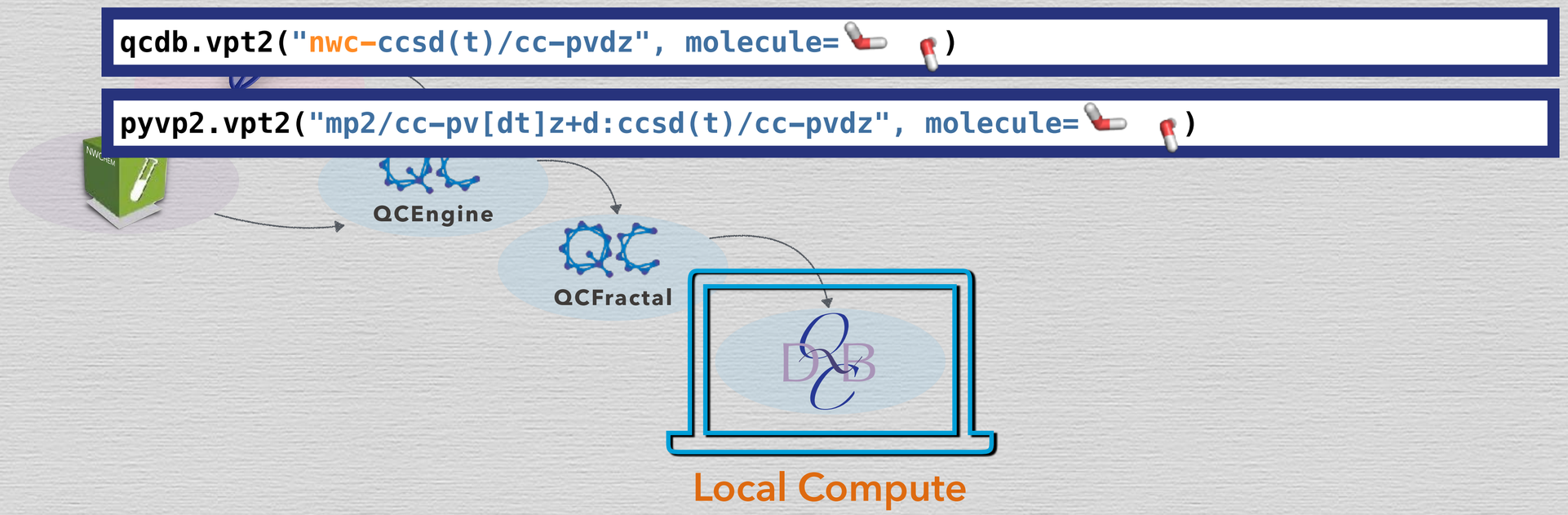


- pyvpt2: **AVAILABLE** with pyvp2 (unreleased) & Psi4 & QCEngine & QCFractal.
- qcdb.vpt2: **AVAILABLE** with QCDB dev branch & QCEngine & CFOUR & NWChem (or Psi4 or GAMESS).
- **SPECIFICATION** through single file. Procedure automated, so low risk of user error.
- **RATE-LIMITED** by single longest calc since QCFractal runs parallel in queue.
- **RETRIEVAL** from QCArchive database with query.
- **FLEXIBILITY** multiple implementations, calling multiple QC backends.

Philip Nelson
GaTech

(7) PSI4/QCDB MULTILEVEL ANALYSIS, QCARCHIVE

distributed, searchable, and error-resistant



- pyvpt2: **AVAILABLE** with pyvp2 (unreleased) & PSI4 & QCEngine & QCFractal.
- qcdb.vpt2: **AVAILABLE** with QCDB dev branch & QCEngine & CFOUR & NWChem (or PSI4 or GAMESS).
- **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

(7) PSI4/QCDB MULTILEVEL ANALYSIS, QCARCHIVE

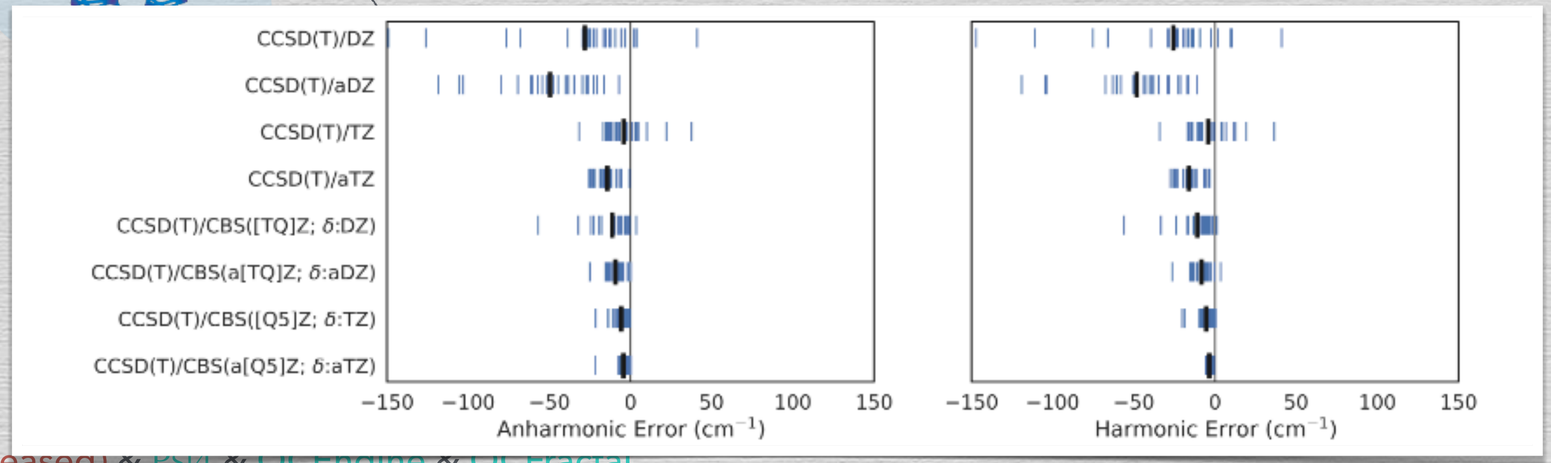
distributed, searchable, and error-resistant

```
qcdb.vpt2("nwc-ccsd(t)/cc-pvdz", molecule=  )
```

```
pyvp2.vpt2("mp2/cc-pv[dt]z+d:ccsd(t)/cc-pvdz", molecule=  )
```

QCEngine

ERROR IN 20 MODES VS CBS



- pyvpt2: **AVAILABLE** with pyvp2 (unreleased) & PSI4 & QCEngine & QCFractal.
- qcdb.vpt2: **AVAILABLE** with QCDB dev branch & QCEngine & CFOUR & NWChem (or PSI4 or GAMESS).
- **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.

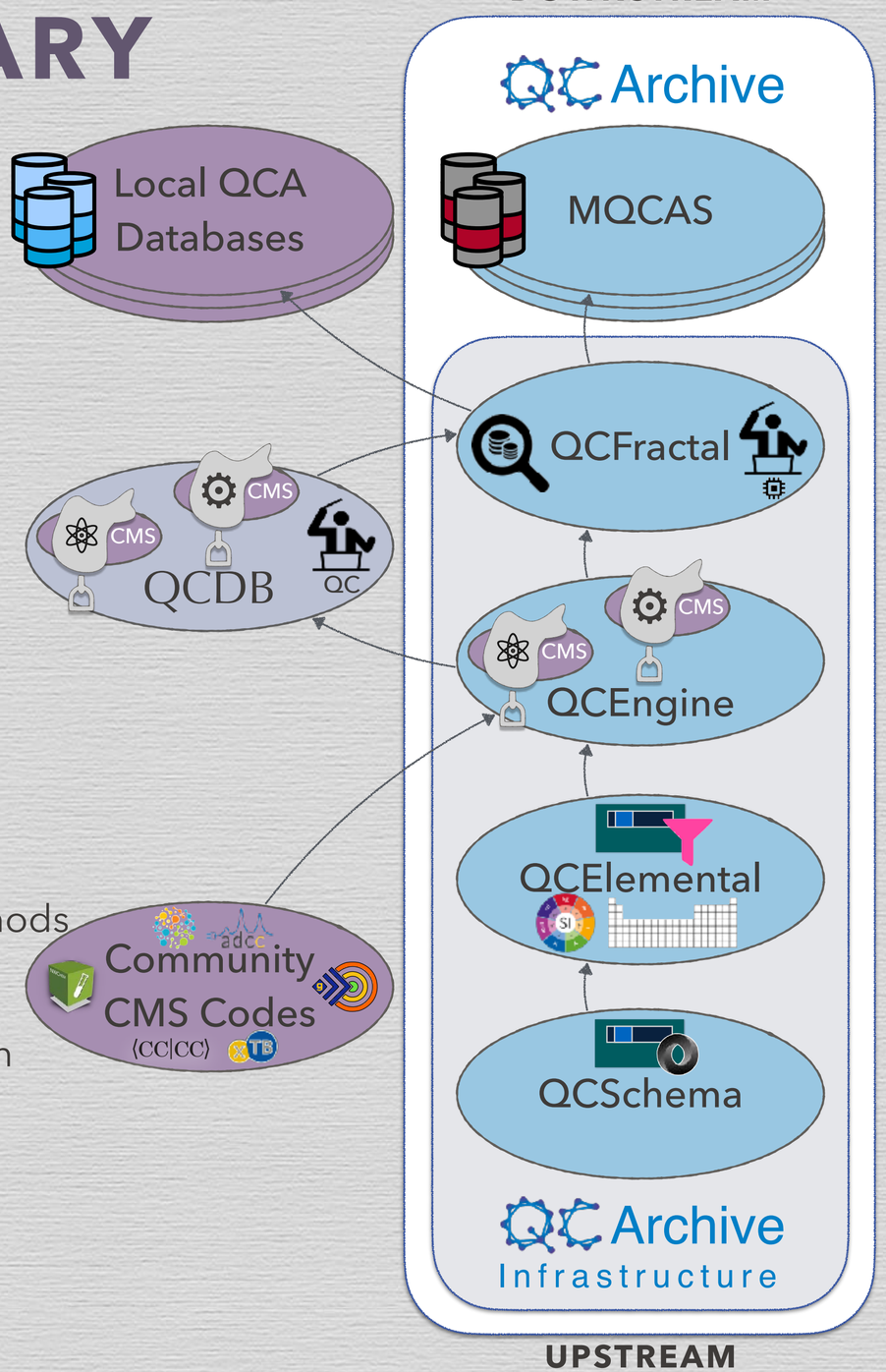
Philip Nelson
GaTech

SUMMARY

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

- DSL or unified input syntax
- multi-QCprogram workflows
- standardization layers

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



- database, universally queryable
- 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

A DFTD3 STORY

hold your interfaces and tests close

A DFTD3 STORY

hold your interfaces and tests close

In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

Stefan Grimme[a)], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

A DFTD3 STORY

hold your interfaces and tests close

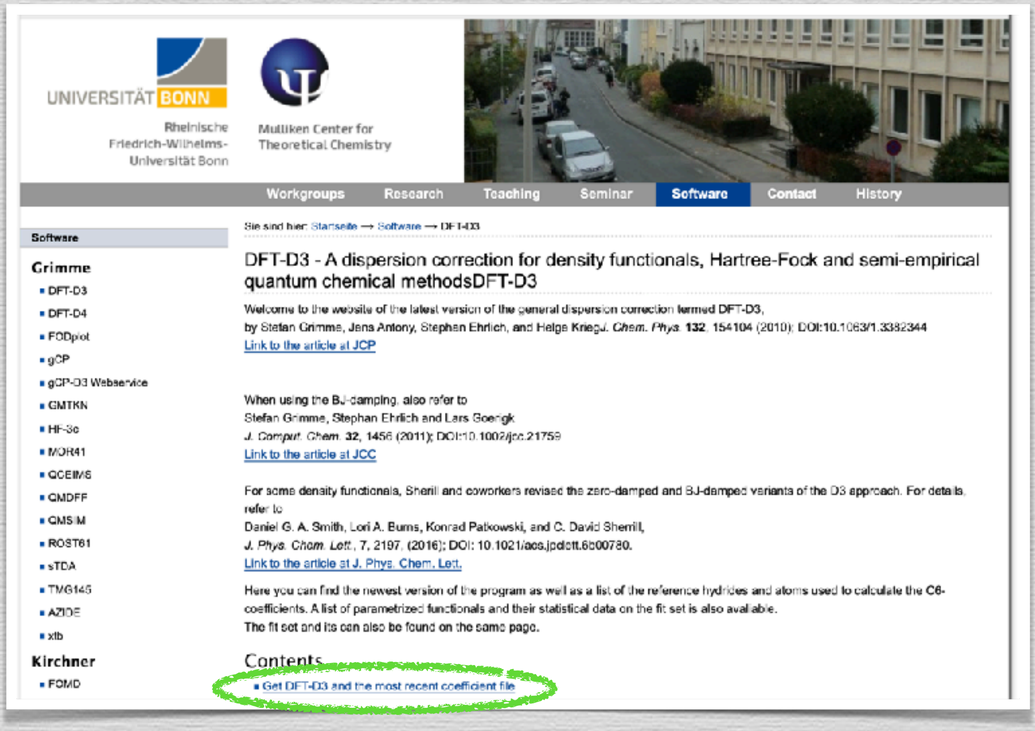
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

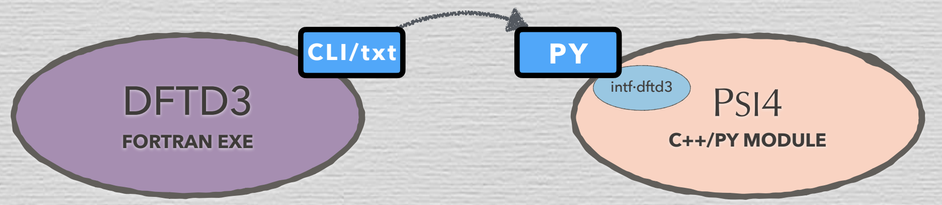
Stefan Grimme[a)], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

