Colmena: Parsl for Intelligent Workflows on Exascale HPC

Logan Ward
Assistant Computational Scientist
Data Science and Learning Division
Argonne National Laboratory

17 October 2023
I've been bothered by high-throughput searches for a decade

Each point is ~5 CPU-hr (~2.5 kW-hr, ~500g coal burned*)

There are few reasons we should be running brute-force

Potential Solution: Machine Learning

Figure: Kirklin et al. Acta Mat (2016)

*Assuming 500W node, US EPA Greenhouse Gas Calculator
There are easy opportunities for adding intelligence to an application. Here's an example of how this might be approached:

```python
futures = [
    do_science(task)
    for task in everything
]
results = [
    task.result()
    for task in futures
]
report_findings(results)
```

Today's Talk: How can we make "The Application" smarter?
Getting More from Your HPC with Colmena
Active Learning is a better plan, and not my idea at all
See also: Bayesian Optimization, Surrogate Optimization, Optimal Experimental Design...

How would be integrate these concepts into our application?
### Inside the Workflow: join_app

**General Idea:** Tasks which make new tasks

**Advantages:**
- DFK handles all effort
- Simple instructions -> simple functions

**Challenges:**
- *How should(!?) join_app tasks share state?*

### Outside the Workflow: Events, Threads

**General Idea:** Steering logic from “main.py”

**Advantages:**
- Clearer control over concurrency
- Explicit control over shared state
- Respond to events besides “task complete”

**Challenges:**
- *How would one write such a thing?*
Colmena is a wrapper over Exascale Workflow tools

Programming Model: Task Queues

```python
# Primitive Units
queue.send_inputs(1)
result = queue.get_result()
```

Programming Model: Agents

```python
class Thinker(BaseThinker):
    @agent
def make_work(self):
        self.queue.send_inputs(1)
```

Task Server:
- Dispatches work requests to compute
- Communicates results back to thinker

Backend:
- Supports most HPC and cloud services
- Easily configure multiple worker types, multi-site workflows
- Limited support for ensembles of MPI applications
- **Future:** Balsam, FuncX, RCT
Example application: “Interleaved,” AI-in-the-loop optimizer

Details: Ward et al. ML4HPC, SC21. 
Tutorial: colmena.rtfd.io/en/latest/examples.html#tutorials
So, what’s new in ‘23?

*Building more apps, learning more requirements!*
ExaMol: An attempt at user-friendly Colmena

Please, don’t look at v0-v4 our molecular design applications

Step 1: Write a spec

```python
recipe = RedoxEnergy(charge=1,
                       compute_config='xtb')

spec = ExaMolSpecification(
    database='training-data.json',
    recipe=[recipe],
    search_space='search_space.smi',
    selector=GreedySelector(n_to_select=8),
    simulator=ASESimulator(scratch_dir='/tmp'),
    scorer=RDKitScorer(recipe),
    models=[[KNeighborsRegressor()]],
    num_to_run=8,
    thinker=SingleStepThinker,
    compute_config=config,
    run_dir='run'
)
```

Step 2: Execute

```
examol run spec.py:spec
```

Disclosure: I watch to see how Alex Brace does things 😌

https://exalearn.github.io/ExaMol/
MOFA: Persistent workers would save time

Large start-up cost, but always running

https://github.com/globus-labs/mof-generation-at-scale
FFF: Could I schedule tasks based on worker availability?

Success: We can run on two systems easily

Ugly Secret: We only do this if GPUs available quickly
Why? Training sets would grow stale in queues

https://github.com/exalearn/fast-finetuned-forcefields*

*I want to merge this with PsiFlow!
Conclusions and Future for Colmena

What did we build?

Colmena lets you build complex steering policies

- Thinker
- QC-Scorer
- QC-Recorder
- Trainer
- Updater
- ML-Scorer
- ML-Recorder

... that get more out of your HPC

What to watch for next year?

- More Colmena applications
  - PsiFlow: If Sander agrees 😃
  - ExaMol: Maybe with some Real Chemist users
  - MOFA: A target for many AI apps on ExaScale

- Continued integration with Parsl/Globus Compute
  - Demonstrating Yadu’s MPI support?
  - Apps that produce intermediate results?
  - Events and Hooks from Providers?
Acknowledgements: The (growing!) team

**Argonne: ExaLearn** – Using AI with HPC
Yadu Babuji, Ben Blaiszik, Ryan Chard, Kyle Chard, Ian Foster, Greg Pauloski, Ganesh Sivaraman, Rajeev Thakur

**Argonne: JCESR** – Molecular modeling for batteries
Rajeev Assary, Larry Curtiss, Naveen Dandu, Paul Redfern

**MoLSSI** – Workflows for quantum chemistry
Lori A. Burns, Daniel Smith, Matt Welborn, *many other open-source contributors*

**PNNL: ExaLearn** – Graph algorithms for learning
Sutanay Choudhury, Jenna Pope

**BNL: ExaLearn** – Optimal experimental design
Frank Alexander, Shantenu Jha, Kris Reyes, Li Tan, Byung Jun, *and more*

**Argonne ALCF** – AI, Data and Simulation on HPC
Murali Emani, Alvaro Vazquez-Mayagoitia, Venkat Vishnawath

**ExaWorks** – Interfacing to HPC
Ayman Alsaadi, Matteo Turilli, Shantenu Jha, Kyle Chard

**FuncX** – Seamless multisite deployment
Kevin Hunter Kesling, Kyle Chard, Ryan Chard, Ben Clifford, *and more*

**Ensemble Group** – Defining ensemble needs
John-Luke Navarro, Jonathon Ozik, Tom Peterka, Stephen Hudson, Orçun Yildiz, Alex Brace, Arvind Ramanathan, *and more*