Parsl: Productive Parallel Programming in Python

http://parsl-project.org


*University of Chicago, **Argonne National Laboratory, ***Globus, **Parallel Works

"National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign"

Motivation

- Software is increasingly assembled rather than written, using high-level languages
- New scientific methods demand many-task, parallel & distributed computing
- Python is pervasive in science and engineering
- Developers require parallel runtimes & remote execution for new science applications

Parsl aims to bring together the simplicity and high productivity possible with Python with the complex workflow patterns and extreme scale demanded by scientific computing

Why Parsl?

- Pure Python: Easily parallelize Python code
- Implicit dataflow: Apps execute concurrently while respecting data dependencies
- Write once, run anywhere: On clouds, clusters, and supercomputers

Automated data management: Implicit and flexible wide area and local staging

Python is pervasive in science and engineering

Processing with Parsl

@python_app

def pi(num_points):
    from random import random
    inside = 0
    for i in range(num_points):
        x, y = random(), random()
        if x**2 + y**2 < 1:
            inside += 1
    result = (inside / num_points)
    # App that computes the mean of three values
    @python_app
    def mean(a, b, c):
        return (a + b + c) / 3
    # Estimate three values for pi
    a, b, c = pi(10**6), pi(10**6), pi(10**6)
    # Compute the mean of the three estimates
    mean_pi = mean(a, b, c)
    # Print the results
    print("Average: {:.3f}".format(mean_pi.result()))

Features

- Resource abstraction. Block-based model overlaying different providers and resources
- Fault tolerance. Support for retries, checkpointing, and memoization
- Multi site. Combining executors/providers for execution across different resources
- Elasticity. Automated resource expansion/retraction based on workload
- Monitoring. Workflow and resource monitoring and visualization

Applications (users performing data analysis, simulation, etc.)

- AstroDEM: Satellite Image Processing
- LSST-DESC: Simulated Sky Survey
- Designing new battery materials with reinforcement learning
- High-throughput HEP on heterogeneous systems, combining ML training, simulations, inside selection, and inference

Platforms (tools on which users analyze data, simulate, etc.)

- DLHub: Machine Learning Inference
- Interactive execution of user provided machine learning models in real-time
- Comet_config
- Use arbitrary resources
- Providers for clouds, clusters, supercomputers
- Separation of code and config
- Write once, run anywhere

Education (teaching and embedding parallelism)

- Lewis & Clark College’s Campus Cluster Interface
- QuarkNet: Teaching Data Science with Cosmic Ray Data

Scaling Performance

- Scaling data on Blue Waters
- Outperforms other Python-based approaches
- Scales beyond ~2M tasks
- Easily scales to >2k nodes, with >1K tasks/s

Use arbitrary resources

- Providers for clouds, clusters, supercomputers
- Separation of code and config
- Write once, run anywhere

Parsifest 2019 attendees

- Install from PyPi or Conda Forge
- Open source (Apache 2.0 license)
- Open community (~200 GitHub stars, ~35 contributors, used by ~30 projects)
- http://parsl-project.org

Parsl is supported by the National Science Foundation under awards 1550476, 1550475, 1550528, 1550562, 1550588 and Argonne National Laboratory’s Laboratory Directed Research and Development (LDRD) program.

* Parsl”- Distinguished paper award at SC19

** Argonne National Laboratory’s Laboratory Directed Research and Development (LDRD) program

<table>
<thead>
<tr>
<th>Configuration</th>
<th>Use arbitrary resources</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Providers for clouds, clusters, supercomputers</td>
</tr>
<tr>
<td></td>
<td>Separation of code and config</td>
</tr>
<tr>
<td></td>
<td>Write once, run anywhere</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Use arbitrary resources</th>
<th>Providers for clouds, clusters, supercomputers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Separation of code and config</td>
</tr>
<tr>
<td></td>
<td>Write once, run anywhere</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Provisioning</th>
<th>Use arbitrary resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Providers for clouds, clusters, supercomputers</td>
<td>Separation of code and config</td>
</tr>
<tr>
<td>Write once, run anywhere</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Powering</th>
<th>Use arbitrary resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Providers for clouds, clusters, supercomputers</td>
<td>Separation of code and config</td>
</tr>
<tr>
<td>Write once, run anywhere</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HPC scheduler</th>
<th>Use arbitrary resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Argonne National Laboratory</td>
<td>Separation of code and config</td>
</tr>
<tr>
<td>Write once, run anywhere</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>HPC platforms</th>
<th>Use arbitrary resources</th>
</tr>
</thead>
<tbody>
<tr>
<td>Providers for clouds, clusters, supercomputers</td>
<td>Separation of code and config</td>
</tr>
<tr>
<td>Write once, run anywhere</td>
<td></td>
</tr>
</tbody>
</table>

* Parsl”- Distinguished paper award at SC19

** Argonne National Laboratory’s Laboratory Directed Research and Development (LDRD) program

<table>
<thead>
<tr>
<th>Parsifest 2019 attendees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install from PyPi or Conda Forge</td>
</tr>
<tr>
<td>Open source (Apache 2.0 license)</td>
</tr>
<tr>
<td>Open community (~200 GitHub stars, ~35 contributors, used by ~30 projects)</td>
</tr>
<tr>
<td><a href="http://parsl-project.org">http://parsl-project.org</a></td>
</tr>
</tbody>
</table>

Parsl is supported by the National Science Foundation under awards 1550476, 1550475, 1550528, 1550562, 1550588 and Argonne National Laboratory’s Laboratory Directed Research and Development (LDRD) program.