Still better, the **CODE** for a Fortran executable



A DFTD3 STORY

hold your interfaces and tests close



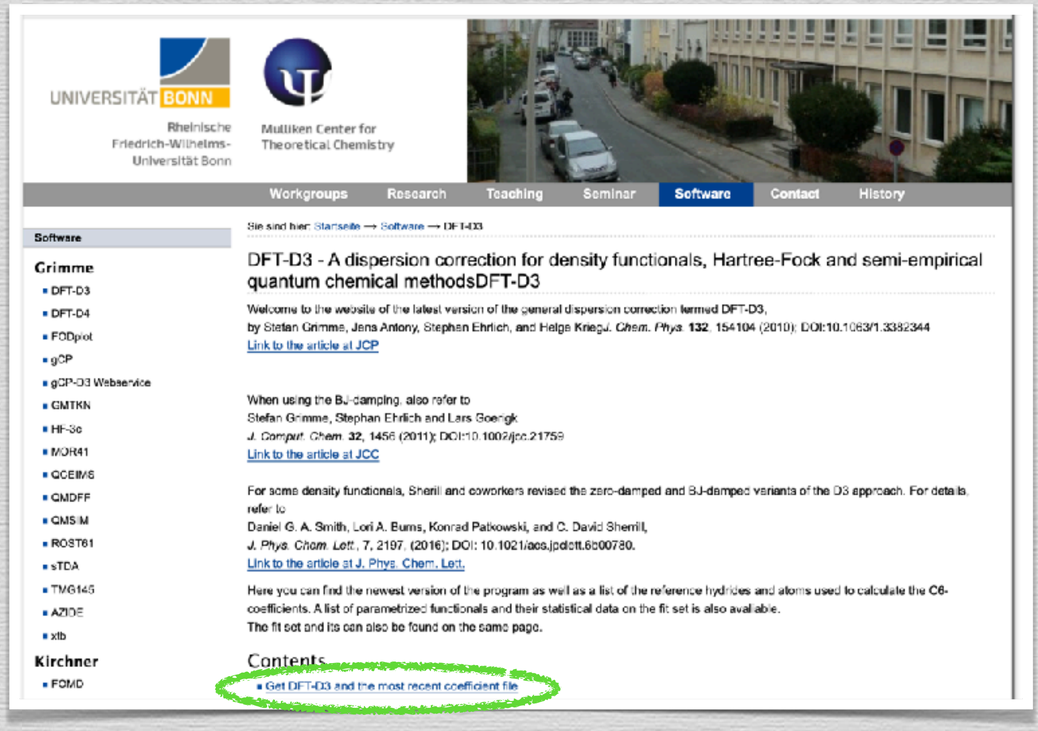
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

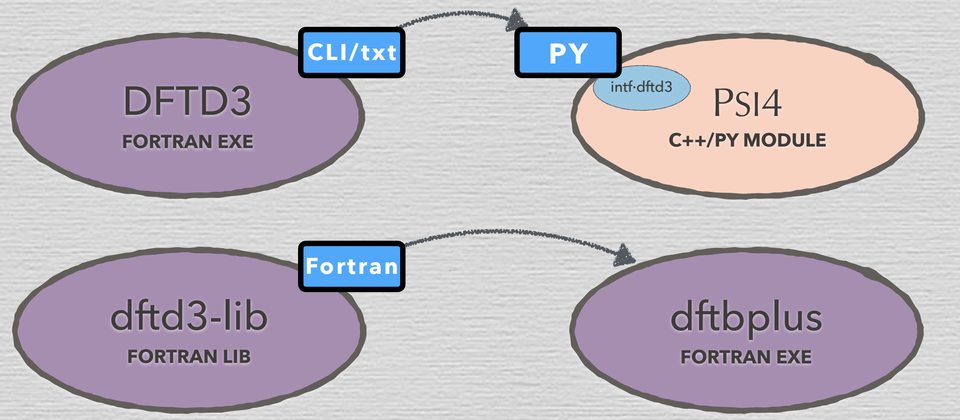
Stefan Grimme[a)], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

Still better, the **CODE** for a Fortran executable



A DFTD3 STORY

hold your interfaces and tests close



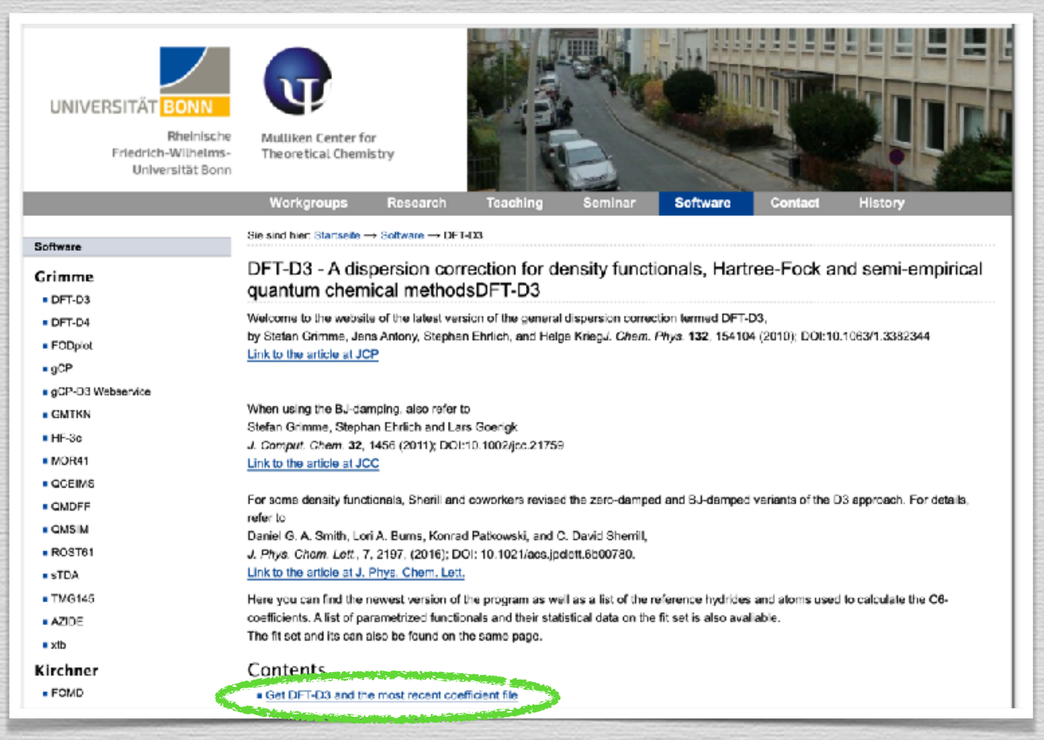
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

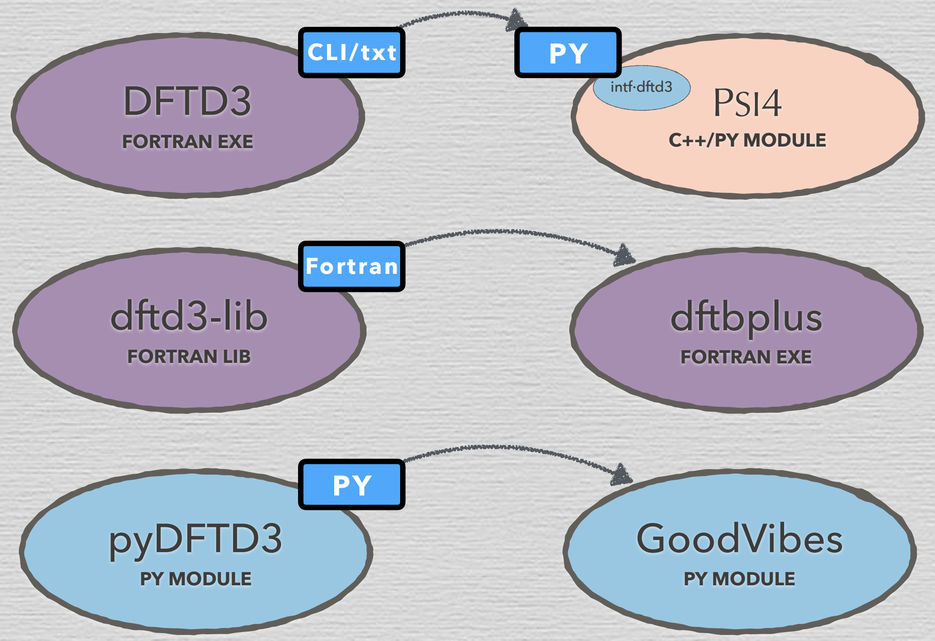
Stefan Grimme[a)], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

Still better, the **CODE** for a Fortran executable



A DFTD3 STORY

hold your interfaces and tests close



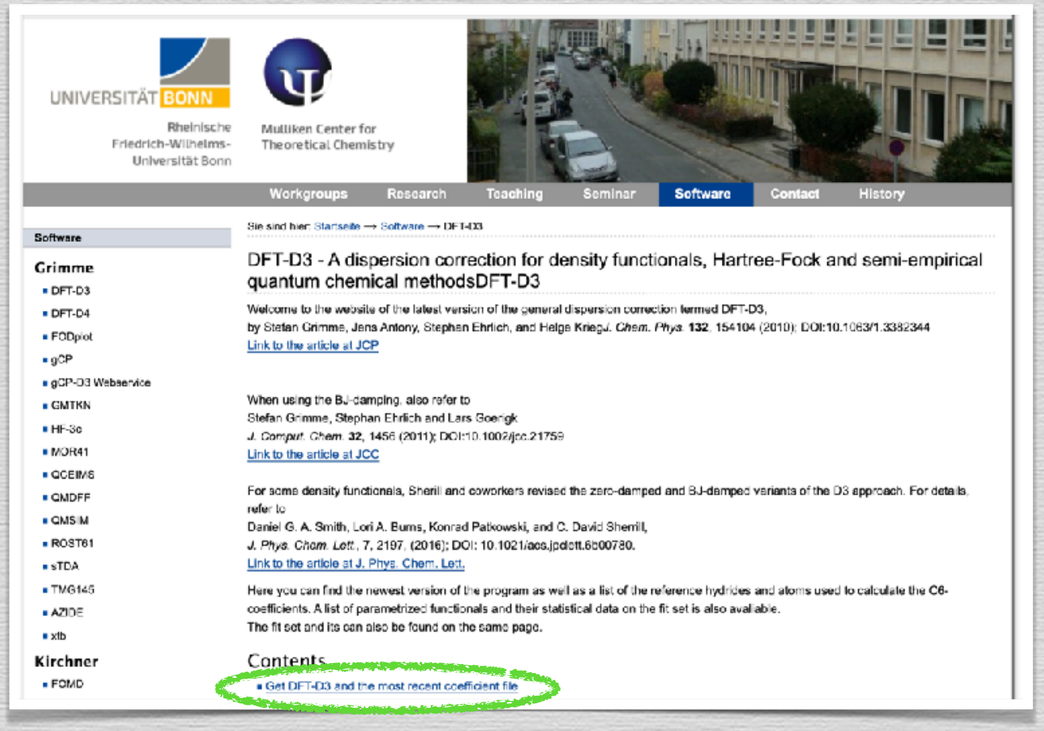
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

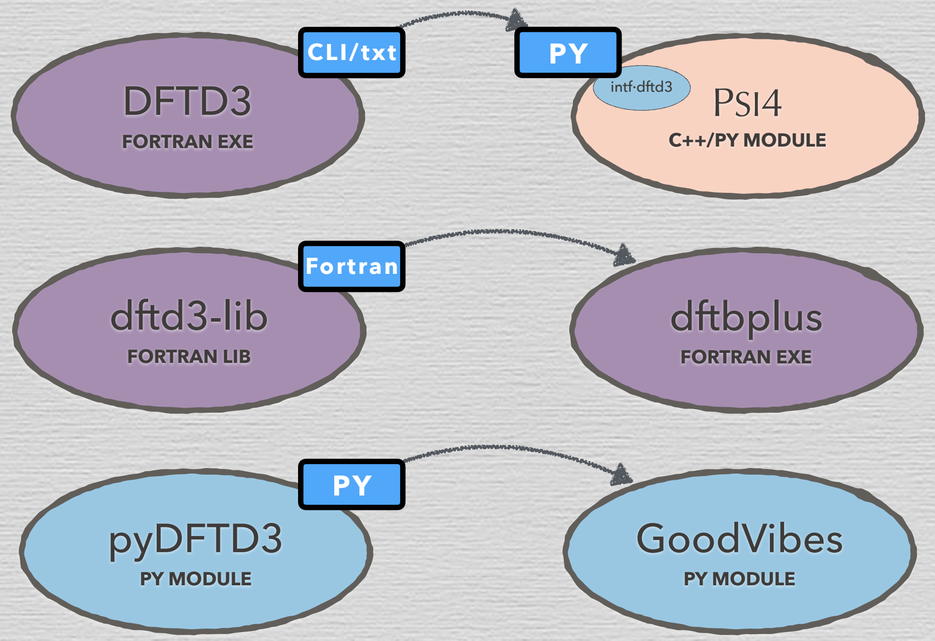
Stefan Grimme[a], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

