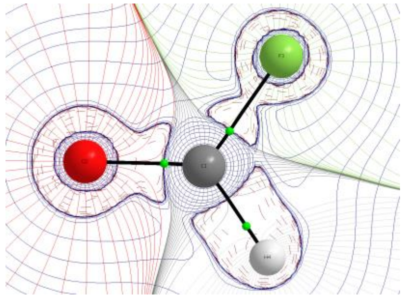
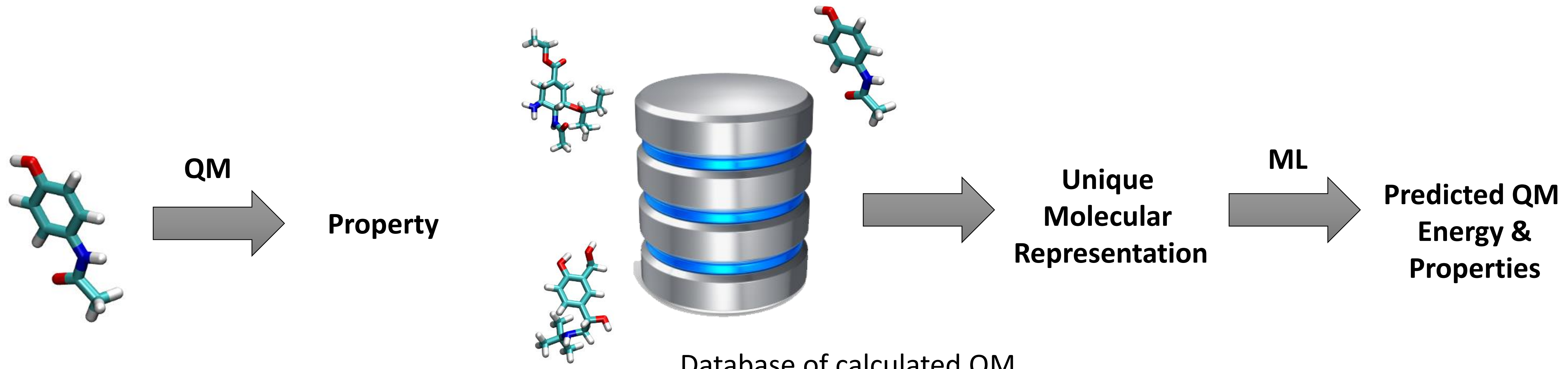


