Parsl: Developing Interactive Parallel Workflows in Python using Parsl

Kyle Chard (chard@uchicago.edu)
Yadu Babuji, Anna Woodard, Zhuozhao Li, Ben Clifford, Ian Foster, Dan Katz, Mike Wilde, Justin Wozniak

http://parsl-project.org
Parsl: Interactive parallel scripting in Python

Annotate functions to make Parsl apps
- Python apps call Python functions
- Bash apps call external applications

Apps return “futures”: a proxy for a result that might not yet be available

Apps run concurrently respecting data dependencies. Natural parallel programming!

Parsl scripts are independent of where they run. Write once run anywhere!

```
@python_app
def hello():
    return 'Hello World!'

print(hello().result())
Hello World!
```

```
@bash_app
def echo_hello(stdout='echo-hello.stdout'):
    return 'echo "Hello World!"'

echo_hello().result()

with open('echo-hello.stdout', 'r') as f:
    print(f.read())
Hello World!
```
Creating dynamic dataflows

Parsl creates a dynamic graph of tasks and their data dependencies
- Implicit based on shared input/output between apps

Tasks are only executed when their dependencies are met

Tasks without shared dependencies execute concurrently
Parsl in action: dynamic dataflow execution
Parsl is Python

- Use Python libraries natively
- Stage Python data transparently
- Integrates with Python ecosystem

```
pip3 install parsl
```
Parsl scripts are execution provider and execution model independent

The same script can be run locally, on grids, clouds, or supercomputers

A single script may use many execution providers
- Local, Cloud (AWS, Azure, private), Slurm, Torque, Condor, Cobalt

A single script may use various execution models
- Threads, pilot jobs, extreme scale

Configuration file describes how to use resources
Separation of code and execution environment

```python
from libsubmit.channels import SSHChannel
from libsubmit.providers import SlurmProvider

import parsl
from parsl.config import Config
from parsl.executors.ipp import IPyParallelExecutor
from parsl.executors.threads import ThreadPoolExecutor

config = Config(
    executors=[
        IPyParallelExecutor(
            label='midway',
            provider=SlurmProvider(
                'westmere',
                channel=SSHChannel(
                    hostname='swift.rcc.uchicago.edu',
                    username='annawoodard'
                ),
                max_blocks=1000,
                nodes_per_block=1,
                tasks_per_node=6,
                overrides='module load singularity; module load Anaconda3/5.1.0; source activate parsl_py36',
            ),
            ),
        ThreadPoolExecutor(label='local', max_threads=2)
    ],
)
parsl.load(config)
```

* Config format for Parsl 0.6

Pilot jobs on a cluster

Local threads
Extreme scale execution on Theta

4K Nodes (256K cores) for 72 hours in Singularity containers
Interactive supercomputing with Jupyter notebooks

Scalable interactive computing. Run cells, in parallel on large-scale resources

Transparent pass through of authentication tokens in JupyterHub
Parsl provides transparent (wide area) data management

Implicit data movement to/from repositories, laptops, supercomputers, ...

Globus for third-party, high performance and reliable data transfer

- Support for site-specific DTNs

HTTP/FTP direct data download/upload

parsl_file = File(globus://EP/path/file)

www.globus.org
Parsl tutorial

Running the tutorial online:
  - Binder: [https://mybinder.org/v2/gh/Parsl/parsl-tutorial/master](https://mybinder.org/v2/gh/Parsl/parsl-tutorial/master)

Running the tutorial on Blue Waters
  - Set up Parsl and download tutorial
    - module load bwpy
    - pip install --user parsl==v0.6.2-a1
    - git clone [https://github.com/Parsl/parsl-tutorial](https://github.com/Parsl/parsl-tutorial)
    - git checkout bluewaters
  - Execution options
    - Download code and run in terminal
    - **Execute notebook on Blue Waters**
      - [https://bluewaters.ncsa.illinois.edu/pythonnotebooks](https://bluewaters.ncsa.illinois.edu/pythonnotebooks)
    - Execute notebook remotely (e.g., laptop) using Blue Waters
Large-scale applications using Parsl

- Machine learning to predict stopping power in materials
- Protein and biomolecule structure and interaction
- Information extraction to discovery facts in publications
- Materials science at the Advanced Photon Source
- Cosmic ray showers as part of QuarkNet
- Weak lensing using sky surveys
- Machine learning and data analytics
Summary

Parsl takes a highly successful parallel scripting model and brings it to Python

- No porting of existing scripts to other languages
- Support for both Python and external apps
- Implicit and dynamic dataflow from data dependencies

Applied to numerous MTC and HPC application domains and used on many clusters and supercomputers

Deep integration with growing SciPy ecosystem

*Workflow through implicitly parallel dataflow is productive for applications and systems at many scales, including on highest-end systems*
Questions?

http://parsl-project.org

parsl-project.slack.com