Still better, the **CODE** for a Fortran executable



A DFTD3 STORY

hold your interfaces and tests close



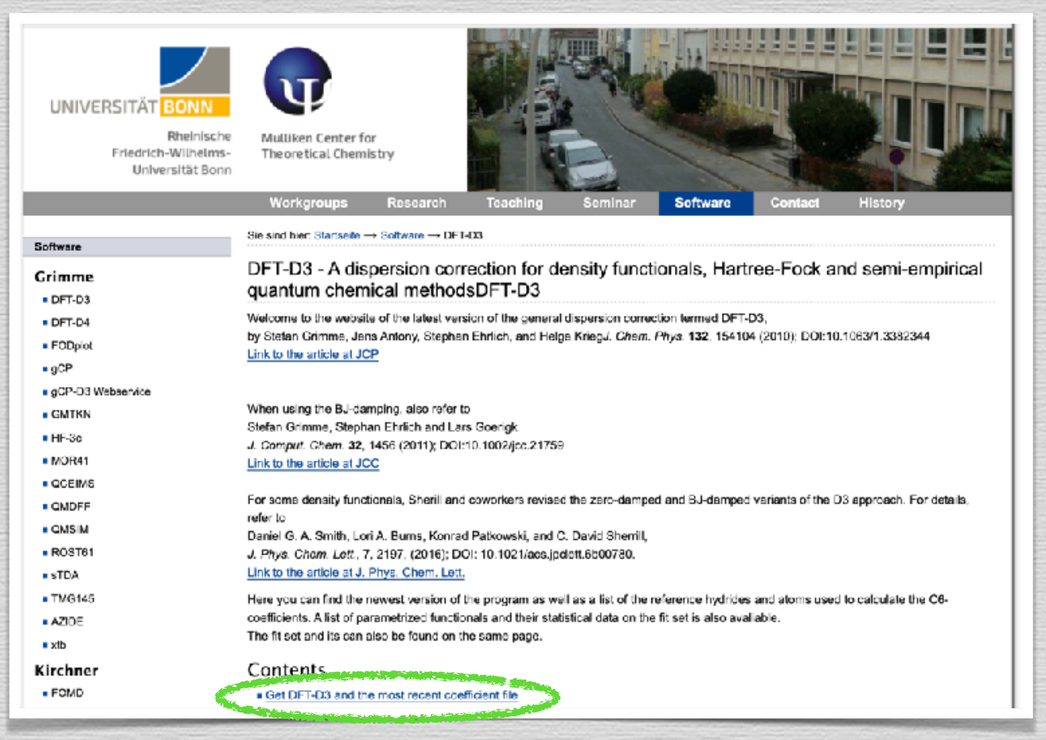
In the beginning was the **METHOD** publication

A consistent and accurate *ab initio* parametrization of density functional dispersion correction (DFT-D) for the 94 elements H-Pu

J. Chem. Phys. **132**, 154104 (2010); https://doi.org/10.1063/1.3382344

Stefan Grimme[a)], Jens Antony, Stephan Ehrlich, *and* Helge Krieg

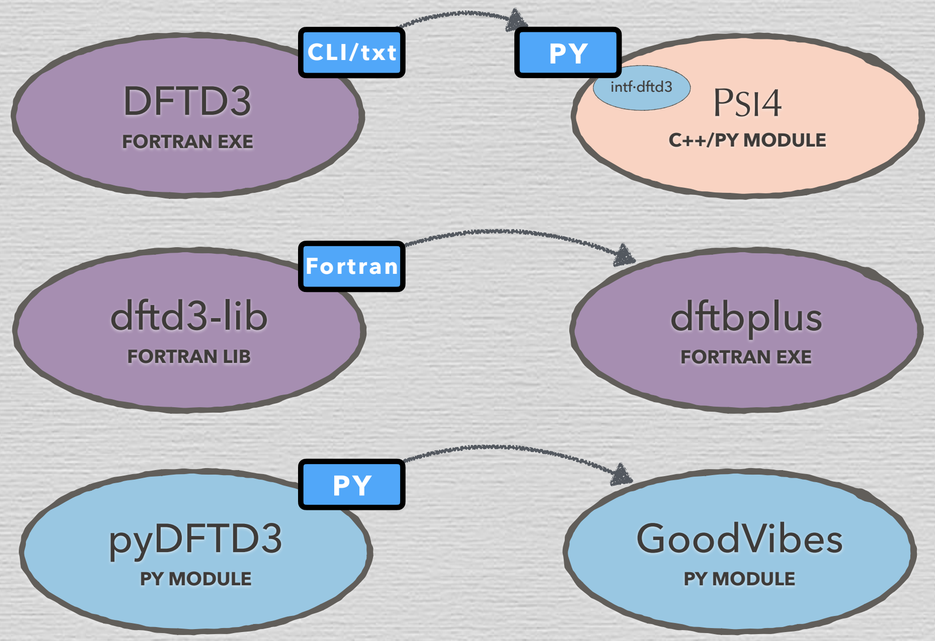
Still better, the **CODE** for a Fortran executable



OPEN-SOURCE projects proliferated until inevitably ...

A DFTD3 STORY

hold your interfaces and tests close



OPEN-SOURCE projects proliferated until inevitably ...

Which dftd3 should one use? #4

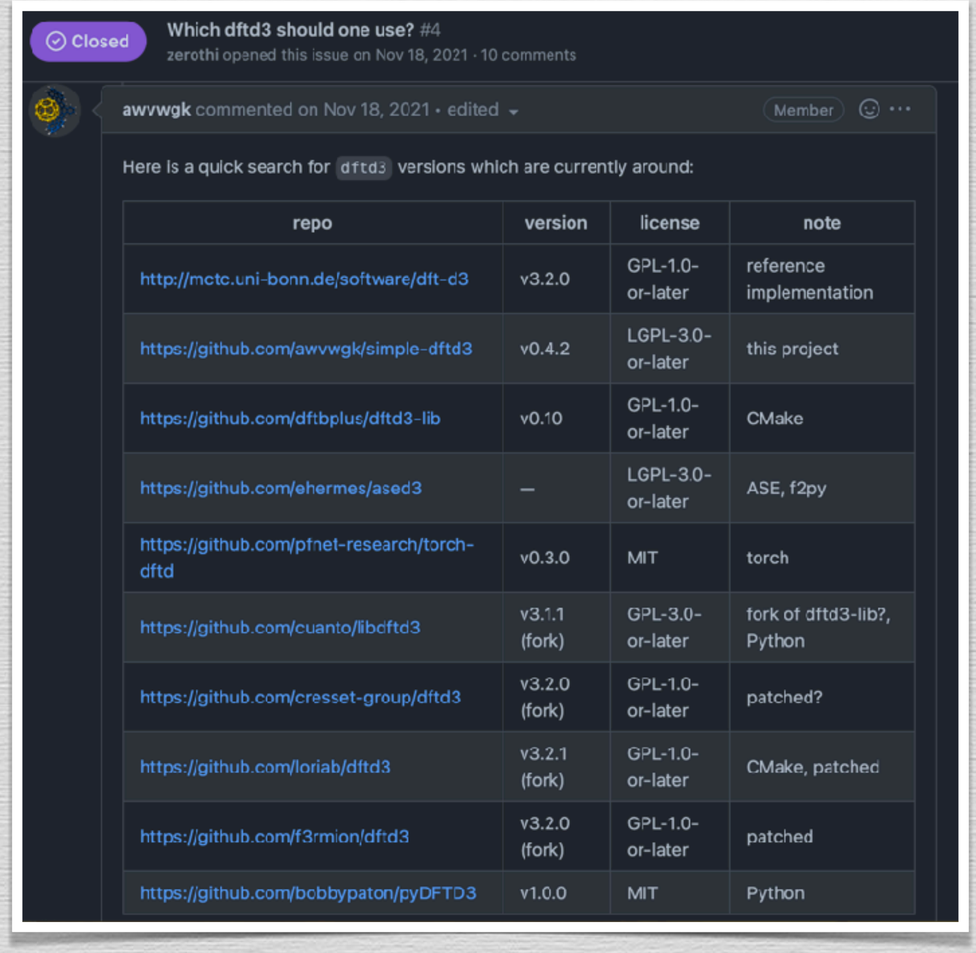
Closed zerothi opened this issue on Nov 18, 2021 · 10 comments

zerothi commented on Nov 18, 2021 • edited

Thanks for the work on this!

We are bit in a dead-lock now, since there exists 3 different libraries which all say do the same thing? :)
So which one should we support :)

... at least ten **TEN IMPLEMENTATIONS** were identified



Closed Which dftd3 should one use? #4

zerothi opened this issue on Nov 18, 2021 · 10 comments

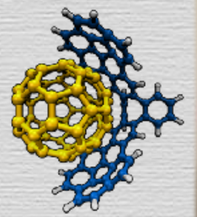
awvwgk commented on Nov 18, 2021 • edited Member

Here is a quick search for `dftd3` versions which are currently around:

repo	version	license	note
http://mctc.uni-bonn.de/software/dft-d3	v3.2.0	GPL-1.0-or-later	reference implementation
https://github.com/awvwgk/simple-dftd3	v0.4.2	LGPL-3.0-or-later	this project
https://github.com/dftbplus/dftd3-lib	v0.10	GPL-1.0-or-later	CMake
https://github.com/ehermes/ased3	—	LGPL-3.0-or-later	ASE, f2py
https://github.com/pfnet-research/torch-dftd	v0.3.0	MIT	torch
https://github.com/cuanto/libdftd3	v3.1.1 (fork)	GPL-3.0-or-later	fork of dftd3-lib?, Python
https://github.com/cresset-group/dftd3	v3.2.0 (fork)	GPL-1.0-or-later	patched?
https://github.com/loriab/dftd3	v3.2.1 (fork)	GPL-1.0-or-later	CMake, patched
https://github.com/f3rmion/dftd3	v3.2.0 (fork)	GPL-1.0-or-later	patched
https://github.com/bobbypaton/pyDFTD3	v1.0.0	MIT	Python

A DFTD3 STORY

hold your interfaces and tests close

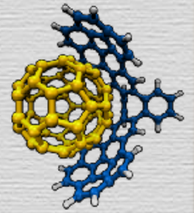


Sebastian Ehlert
Bonn

Holger Kruse
Czech Acad. Sci.

A DFTD3 STORY

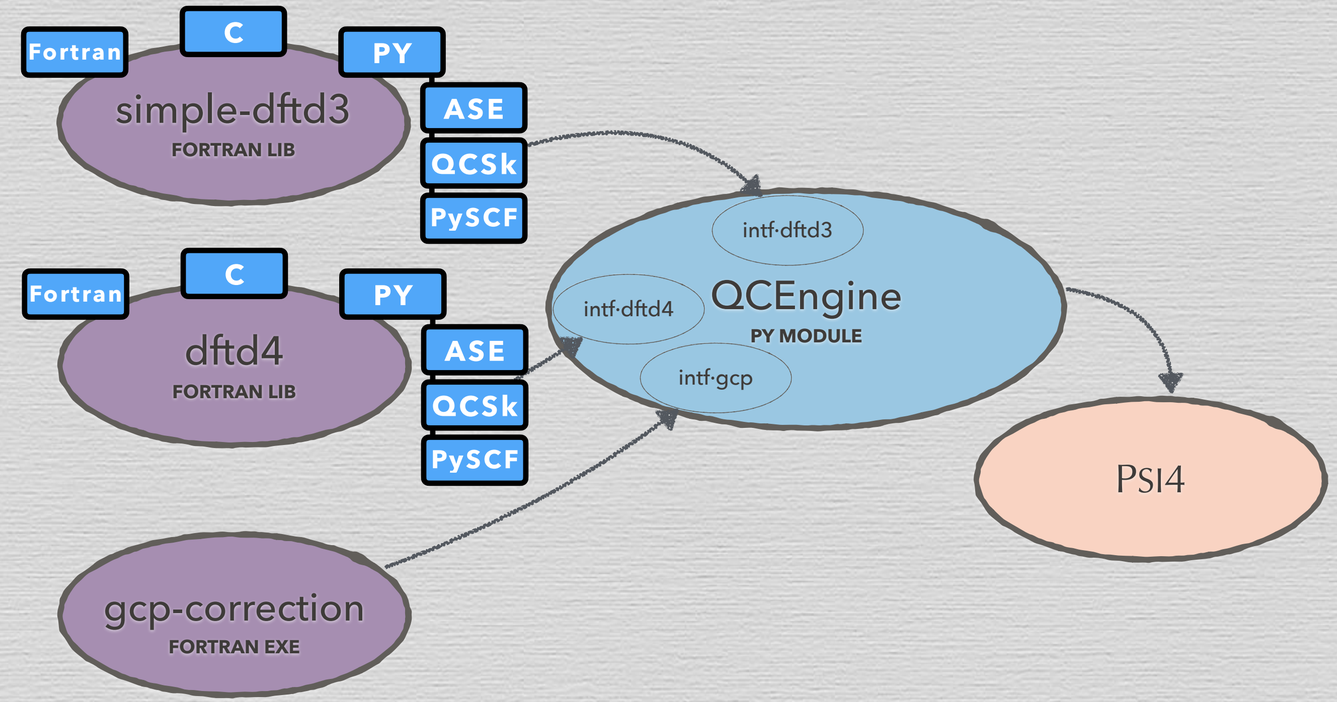
hold your interfaces and tests close



Sebastian Ehlert
Bonn

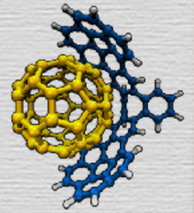
Holger Kruse
Czech Acad. Sci.

RE-IMPLEMENTED upstream with abundant interfaces



A DFTD3 STORY

hold your interfaces and tests close



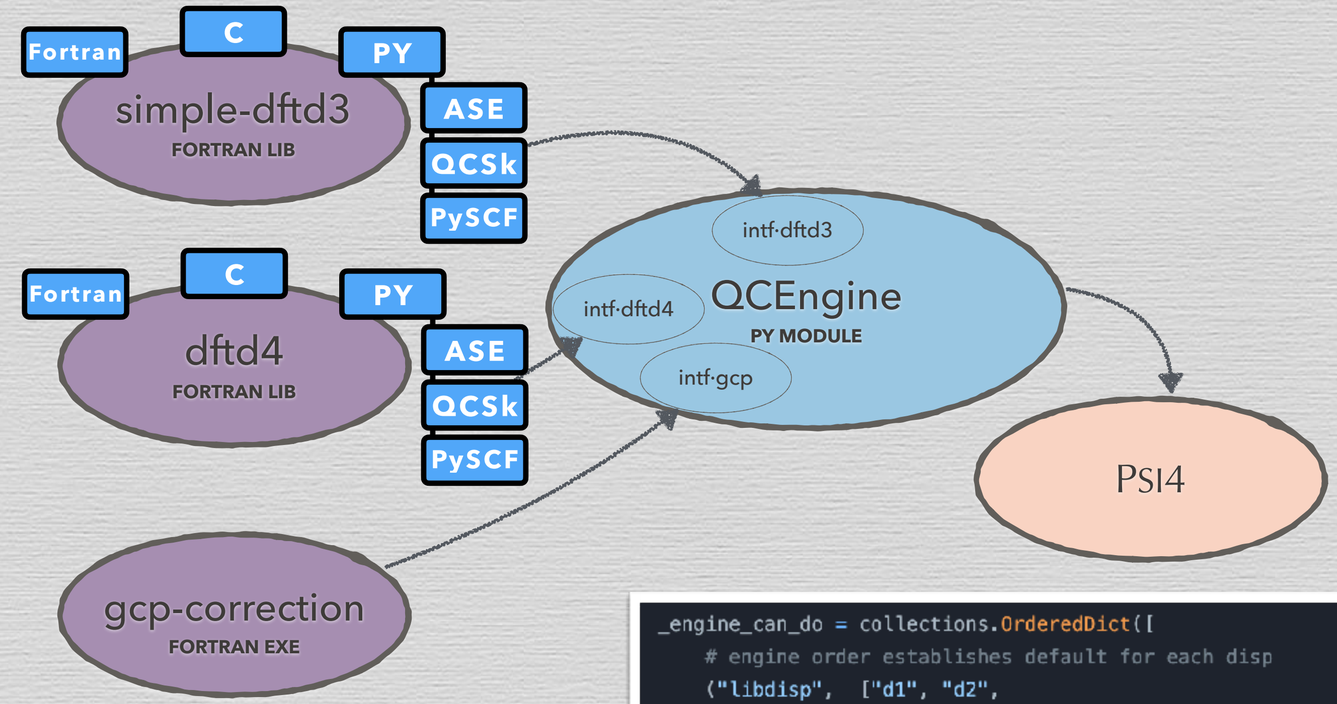
Sebastian Ehlert
Bonn

Holger Kruse
Czech Acad. Sci.