Use of master-worker and integration with OSG Connect

Roman Zubatyuk

Department of Chemistry
Carnegie Mellon University

Supervised machine learning on quantum-chemical data

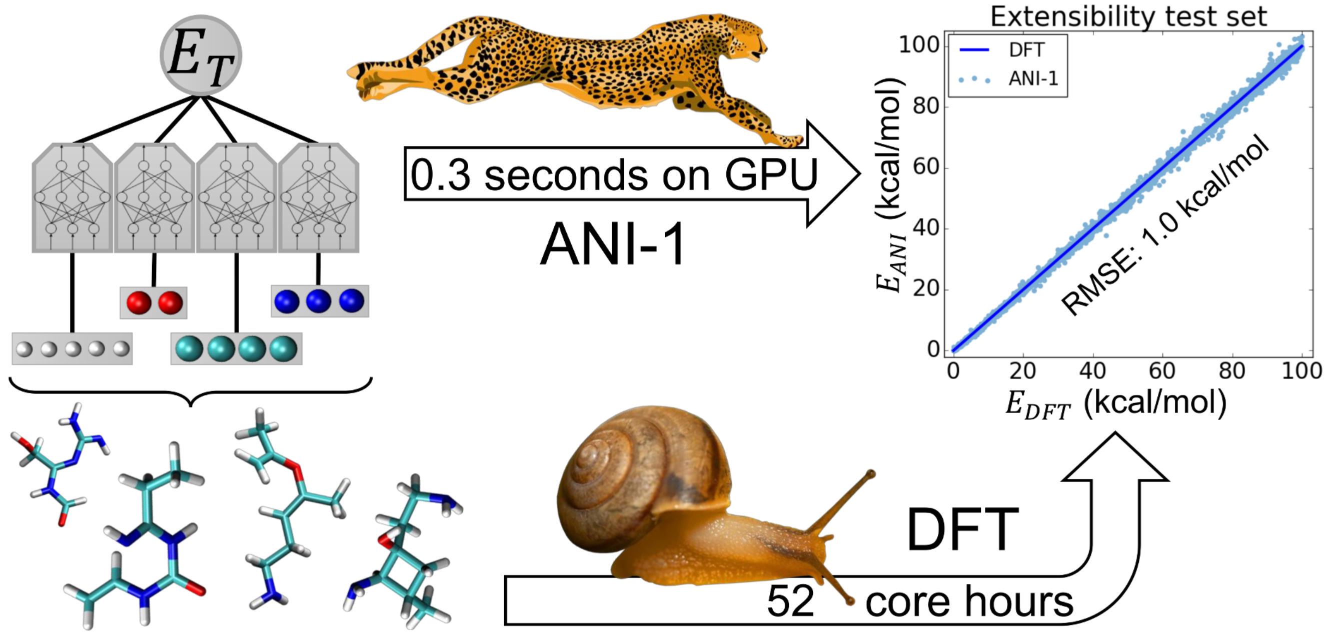

QM
Property
Unique Molecular Representation
ML
Predicted QM Energy & Properties

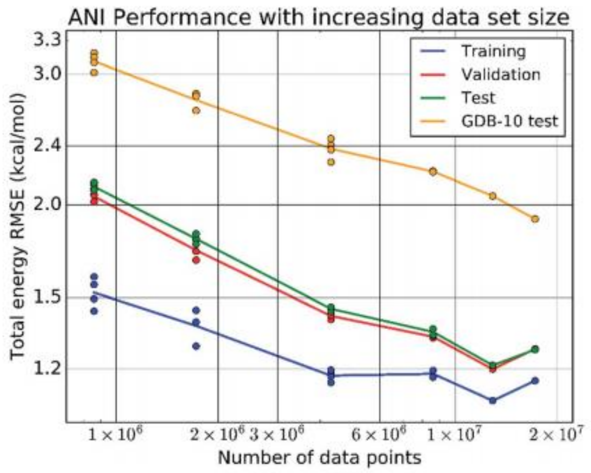
Database of calculated QM energies and properties for 50M molecules

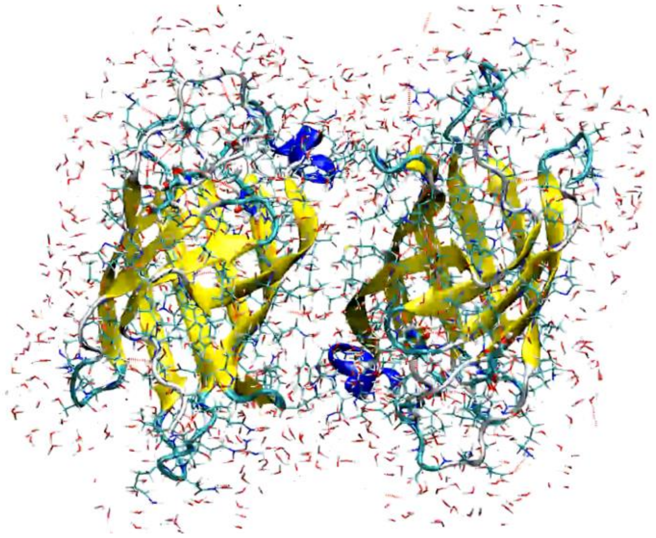
- Traditional quantum mechanics: Slow calculations, one molecule at a time
- QSAR: Statistical modeling of historical experimental data
- Now we could use accumulated historical QM data to train a statistical models that can accurately predict results of quantum mechanics

J. Smith, O. Isayev, A. Roitberg. ***Chem. Sci.***, 2017, **8**, 3192-3203

Fast, accurate, transferable and extensible neural network potentials







ANI1: 20M DFT calculations (CHNO)
ANI-1x: 5M DFT calculations (CHNO)
ANI-1ccx: 0.5M CCSD(T)/CBS (CHNO)
ANI-2x, AIMNet: 9M DFT calculations (CHNOSFCl)

Current development: more chemical elements, charged molecules.

