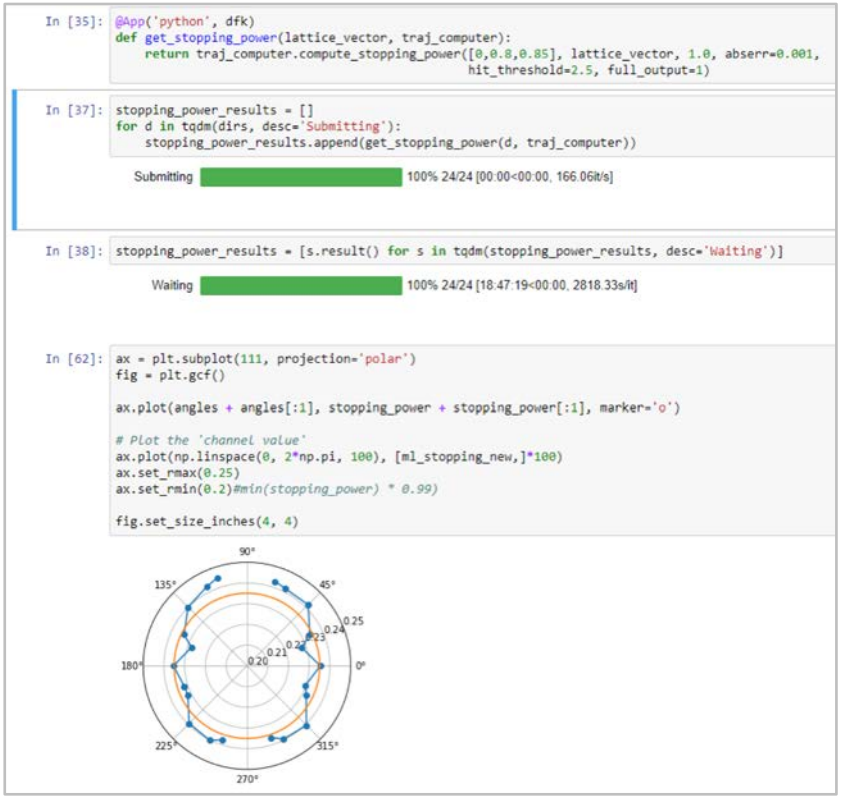
Python-based parallelism with Parsl & software sustainability

Daniel S. Katz, dskatz@illinois.edu, d.katz@ieee.org, @danielskatz
Assistant Director for Scientific Software & Applications, NCSA
Research Associate Professor, CS, ECE, iSchool

ILLINOIS
NCSA | National Center for Supercomputing Applications

An increasingly common story…

- I'm developing an application and I need to link together external tools + functions
  - (where each tool is dependent on data from the previous tool)
- I have a notebook that does *X* and I need to run it on a cloud, cluster, supercomputer
- I need to run my analysis using a range of local and distributed datasets
- I like Python
- …
- And I want to do this in an interactive environment



Parsl: Interactive parallel scripting in Python

Annotate functions to make Parsl *apps*

- Python apps call Python functions
- Bash apps call external applications

```
pip install parsl
```

Apps return “futures”: proxies for results that might not be available yet

Apps run concurrently respecting data dependencies - Natural parallel programming!