RE-IMPLEMENTED upstream with abundant interfaces

PAYOFF Psi4 no longer maintains and packages forks

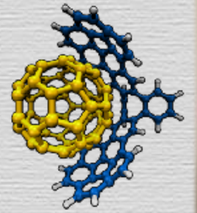
PAYOFF new capabilities easy to add




```
_engine_can_do = collections.OrderedDict([
    # engine order establishes default for each disp
    ("libdisp",  ["d1", "d2",                                                                                                  "chg", "das2009",
    ("s-dftd3",  [           "d3zero2b", "d3bj2b", "d3mzero2b", "d3mbj2b", "d3zeroatm", "d3bjatm", "d3mzeroatm", "d3mbjatm",
    ("dftd3",    [      "d2", "d3zero2b", "d3bj2b", "d3mzero2b", "d3mbj2b",
    ("nl",       [                                                                                                                "nl",
    ("mp2d",     [                                                                                                                "dmp2",
    ("dftd4",    [                                                                                                  "d4bjeeqatm",
    ("mctc-gcp", [                                                                                                                "3c",
    ("gcp",      [                                                                                                                "3c",
]) # yapf: disable
```

A DFTD3 STORY

hold your interfaces and tests close



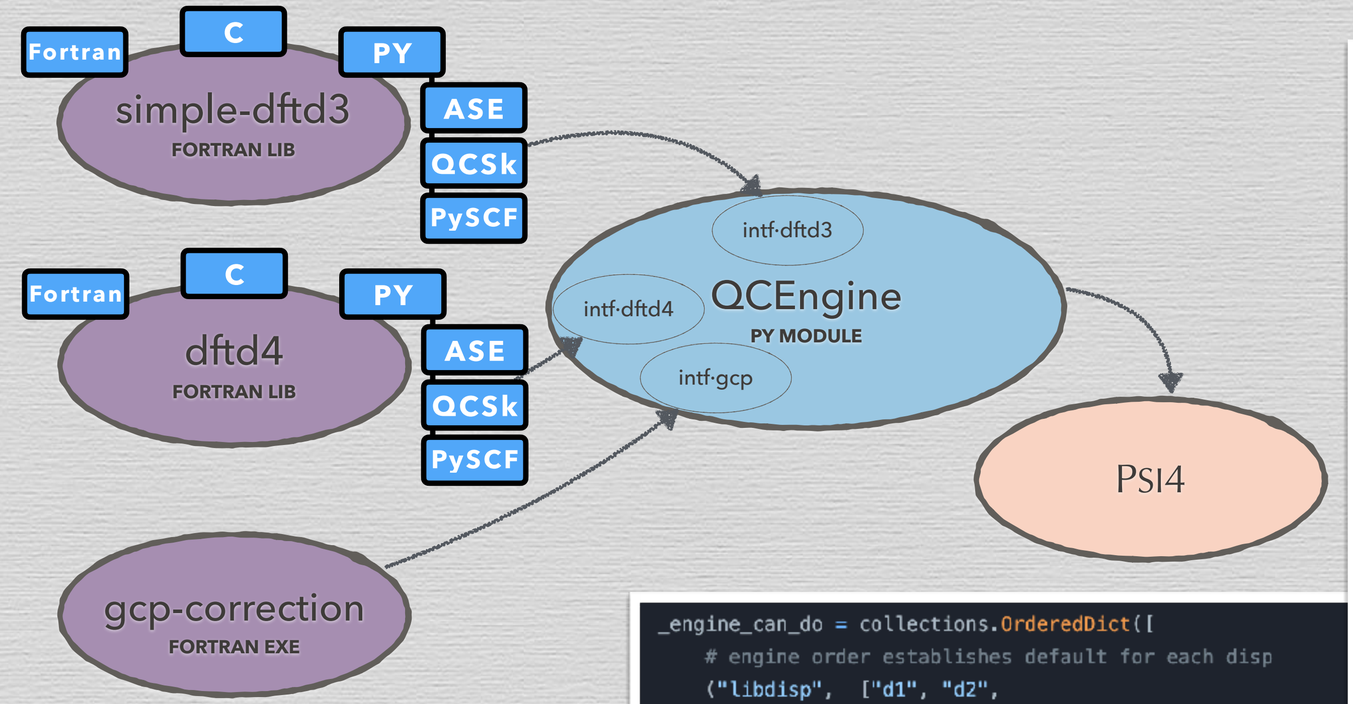
Sebastian Ehlert
Bonn

Holger Kruse
Czech Acad. Sci.

RE-IMPLEMENTED upstream with abundant interfaces

PAYOFF Psi4 no longer maintains and packages forks

PAYOFF new capabilities easy to add



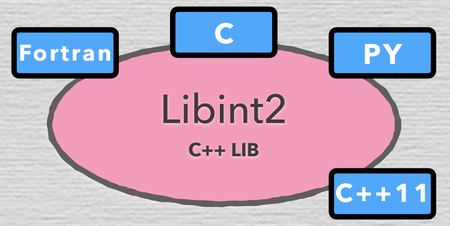

```
249 + funcs.append({
250 +     "name": "R2SCAN3C",
251 +     "alias": ["R2SCAN-3C"],
252 +     "x_functionals": {
253 +         "MGGA_X_R2SCAN": {}
254 +     },
255 +     "c_functionals": {
256 +         "MGGA_C_R2SCAN": {}
257 +     },
258 +     "description": '     r2SCAN Meta-GGA based 3C composite method with a TZ basis set, gCP and D4\n',
259 +     "citation": '     S. Grimme,  A. Hansen,  S. Ehlert, J.-M. Mewes J. Chem. Phys. 154, 064103, 2021\n',
260 +     "doi": "10.1063/5.0040021",
261 +     "dispersion": {
262 +         "type": "d4bjeeqatm",
263 +         "params": {
264 +             'a1': 0.420,
265 +             'a2': 5.650,
266 +             's6': 1.000,
267 +             's8': 0.000,
268 +             's9': 2.000,
269 +             'ga': 2.000,
270 +             'gc': 1.000,
271 +         },
272 +     },
273 + })
```

Adds r2SCAN hybrids; r2SCAN-3c and B97-3c #2842

```
_engine_can_do = collections.OrderedDict([
    # engine order establishes default for each disp
    ("libdisp",   ["d1", "d2",
    ("s-dftd3",   [           "d3zero2b", "d3bj2b", "d3mze
    ("dftd3",     [     "d2", "d3zero2b", "d3bj2b", "d3mzero2b", "d3mbj2b",
    ("nl",        [                                                                  "nl",
    ("mp2d",      [                                                                  "dmp2",
    ("dftd4",     [                                                        "d4bjeeqatm",
    ("mctc-gcp", [                                                                   "3c",
    ("gcp",       [                                                                  "3c",
]) # yapf: disable
```

A LIBINT STORY

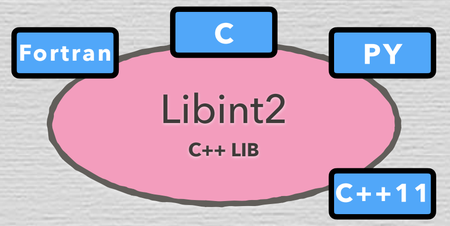
play nicely in the *runtime* software stack



- **DEVELOPED** by E. Valeev for 20y
- **FULL** service integrals library
- **INTERFACES** in multiple languages/APIs
- **HIGHLY** configurable by integral classes, derivatives, orderings, & angular momentum levels
- **LARGE** size at production configurations, >GB generated source or built library

A LIBINT STORY

play nicely in the *runtime* software stack



- **DEVELOPED** by E. Valeev for 20y
- **FULL** service integrals library
- **INTERFACES** in multiple languages/APIs
- **HIGHLY** configurable by integral classes, derivatives, orderings, & angular momentum levels
- **LARGE** size at production configurations, >GB generated source or built library

In **2020** Psi4 started updating from 15yo Libint1 to maintained Libint2

Open **Libint2 2022 Strategy** #2442

loriab opened this issue on Feb 14 · 2 comments

step	status	libint ver & branch	Psi4
1[3]	longstanding L1	L1 evaleev:5c89451	v1.3
2[4]	TEI L2	loriab:l2cmake evaleev/libint#148	20Nov20, after #1721, v1.4, 1.5
3[5]	OEI L2	ditto step 2	11Mar22, after #2388
B[6]	upstream L2 cmake	loriab:new-cmake-harness-lab-rb1 evaleev/libint#233	23Mar22, after #2413, v1.6
C[2]	McMurchie Davidson	any	31Mar22, after #2414, v1.6
A[7]	standardize ordering	ditto step B	#2537

A LIBINT STORY

play nicely in the *runtime* software stack

both projects subject to a bit of tunnelvision:

Libint assumed targeted toward **SINGLE** QC consumer

Psi4 vulnerable to **SEGFAULT** if meagerly configured

Psi4 vulnerable to **WRONG RESULTS** if configured w/std orderings

Portable libint #190

Open · susilehtola opened this issue on Sep 3, 2020 · 26 comments · May be fixed by #205

susilehtola commented on Sep 3, 2020 — Contributor

Hi,

it looks like libint2 can be compiled in `Nsph x Ncart x Nshell_set = 2 x 5 x 2 = 20` different possible ways. Even though only 5 of these are currently used by major codes, this is a nightmare for distribution packaging, since there's no way we can ship 20 mutually incompatible copies of a library; I'm horrified about shipping even two since it takes forever to compile libint on the Fedora build system.

Would it be possible to modify libint so that the orderings for the cartesians, solid harmonics, and shell sets are set by some function, and libint does all the necessary shuffles behind the scenes? This would be a minimal solution to the problem, since one could still set the default orderings at configure time, thus avoiding any extra shuffles if the compiled ordering matches the wanted one.

020 · edited — Collaborator

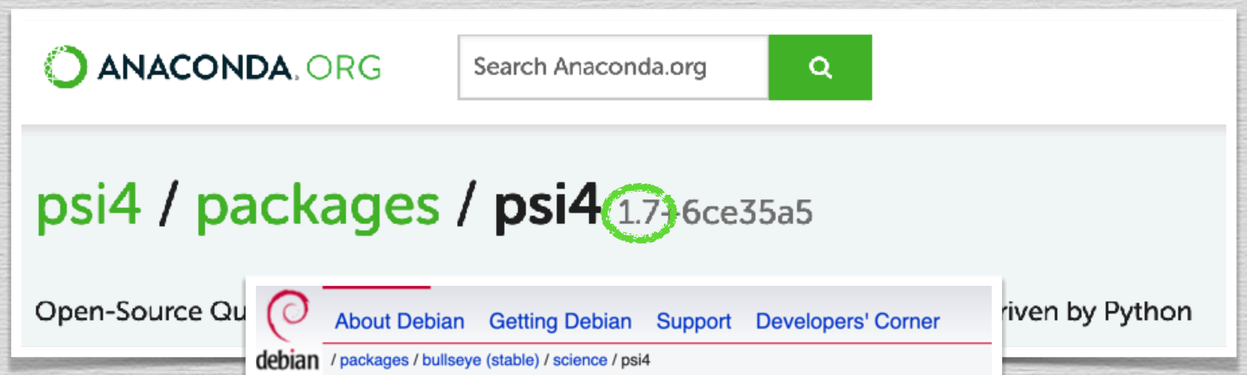
psi is non-standard for solid harmonic and probably precipitated this thread. (below is harvested from libint wiki notes.) small mercy is that psi differs in the one dimension that can be chosen at library build time, not generator build time.