Smith, Justin S., Olexandr Isayev, and Adrian E. Roitberg. "ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost." *Chemical science* 8.4 (2017): 3192-3203.
Smith, Justin S., et al. "Less is more: Sampling chemical space with active learning." *The Journal of chemical physics* 148.24 (2018): 241733.
Smith, Justin S., et al. "Approaching coupled cluster accuracy with a general-purpose neural network potential through transfer learning." *Nature communications* 10.1 (2019): 2903.
Zubatyuk, Roman et al., “Accurate and Transferable Multitask Prediction of Chemical Properties with an Atoms-in-Molecules Neural Network.” Sci. Adv. 2019, 5 (8), eaav6490.

Our computational tasks

- DFT calculations for small molecules (ORCA)
- Semiempirical molecular dynamics (XTB)
- Molecular docking (AutoDock Vina)

- ~ 10^4 – 10^6 tasks
- Short (20s - 2h CPU time)
- Small input size (1 – 100 kb)
- Portable software stack

=> Ideal for OSG

“Free” opportunistic computational resource

Single core, small memory, small disk, short run time.

Standard workflow

- Create inputs for tasks, transfer to submit host.
- Transfer to submit host
- Write job execution script
- Submit to Condor (or SLURM)
- Wait
- Resubmit failed tasks

~ 10^6 tasks

group several tasks together: all done or all fail

Master-worker workflow

- Put tasks to a master database.
- Write script to perform single task
- Launch workers what execute tasks.
- Workers communicate with the master (get task, put results)

$\Rightarrow$ **Data are organized, tasks are independent, computer resource use is efficient**

