THE DISTRIBUTED DRIVER, DDD

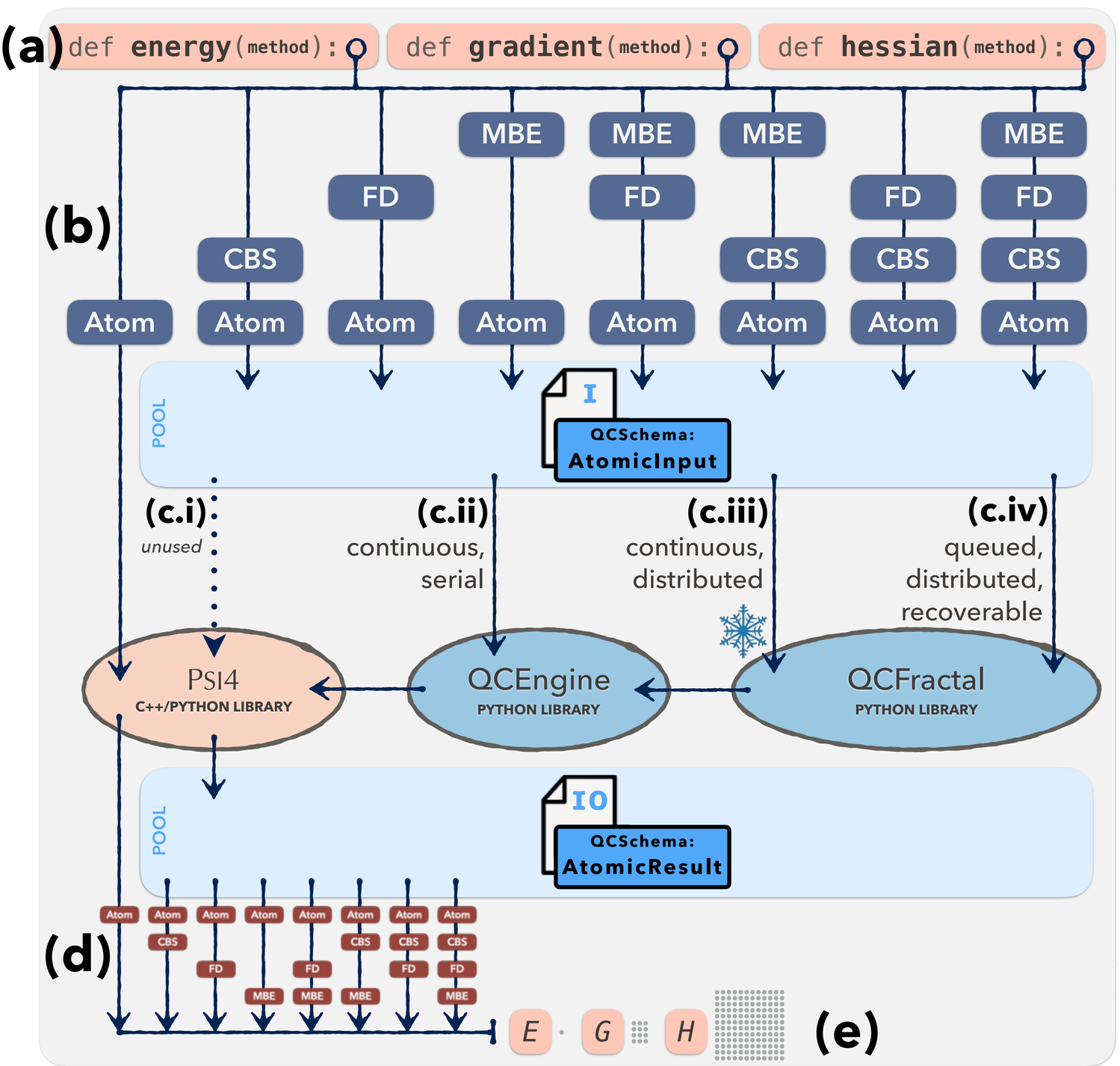
LORI A. BURNS

PSICON 2020, BLUEJEANS
3-5 DECEMBER 2020

DISTRIBUTED DRIVER

why and when

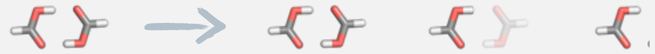
- **PARALLELISM** several popular driver features (e.g., optimization, cbs, nbody) suited to plesant parallelism but couldn't take advantage.
- **OUTPUT** mix of output file and logging file so watch for .out and .log and make suggestions
- **GLOBALS** nice side effect is that it's forcing local storage (wfn, not globals) and checking API contracts and forcing independence of consecutive API jobs
- **TIMELINE** first outlined by Daniel Smith at PsiCon2018 and been filled out since.
- **WHAT** separates our multi-job post-processing into plan, compute, assemble stages with jobs stored as QCSchema AtomicInputs. post-processing is nestable into cbs(findif(single-points)).
- **MERGED?** no, as soon as v1.4 is minted. needs more eyes before stable release.



(z)

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
```

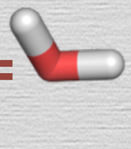
QCSchema

ASM Assemble extrapolations & total results from modelchems.

class AtomicComputer ():

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

ASM Return analytic energy, gradient, or Hessian.

psi4.optimize('HF/cc-pv[d,t]z', bsse_type='cp', molecule= )

DDD & FRACTAL DEMO

CP-CORR CBS OPT

```python
from qcfractal import FractalSnowflake
from qcfractal.interface import FractalClient

# Build a active server and client
snowflake = FractalSnowflake()
client = FractalClient(snowflake)
print(client)


def psi_model(coords):

    dimer = psi4.geometry("""O 0 0 0\n H 1 0 0\n H 0 1 0\n --\n O 3 3 3\n H
    dimer.update_geometry()
    dimer.set_geometry(psi4.core.Matrix.from_array(coords))

    plan = psi4.gradient("HF/cc-pV[D, T]Z", bsse_type="CP", molecule=dimer,
                         return_plan=True, return_total_data=True)

    plan.compute(client)

    snowflake.await_results()
    ret = plan.get_results(client)

    return (ret["extras"]["qcvars"]["CURRENT ENERGY"],
   np.array(ret["extras"]["qcvars"]["CURRENT GRADIENT"]).reshape(-1, 3))
```

QC Archive

A MolSSI Project

psi4.optimize('HF/cc-pv[d,t]z', bsse_type='cp', molecule=