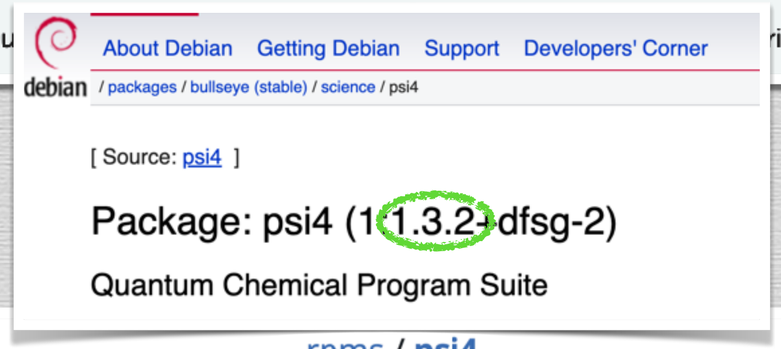
```
#                  sh         cart       shell_set  used_by
#                  --------   --------   ---------  -------
#   sss - search for standard + standard + standard = mpqc4, cp2k
#   sso - search for                     + orca
#   sis - search for          + intv3    + standard
#   sio - search for                     + orca
#   sgs - search for          + gamess   + standard = gamess
#   sgo - search for                     + orca
#   sos - search for          + orca     + standard
#   soo - search for                     + orca     = orca
#   sbs - search for          + bagel    + standard = bagel
#   sbo - search for                     + orca
#   gss - search for gaussian + standard + standard = psi4
#   gso - search for                     + orca
#   gis - search for          + intv3    + standard
#   gio - search for                     + orca
#   ggs - search for          + gamess   + standard
#   ggo - search for                     + orca
#   gos - search for          + orca     + standard
#   goo - search for                     + orca
#   gbs - search for          + bagel    + standard
#   gbo - search for                     + orca
```

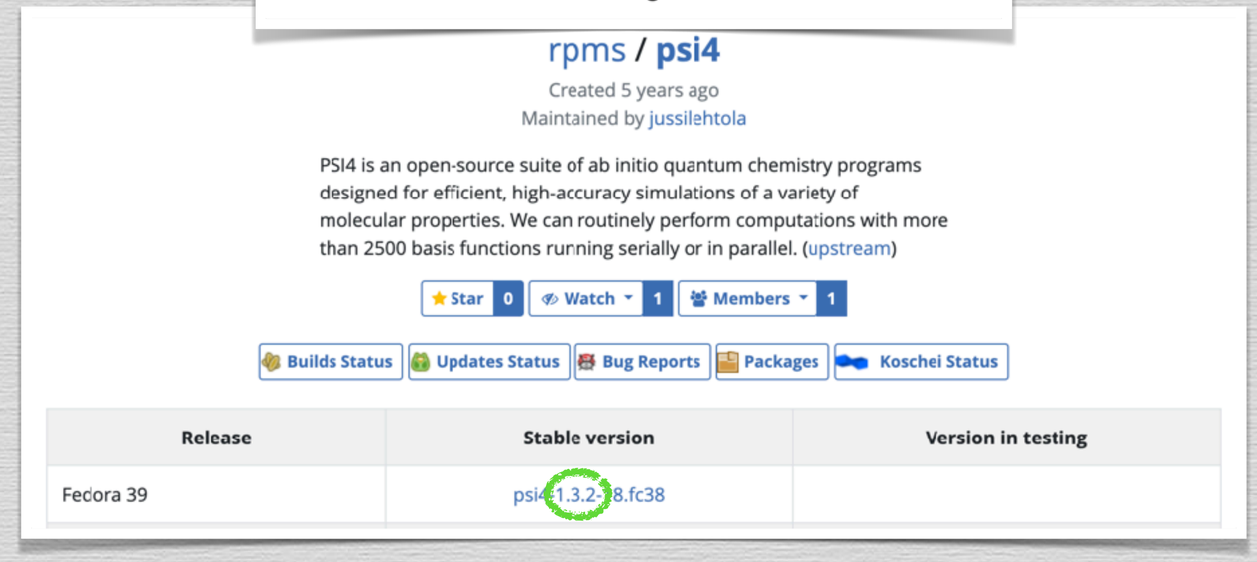
A LIBINT STORY

play nicely in the *runtime* software stack

incompatibilities frustrate **PACKAGING**, so stuck c. 2019







so Psi4 has contributed back upstream

expanded to Windows **ARCH** & CMake **BUILDSYS**

added **LINK-TIME** & **RUN-TIME** configuration checking

enabled **RUN-TIME** ordering selection for OSS consumers

PAYOFF delete much home-grown code

PAYOFF expand capabilities with new integrals types

PAYOFF return to upstream so new Psi4 releases packageable

PAYOFF conda-forge packaging

CODE DEFENSE

burden the bots to spare the people

CODE DEFENSE

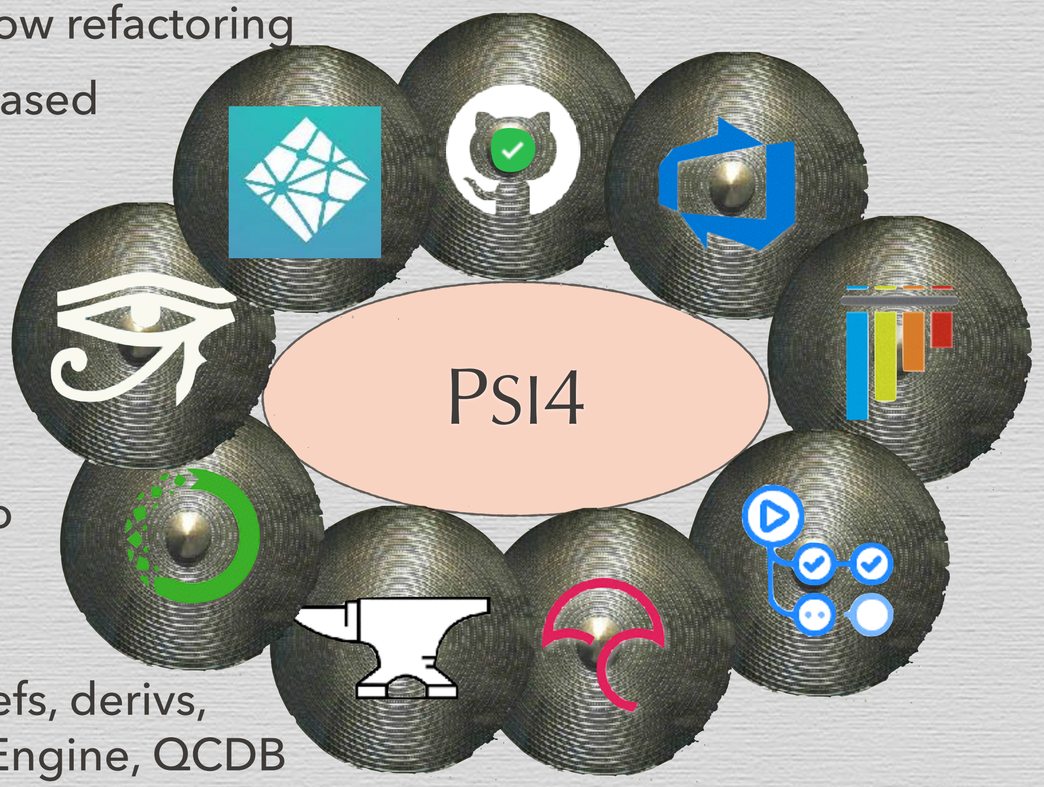
burden the bots to spare the people

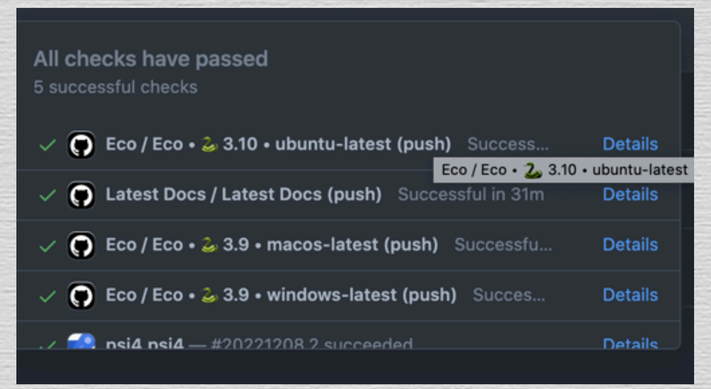
CODE DEFENSE

burden the bots to spare the people

TESTING

- **VITAL** to consolidate gains & allow refactoring
- **PYTEST** runs entire suite, infile-based (CTest) & API-based (pytest)
- **7613** tests in entire suite
- **2296** tests run on each PR
- **386** tests integrating with external software
- **ADDONS** detected at runtime, so safe to test full ecosystem
- **5132** tests in "standard suite": systematic coverage of mtds, refs, derivs, algorithms shared by Psi4, QCEngine, QCDB



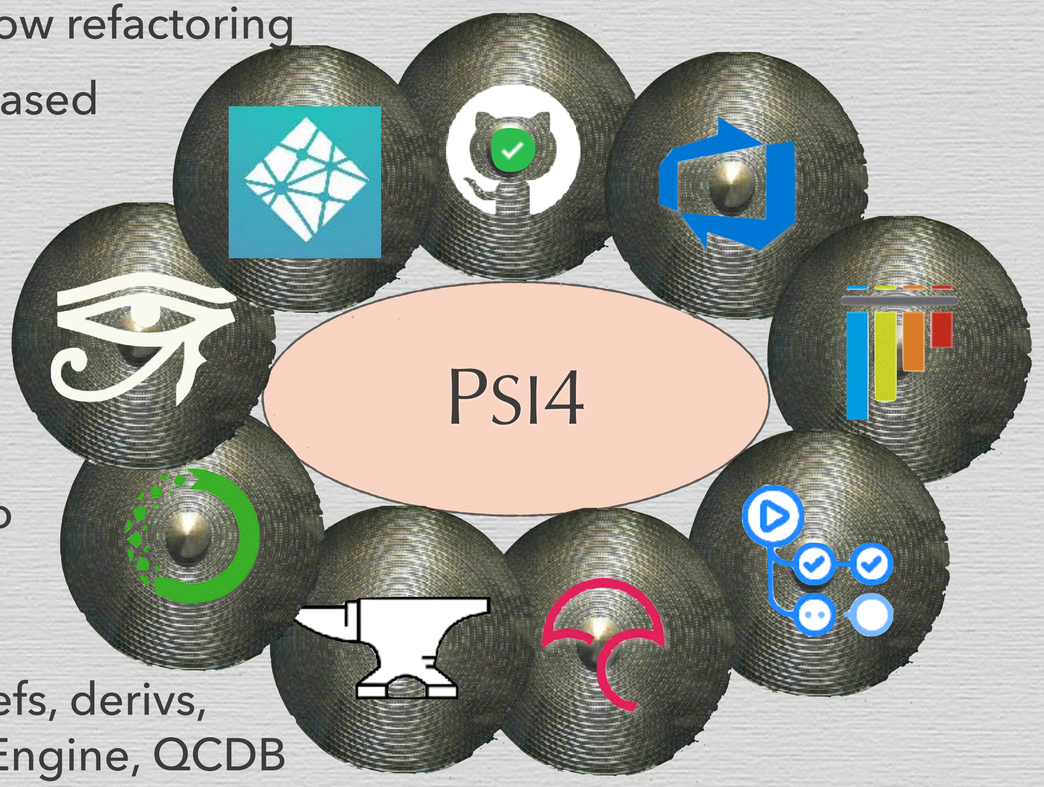


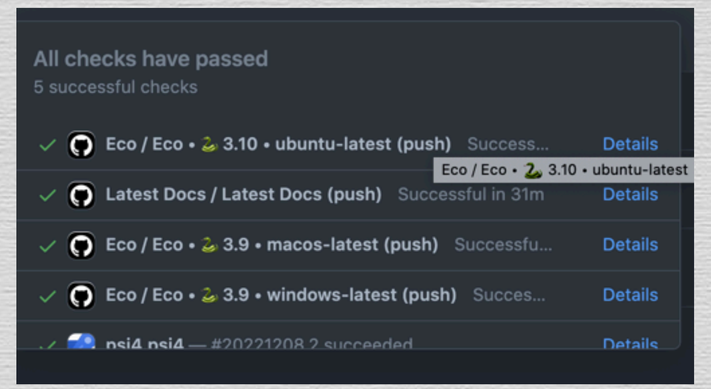
CODE DEFENSE

burden the bots to spare the people

TESTING

- **VITAL** to consolidate gains & allow refactoring
- **PYTEST** runs entire suite, infile-based (CTest) & API-based (pytest)
- **7613** tests in entire suite
- **2296** tests run on each PR
- **386** tests integrating with external software
- **ADDONS** detected at runtime, so safe to test full ecosystem
- **5132** tests in "standard suite": systematic coverage of mtds, refs, derivs, algorithms shared by Psi4, QCEngine, QCDB





CODE DEFENSE

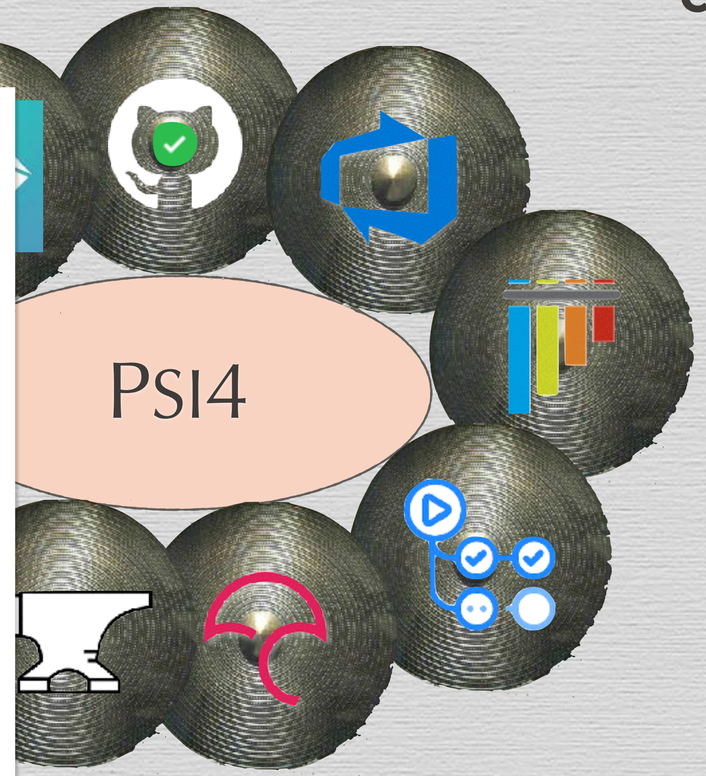
burden the bots to spare the people

AUTO-GEN QCDB CAPABILITIES

Capabilities of QCDB, including details of overlapping modules. "✓" runs analytically. "!" produces nonstandard results.