$\Rightarrow$ **Easy to use**

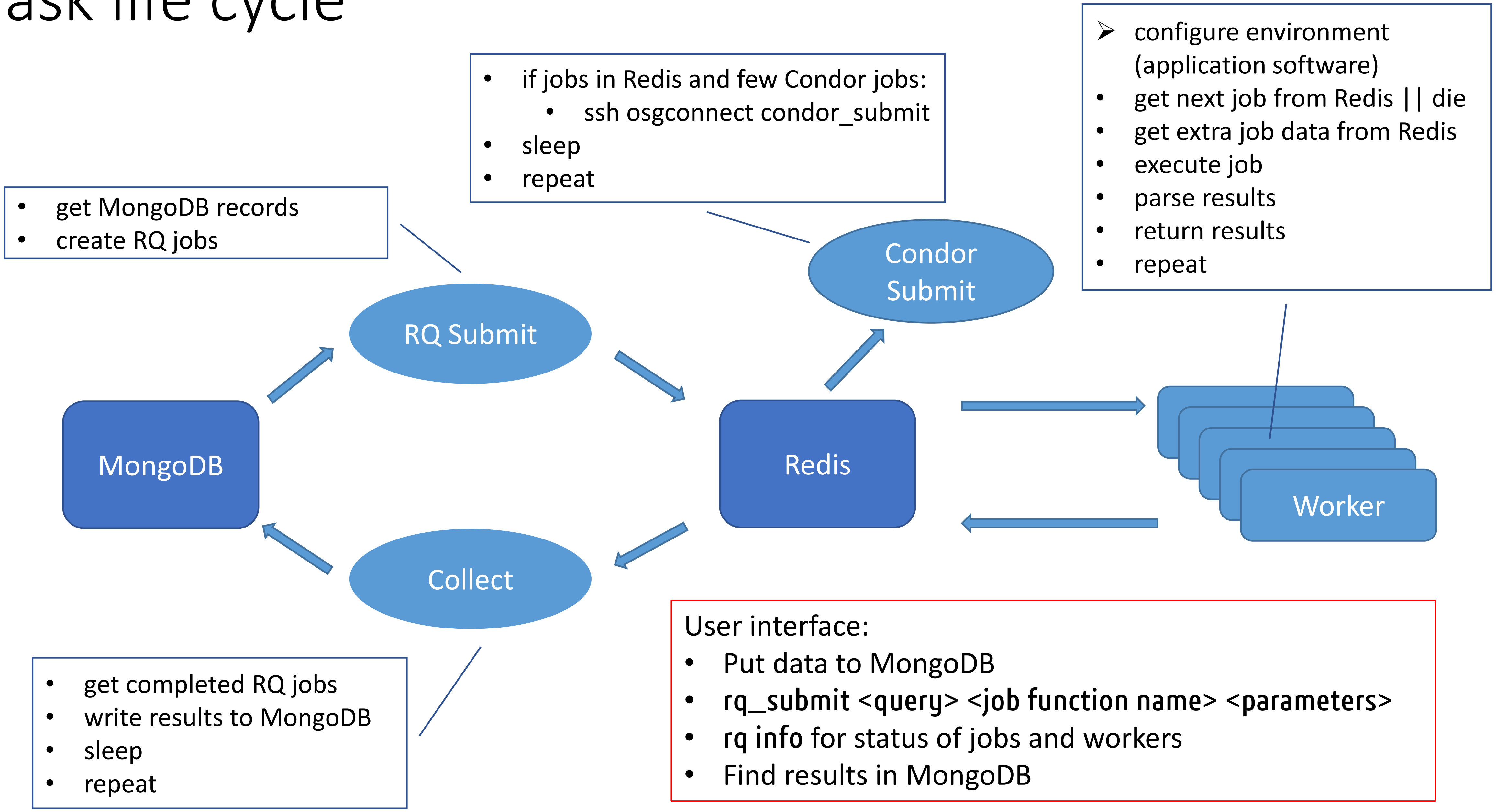
Master-worker implementation

- Message queue database (message == task)
- Consumers execute tasks (each message should be delivered only once)

- Implemented RQ (*Redis Queue*) Python library
  - Other alternatives are Celery, Huey2, Dramatiq, Ray, Uber Cherami, and many more.

- We did not re-invented the wheel. We made a wheel without bells and whistles that fits our vehicle:
  - RQ is simple, scalable and customizable
  - RQ workers need very basic environment
  - Tasks are simple Python functions
  - Single-user environment

Task life cycle



- get MongoDB records
- create RQ jobs

- if jobs in Redis and few Condor jobs:
  - ssh osgconnect condor_submit
- sleep
- repeat

- ➢ configure environment (application software)
- get next job from Redis || die
- get extra job data from Redis
- execute job
- parse results
- return results
- repeat

- get completed RQ jobs
- write results to MongoDB
- sleep
- repeat

User interface:

- Put data to MongoDB
- **rq_submit <query> <job function name> <parameters>**
- **rq info** for status of jobs and workers
- Find results in MongoDB

Job definition file

```
campaign: wb97mv

campaign_config:
  func_name: htrq.htrun.scripts.orca.sequential     <- job function (run ORCA)
  job_timeout: 15000
  args:
    - name: wb97md3bj
      route: |
        ! wB97m-d3bj def2-tzvpp def2/j rijcosx engrad tightscf
        %elprop dipole true quadrupole true end               <- what to compute (energy & gradient)
        %output PrintLevel mini Print[P_DFTD_GRAD] 1 end
        %scf maxiter 256 end
      parsers:
        stdout:
          - htrq.htrun.parser.orca.total_energy
          - htrq.htrun.parser.orca.gradient          <- how to parse data
          - htrq.htrun.parser.orca.dipole
      keep_files:
        stdout: '{id}_wb97md3bj.out'
        orca.gbw: '{id}_wb97md3bj.gbw’          <- some results could be stored on file system instead of MongoDB
  kwargs:
    mongo_output_key: wb97mv               <- how to store results in MongoDB

submit:
  query:
    wb97mv.wb97md3bj: null
  projection: []                           <- how to select and submit jobs
  queue: orca:high
```

RQ job data

- Database, ID
- Input data (coordinates of atoms, etc.)
- Job Python function *(e.g. DFT energy + gradient calculation)*
- Redis key containing parameters of job function *(e.g. DFT functional and basis set)*

=> Unique job ID

Worker environment

- Python 3.5+ *(OCG Connect CVMFS)*
- python-rq and python code to execute task *(OSG Connect Stash < 10 MB)*
- Application software binaries (OSG Connect Stash < 500 MB)

Performance of RQ

- MongoDB, Redis, submit and collect scripts on a single mid-grade workstation (i7, 32GB RAM)
    - 10M tasks in Redis Queue
    - 10,000 workers at a time (probably, could be few times more)
    - 100 jobs/sec
    - 1 Gbps sustained incoming network traffic to Redis

About 20 M CPU core hours consumed on OSG, XSEDE and TACC Frontera with same RQ-based framework!

Acknowledgements

Funding:

NSF

CHE-1802789

DEPARTMENT OF THE NAVY ONR Science & Technology

UNC ESHELMAN INSTITUTE FOR INNOVATION

HPC Computing:

Open Science Grid

XSEDE Extreme Science and Engineering Discovery Environment

Los Alamos NATIONAL LABORATORY EST. 1943

NVIDIA