□	qc_module ↓	QCDB Capabilities																	
□	scf_reference →	Restricted (RHF)						Unrestricted (UHF)						Restricted Open (ROHF)					
name ↓ →	□	E		G		H		E		G		H		E		G		H	
□	freeze_core[1] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
mp2	CFOUR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	!	!
	GAMESS-DDI	✓	✓	✓	✓			✓	✓	✓	✓								
	GAMESS-IMS	✓	✓	✓	✓														
	GAMESS-SERIAL	✓	✓	✓	✓			✓	✓	✓	✓			✓	✓	!	!		
	NWChem-DIRECTMP2	✓	✓																
	NWChem-MP2GRAD	✓	✓	✓	✓			✓	✓	✓	✓								
	NWChem-TCE	✓	✓					✓	✓					!	!	!	!		
	Psi4-FNOCC	✓	✓																
	Psi4-OCC	✓	✓	✓				✓	✓	✓				✓	✓				
ccsd(t)	CFOUR-ECC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	CFOUR-NCC	✓	✓	✓	!														
	CFOUR-VCC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	GAMESS	✓	✓																
	NWChem-CC	✓	✓																
	NWChem-TCE	✓	✓					✓	✓					!	!	!			
	Psi4-CCENERGY	✓	✓					✓	✓					✓	✓				
	Psi4-FNOCC	✓	✓																
	Psi4-MRCC	✓	✓					✓	✓					✓	✓				

[1] Active orbital values to the right: all-electron `A` and frozen-core `F`.



CONTINUOUS INTEGRATION

- **AZURE** builds Linux & Windows, tests internal
- **GHA** builds Linux, Mac, & Windows, tests ecosystem
- **DOCS** built for each PR checking format & some coverage
- **CAPABILITIES** docs are generated from stdsuite for Psi4 & QCDB
- **WINDOWS** conda package built continuously from master by Azure; Linux soon
- **WEBSITE** updated nightly with master docs & download stats

CODE DEFENSE

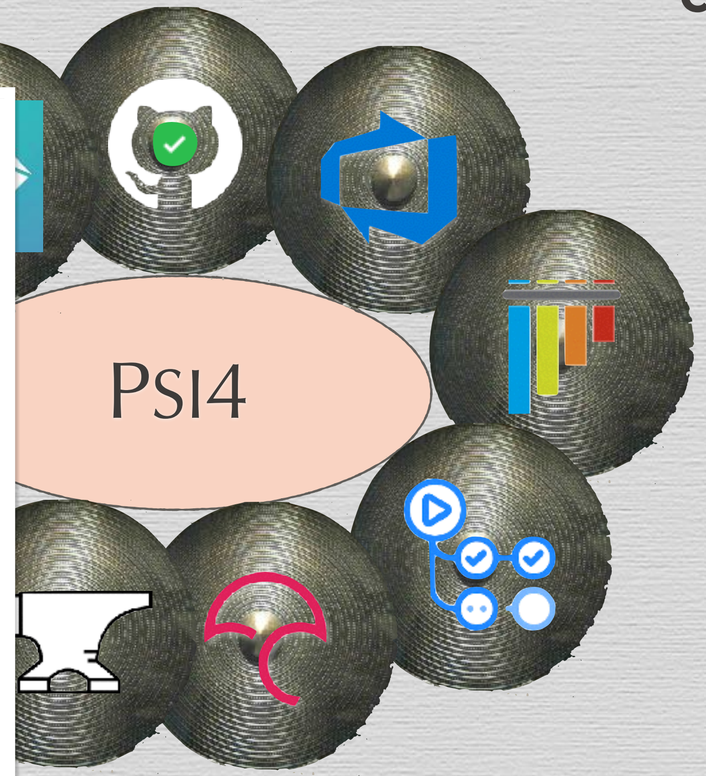
burden the bots to spare the people

AUTO-GEN QCDB CAPABILITIES

Capabilities of QCDB, including details of overlapping modules. "✓" runs analytically. "!" produces nonstandard results.

□	qc_module ↓	QCDB Capabilities																	
□	scf_reference →	Restricted (RHF)						Unrestricted (UHF)						Restricted Open (ROHF)					
name ↓ →	□	E		G		H		E		G		H		E		G		H	
□	freeze_core[1] →	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F	A	F
mp2	CFOUR	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	!	!
	GAMESS-DDI	✓	✓	✓	✓			✓	✓	✓	✓								
	GAMESS-IMS	✓	✓	✓	✓														
	GAMESS-SERIAL	✓	✓	✓	✓			✓	✓	✓	✓			✓	✓	!	!		
	NWChem-DIRECTMP2	✓	✓																
	NWChem-MP2GRAD	✓	✓	✓	✓			✓	✓	✓	✓								
	NWChem-TCE	✓	✓					✓	✓					!	!	!	!		
	Psi4-FNOCC	✓	✓																
	Psi4-OCC	✓	✓	✓				✓	✓	✓				✓	✓				
ccsd(t)	CFOUR-ECC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	CFOUR-NCC	✓	✓	✓	!														
	CFOUR-VCC	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓		
	GAMESS	✓	✓																
	NWChem-CC	✓	✓																
	NWChem-TCE	✓	✓					✓	✓					!	!	!			
	Psi4-CCENERGY	✓	✓					✓	✓					✓	✓				
	Psi4-FNOCC	✓	✓																
	Psi4-MRCC	✓	✓					✓	✓					✓	✓				

[1] Active orbital values to the right: all-electron `A` and frozen-core `F`.

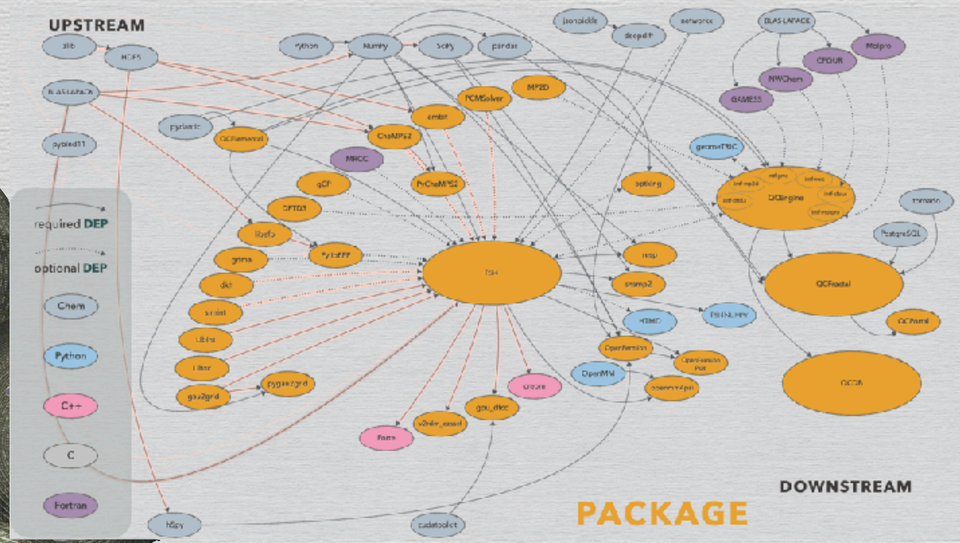


CONTINUOUS INTEGRATION

- **AZURE** builds Linux & Windows, tests internal
- **GHA** builds Linux, Mac, & Windows, tests ecosystem
- **DOCS** built for each PR checking format & some coverage
- **CAPABILITIES** docs are generated from stdsuite for Psi4 & QCDB
- **WINDOWS** conda package built continuously from master by Azure; Linux soon
- **WEBSITE** updated nightly with master docs & download stats

CODE DEFENSE

burden the bots to spare the people



PACKAGING

- **CONDA** is a cross-platform, cross-language binary package manager supporting community packaging
- **STARTING** 2015 for Linux, `conda install psi4 -c psi4` has been recc.
- **PRESENTLY** Linux, Mac, & Windows Psi4 ecosystems available from conda
- **THIRTY** ecosystem projects are or have been packaged by Psi4
- **CONDA-FORGE** is increasingly the home for downstream & dep. projects, so for robust env. solving, transitioning to c-f

BINARY INSTALLER

https://psicode.org/installs/latest

BINARY INSTALLER

https://psicode.org/installs/latest

Get Started with Psi4

Select Preferences

LINUX | MACOS | WINDOWS WSL | WINDOWS

INSTALLER | CONDA | SOURCE

3.8 | 3.9 | 3.10

PREV RELEASE, v1.6.1 | STABLE RELEASE, v1.7.0 | NIGHTLY BUILD

64-bit; glibc 2.17 or higher. 64-bit; OS X 10.10 or higher. 64-bit; select between Windows Subsystem for Linux and native Windows (new with v1.4!). Download standalone command-line installer. Use conda package manager. Build from source using tools and dependencies from conda. Python included, so choose the version you *want*, regardless of any you *have*.

Run this command

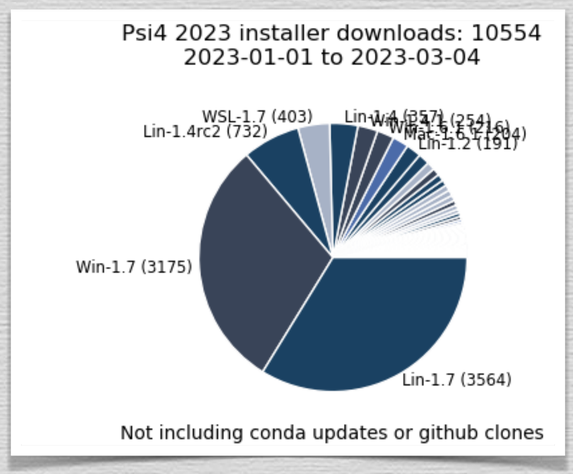
DOWNLOAD PSI4CONDA INSTALLER (ALT. CURL CMD BELOW)

```
# download via button above  -OR-  following line
>_ curl "http://vergil.chemistry.gatech.edu/psicode-download/Psi4conda-1.7-py310-Linux-x86_64.sh" -
>_ bash Psi4conda-1.7-py310-Linux-x86_64.sh -b -p $HOME/psi4conda
>_ echo '. $HOME/psi4conda/etc/profile.d/conda.sh\nconda activate' >> ~/.bashrc
>_ echo "source $HOME/psi4conda/etc/profile.d/conda.csh\nconda activate" >> ~/.tcshrc
# log out, log back in so conda and psi4 in path
>_ psi4 --test
```

PY-WARY USERS: CONDA INSTALLER

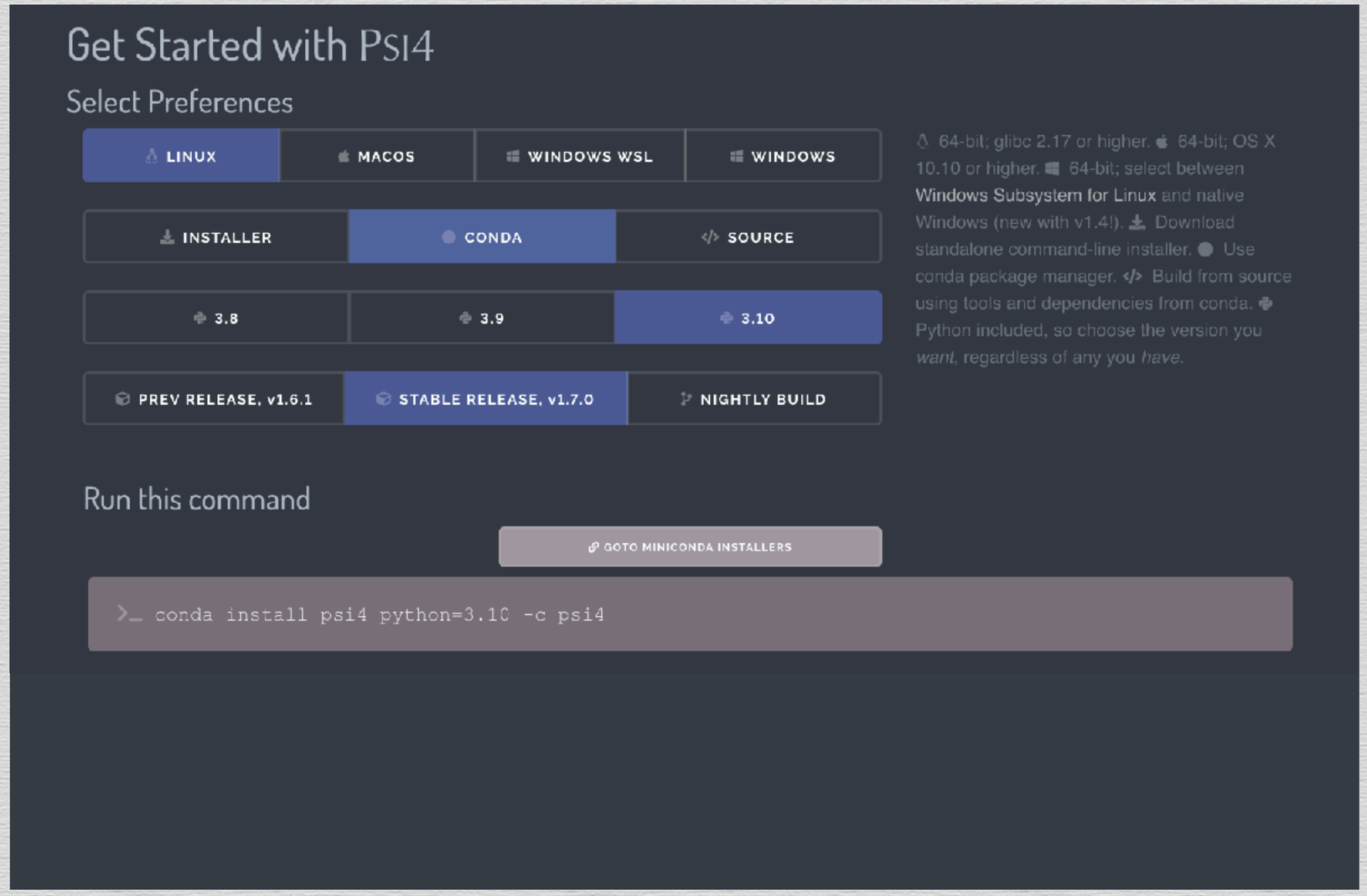
Psi4 · dependencies · add-ons

573K



BINARY INSTALLER

https://psicode.org/installs/latest



PY-WARY USERS: CONDA INSTALLER

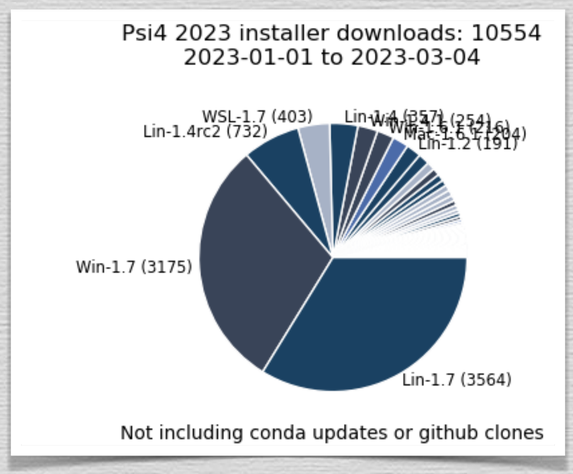
573K

PSI4 · dependencies · add-ons

PY-FRIENDLY USERS: CONDA PACKAGE

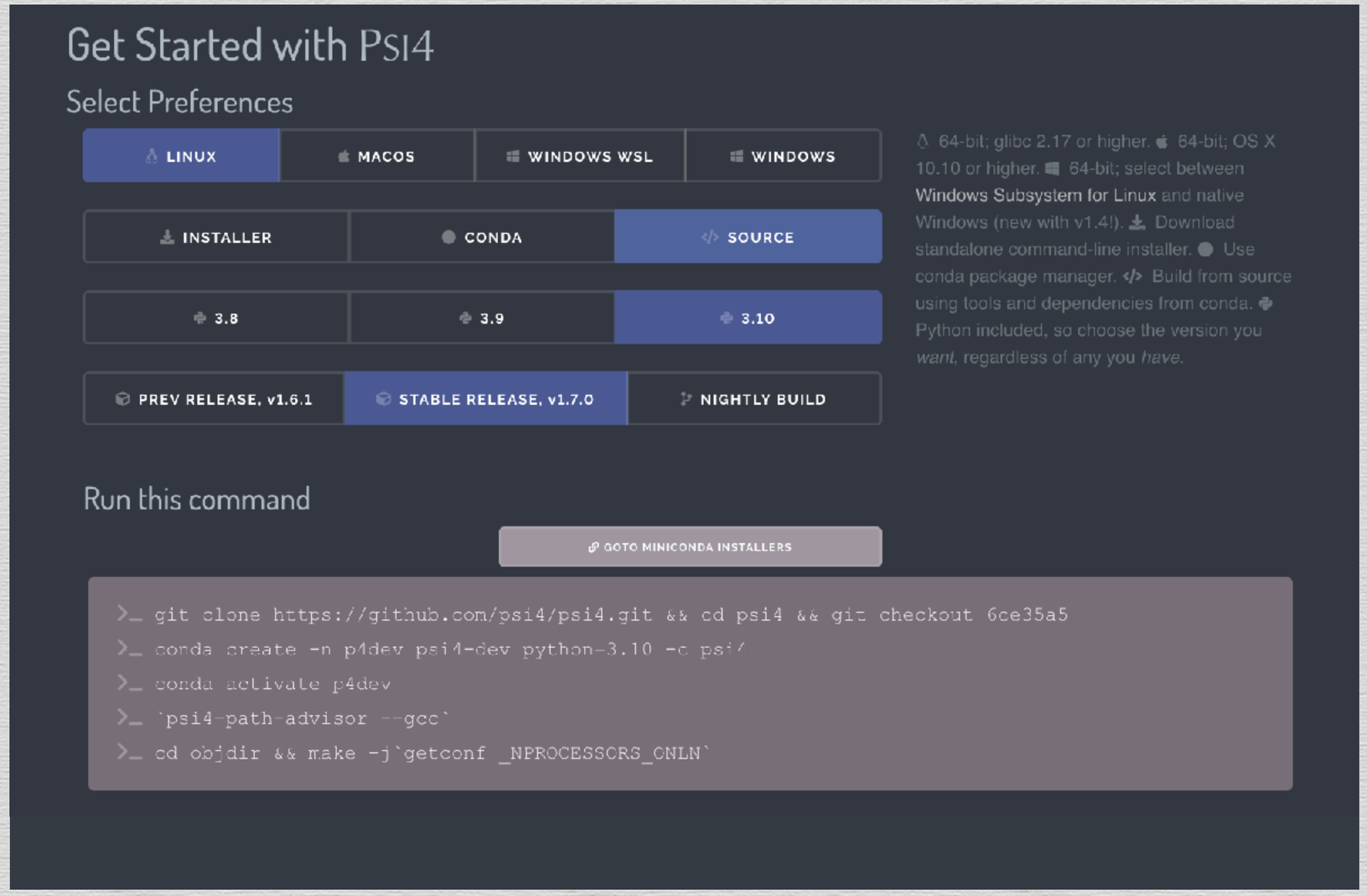
234K

PSI4 · dependencies · add-ons



BINARY INSTALLER

https://psicode.org/installs/latest



PY-WARY USERS: CONDA INSTALLER

573K

Psi4 · dependencies · add-ons

PY-FRIENDLY USERS: CONDA PACKAGE

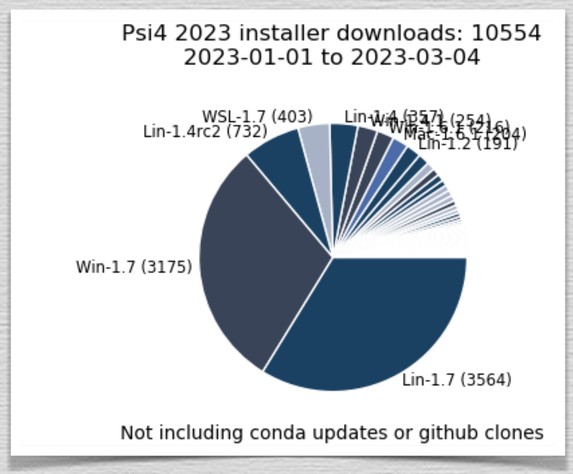
234K

Psi4 · dependencies · add-ons

CORE DEVELOPERS: CONDA-ENABLED SUPERBUILD

10K

deps · dev tools · add-ons · cmake cmd



Psi4

Jerome Gonthier
Berkeley

Rob Parrish
Stanford

Holger Kruse
Czech Acad. Sci.

Rollin King
Bethel

Alexander Sokolov
Ohio State

David Sherrill
GaTech

Lori Burns
GaTech

Asim Alenaizan
KFUPM

Maximilian Scheurer
Heidelberg

Zach Glick
GaTech

Jeff Schriber
Iona

Francesco Evangelista
Emory

Eugene DePrince
FSU

Fritz Schaefer
UGA

Justin Turney
UGA

Daniel Crawford
VaTech

Daniel Smith
MolSSI

Konrad Patkowski
Auburn

Ben Pritchard
MolSSI

Jonathon Misiewicz
Emory

Ed Valeev
VaTech

Andy Simmonett
NIH

Ed Hohenstein
CCNY

Roberto Di Remigio
ENCCS

Ugur Bozkaya
Hacettepe

Peter Kraus
TUB

Susi Lehtola
Helsinki

QCENGINE

ADCC

Maximilian Scheurer
Heidelberg

Michael Herbst
Aachen

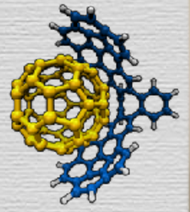
Andreas Dreuw
Heidelberg

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

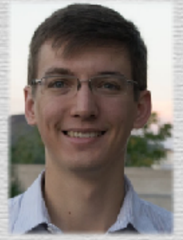
Roberto Di Remigio
Uppsala

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 → Entos

Levi Naden
MolSSI

Doaa Altarawy
MolSSI

Matthew Welborn
MolSSI → Entos

Ben Pritchard
MolSSI

MolSSI

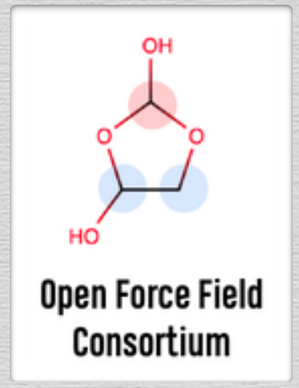
Theresa Windus
Iowa State

Daniel Crawford
VaTech

Jessica Nash
MolSSI

Sam Ellis
MolSSI

Susi Lehtola
Helsinki



openforcefield.org/

MolSSI

qcarchive.molssi.org

QCSchema

Daniel Smith
MolSSI → Entos

Aaron Virshup
Arzeda

Bert de Jong
LBNL

Geoff Hutchison
U Pittsburgh

Marcus Hanwell
Kitware → Brookhaven

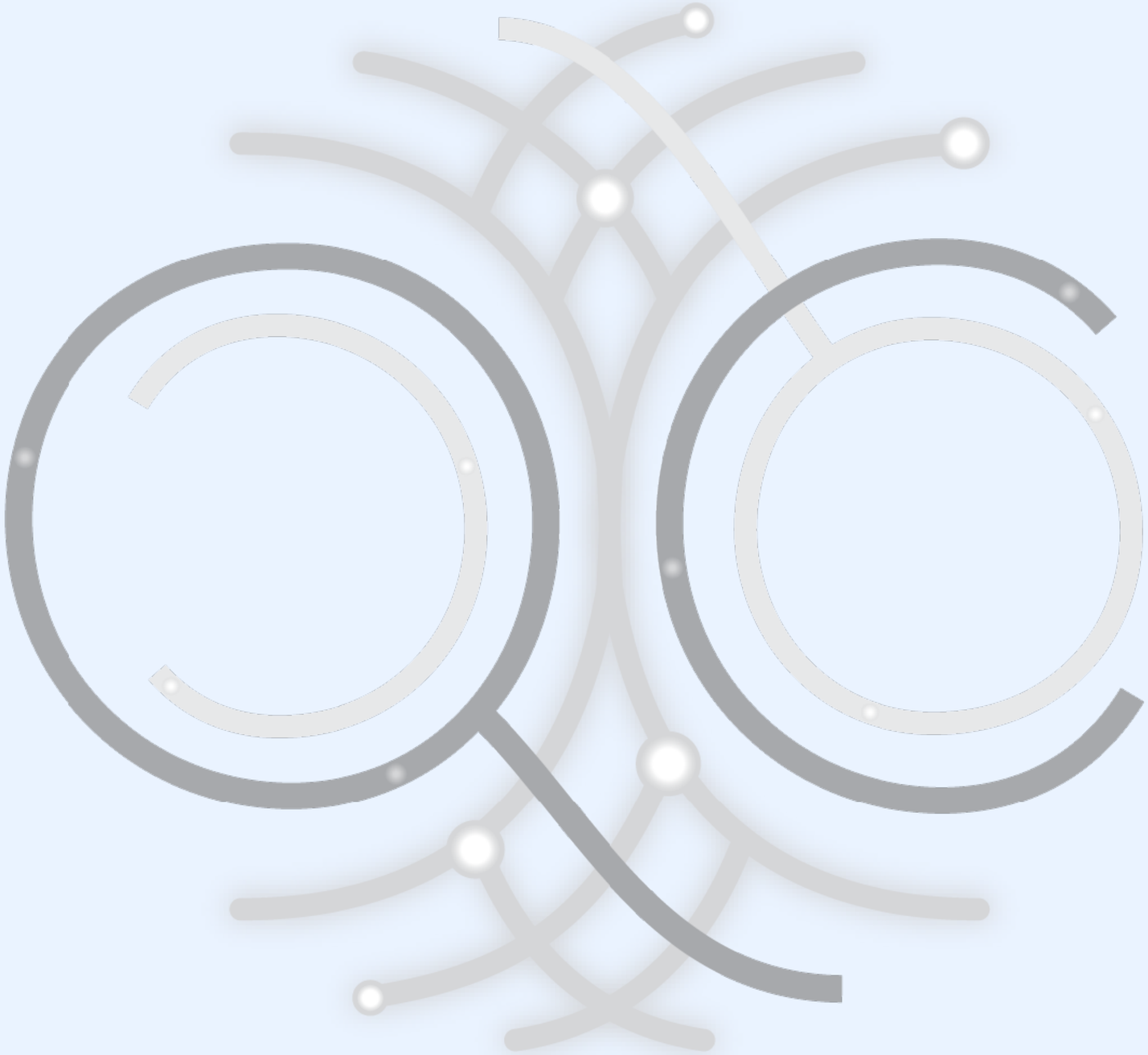
Eric Berquist
Q-Chem

Ben Pritchard
MolSSI

Lori Burns
GaTech

Matthew Welborn
MolSSI → Entos

Colton Hicks
Stanford



PUBLICATIONS

Home > The Journal of Chemical Physics > Volume 152, Issue 18 > 10.1063/5.0006002

No Access • Submitted: 26 February 2020 • Accepted: 12 April 2020 • Published Online: 13 May 2020

PSI4 1.4: Open-source software for high-throughput quantum chemistry

J. Chem. Phys. 152, 184108 (2020); https://doi.org/10.1063/5.0006002

Daniel G. A. Smith[1], Lori A. Burns[2], Andrew C. Simmonett[3], Robert M. Parrish[2], Matthew C. Schieber[2], Raimondas Galvelis[4], Peter Kraus[5], Holger Kruse[6], Roberto Di Remigio[7], Asem Alenaizan[2], Andrew M. James[8], Susi Lehtola[9], Jonathon P. Misiewicz[10], Maximilian Scheurer[11], Robert A. Shaw[12], Jeffrey B. Schriber[2], Yi Xie[2], Zachary L. Glick[2], Dominic A. Sirianni[2], Joseph Senan O'Brien[2], Jonathan M. Waldrop[13], Ashutosh Kumar[8], Edward G. Hohenstein[14], Benjamin P. Pritchard[1], Bernard R. Brooks[3], Henry F. Schaefer III[10], Alexander Yu. Sokolov[15], Konrad Patkowski[13], A. Eugene DePrince III[16], Uğur Bozkaya[17], Rollin A. King[18], Francesco A. Evangelista[19], Justin M. Turney[10], T. Daniel Crawford[1,a], *and* C. David Sherrill[2,a)]

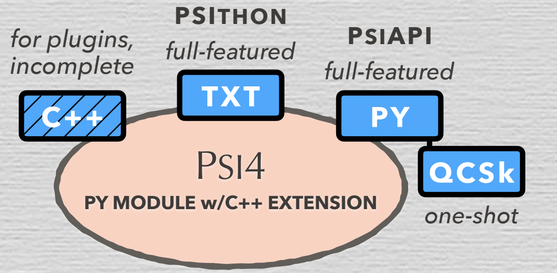
WIREs COMPUTATIONAL MOLECULAR SCIENCE

Advanced Review

The MolSSI QCArchive project: An open-source platform to compute, organize, and share quantum chemistry data

Daniel G. A. Smith, Doaa Altarawy, Lori A. Burns, Matthew Welborn, Levi N. Naden, Logan Ward, Sam Ellis, Benjamin P. Pritchard, T. Daniel Crawford

First published: 31 July 2020 | https://doi.org/10.1002/wcms.1491 | Cita



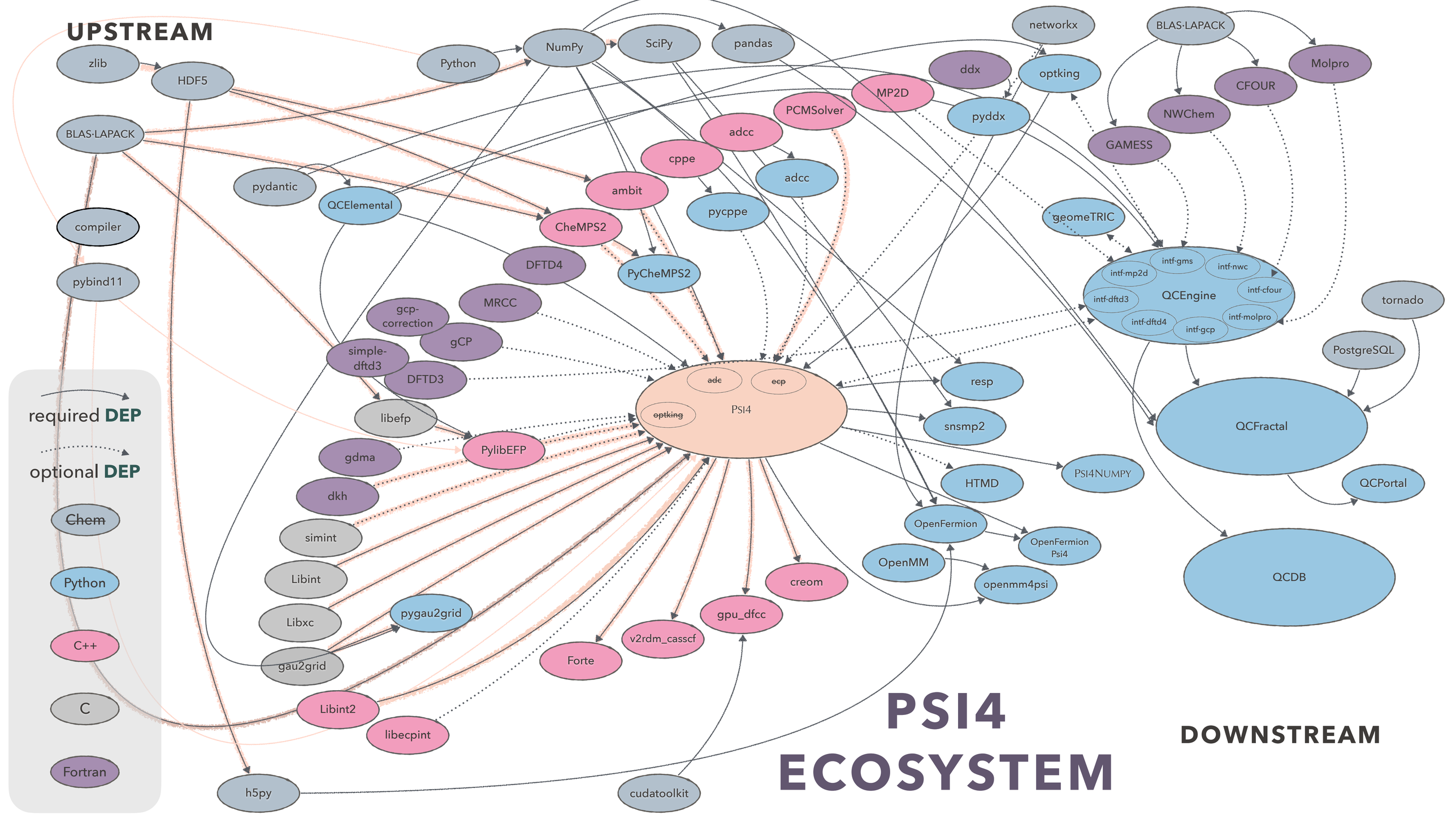
Home > The Journal of Chemical Physics > Volume 155, Issue 20 > 10.1063/5.0059356

No Access • Submitted: 08 June 2021 • Accepted: 01 October 2021 • Published Online: 22 November 2021

Quantum Chemistry Common Driver and Databases (QCDB) and Quantum Chemistry Engine (QCEngine): Automation and interoperability among computational chemistry programs

J. Chem. Phys. 155, 204801 (2021); https://doi.org/10.1063/5.0059356

Daniel G. A. Smith[1,a)], Annabelle T. Lolinco[2], Zachary L. Glick[3], Jiyoung Lee[2,b)], Asem Alenaizan[3,c)], Taylor A. Barnes[1], Carlos H. Borca[3], Roberto Di Remigio[4,d)], David L. Dotson[5], Sebastian Ehlert[6], Alexander G. Heide[7], Michael F. Herbst[8], Jan Hermann[9], Colton B. Hicks[10,11], Joshua T. Horton[12], Adrian G. Hurtado[13], Peter Kraus[14], Holger Kruse[15], Sebastian J. R. Lee[16], Jonathon P. Misiewicz[7,e)], Levi N. Naden[1], Farhad Ramezanghorbani[17], Maximilian Scheurer[18], Jeffrey B. Schriber[3], Andrew C. Simmonett[19], Johannes Steinmetzer[20], Jeffrey R. Wagner[5,21], Logan Ward[22], Matthew Welborn[1,a)], Doaa Altarawy[1,23], Jamshed Anwar[12], John D. Chodera[24], Andreas Dreuw[18], Heather J. Kulik[25], Fang Liu[25,e)], Todd J. Martínez[10,11], Devin A. Matthews[26,f)], Henry F. Schaefer III[7], Jiří Šponer[15], Justin M. Turney[7], Lee-Ping Wang[27], Nuwan De Silva[2,g)], Rollin A. King[28], John F. Stanton[29], Mark S. Gordon[30], Theresa L. Windus[30], C. David Sherrill[3], *and* Lori A. Burns[3,h)]


UPSTREAM
zlib
HDF5
Python
NumPy
SciPy
pandas
networkx
BLAS·LAPACK
Molpro
ddx
optking
CFOUR
MP2D
PCMSolver
pyddx
NWChem
BLAS·LAPACK
adcc
GAMESS
cppe
adcc
pydantic
ambit
QCElemental
pycppe
geomeTRIC
compiler
CheMPS2
DFTD4
PyCheMPS2
intf·gms
intf·nwc
intf·mp2d
QCEngine
intf·cfour
intf·dftd3
tornado
pybind11
MRCC
gcp-correction
gCP
intf·molpro
intf·dftd4
intf·gcp
PostgreSQL
simple-dftd3
DFTD3
resp
adc
ecp
PSI4
optking
QCFractal
libefp
snsmp2
PylibEFP
gdma
PSI4NUMPY
HTMD
QCPortal
dkh
OpenFermion
simint
OpenFermion Psi4
OpenMM
Libint
creom
QCDB
openmm4psi
Libxc
pygau2grid
gpu_dfcc
v2rdm_casscf
gau2grid
Forte
Libint2
libecpint
h5py
cudatoolkit
required DEP
optional DEP
Chem
Python
C++
C
Fortran
PSI4
ECOSYSTEM
DOWNSTREAM


1970
1980
1990
2000
2010
2020

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

Curt Janssen

Ed Seidl

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

PSI3: An open-source *Ab Initio* electronic structure package

T. Daniel Crawford, C. David Sherrill, Edward F. Valeev, Justin T. Fermann, Rollin A. King, Matthew L. Leininger, Shawn T. Brown, Curtis L. Janssen, Edward T. Seidl, Joseph P. Kenny, Wesley D. Allen ... **See fewer authors**

First published: 09 April 2007 | https://doi.org/10.1002/jcc.20573 | Citations: 253

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

2011-2012
PSI4: PAPER & BETAS

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

Andy Simmonett

Rob Parrish

Lori Burns

Daniel Smith

Jonathon Misiewicz

Psi4: an open-source *ab initio* electronic structure program

Justin M. Turney, Andrew C. Simmonett, Robert M. Parrish, Edward G. Hohenstein, Francesco A. Evangelista, Justin T. Fermann, Benjamin J. Mintz, Lori A. Burns, Jeremiah J. Wilke, Micah L. Abrams, Nicholas J. Russ, Matthew L. Leininger, Curtis L. Janssen, Edward T. Seidl, Wesley D. Allen, Henry F. Schaefer, Rollin A. King, Edward F. Valeev, C. David Sherrill, T. Daniel Crawford ... **See fewer authors**

First published: 31 October 2011 | https://doi.org/10.1002/wcms.93 | Citations: 655

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

2011-2012
PSI4: PAPER & BETAS

JULY 2016
PSI4: 1.0 RELEASE

OCT 2016
THE INVERSION

MAY 2017
1.1, LGPL

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

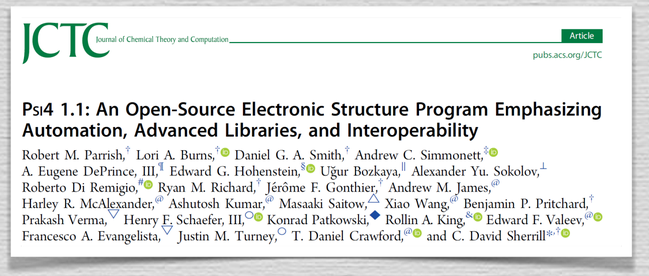
Andy Simmonett

Rob Parrish

Lori Burns

Daniel Smith

Jonathon Misiewicz



JCTC Journal of Chemical Theory and Computation — Article — pubs.acs.org/JCTC

Psi4 1.1: An Open-Source Electronic Structure Program Emphasizing Automation, Advanced Libraries, and Interoperability

Robert M. Parrish, Lori A. Burns, Daniel G. A. Smith, Andrew C. Simmonett, A. Eugene DePrince, III, Edward G. Hohenstein, Uğur Bozkaya, Alexander Yu. Sokolov, Roberto Di Remigio, Ryan M. Richard, Jérôme F. Gonthier, Andrew M. James, Harley R. McAlexander, Ashutosh Kumar, Masaaki Saitow, Xiao Wang, Benjamin P. Pritchard, Prakash Verma, Henry F. Schaefer, III, Konrad Patkowski, Rollin A. King, Edward F. Valeev, Francesco A. Evangelista, Justin M. Turney, T. Daniel Crawford, and C. David Sherrill

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

2011-2012
PSI4: PAPER & BETAS

JULY 2016
PSI4: 1.0 RELEASE

OCT 2016
THE INVERSION

MAY 2017
1.1, LGPL

JUL 2018: 1.2

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

Andy Simmonett

Rob Parrish

Lori Burns

Daniel Smith

Jonathon Misiewicz

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

2011-2012
PSI4: PAPER & BETAS

JULY 2016
PSI4: 1.0 RELEASE

OCT 2016
THE INVERSION

MAY 2017
1.1, LGPL

JUL 2018: 1.2

FEB 2019: 1.3

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

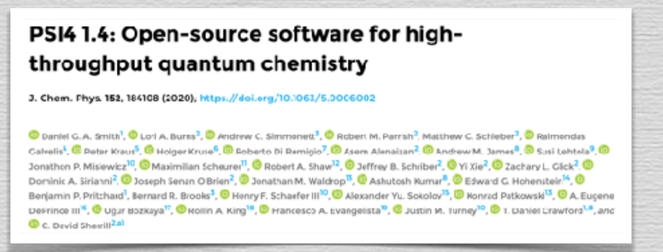
Andy Simmonett

Rob Parrish

Lori Burns

Daniel Smith

Jonathon Misiewicz



PSI4 1.4: Open-source software for high-throughput quantum chemistry

J. Chem. Phys. 152, 184108 (2020), https://doi.org/10.1063/5.0006002

Daniel G. A. Smith[1], Lori A. Burns[2], Andrew C. Simmonett[3], Robert M. Parrish[3], Matthew C. Schieber[3], Raimondas Galvelis[4], Peter Kraus[5], Holger Kruse[6], Roberto Di Remigio[7], Asem Alenaizan[8], Andrew M. James[9], Susi Lehtola[9], Jonathon P. Misiewicz[10], Maximilian Scheurer[11], Robert A. Shaw[12], Jeffrey B. Schriber[2], Yi Xie[2], Zachary L. Glick[2], Dominic A. Sirianni[2], Joseph Senan O'Brien[2], Jonathan M. Waldrop[13], Ashutosh Kumar[8], Edward G. Hohenstein[14], Benjamin P. Pritchard[1], Bernard R. Brooks[3], Henry F. Schaefer III[10], Alexander Yu. Sokolov[15], Konrad Patkowski[13], A. Eugene DePrince III[16], Uğur Bozkaya[17], Rollin A. King[18], Francesco A. Evangelista[19], Justin M. Turney[10], T. Daniel Crawford[1,a], and C. David Sherrill[2,a]

1970 1980 1990 2000 2010 2020

Fritz Schaefer

MISTS OF TIME
BERKELEY: FORTRAN

1987
PSI: FORTRAN IV, UGA

1989
PSI2: FORTRAN/C, UNIX

1999
PSI3: C/C++ OPEN-SOURCE GPL

2007
PSI4: J. TURNEY STARTS LIBMINTS

2011-2012
PSI4: PAPER & BETAS

JULY 2016
PSI4: 1.0 RELEASE

OCT 2016
THE INVERSION

MAY 2017
1.1, LGPL

JUL 2018: 1.2

FEB 2019: 1.3

AUG 2021: 1.4

NOV 2021: 1.5

MAY 2022: 1.6

DEC 2022: 1.7

Curt Janssen

Ed Seidl

David Sherrill

Daniel Crawford

Justin Turney

Andy Simmonett

Rob Parrish

Lori Burns

Daniel Smith

Jonathon Misiewicz