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

Based on ideas in Swift, completely reimplemented

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

print(hello().result())
```

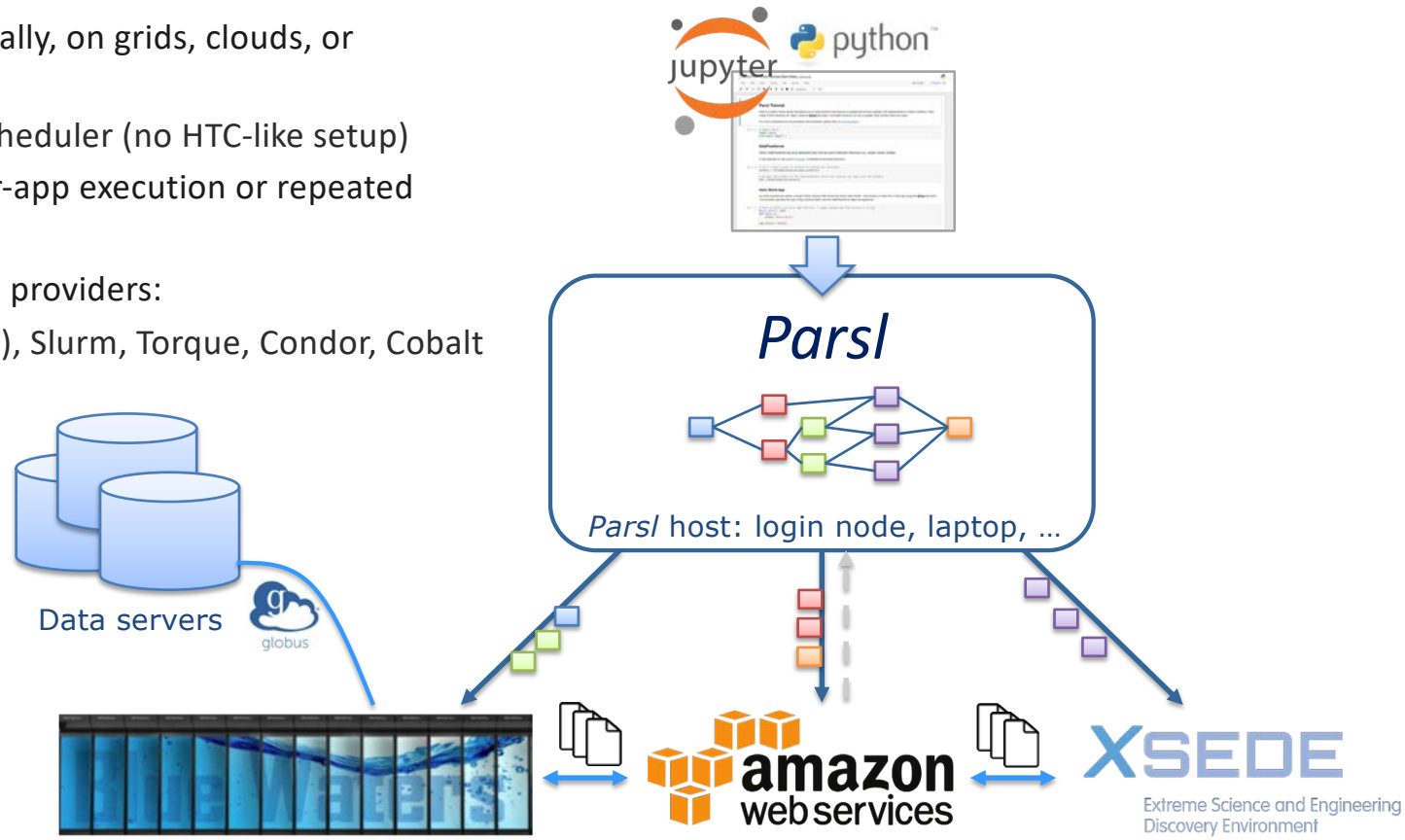
```
Hello World!
```

```bash
@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!
```

Parsl in action: dynamic dataflow execution

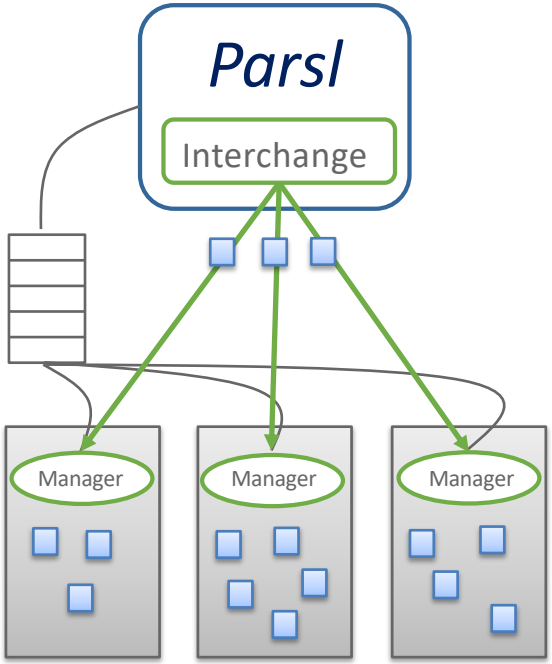
- The same script can be run locally, on grids, clouds, or supercomputers (or multiple)
  - Works directly with the scheduler (no HTC-like setup)
- Containers can be used for per-app execution or repeated invocation of the same app
- Currently supported execution providers:
  - Local, Cloud (AWS, private), Slurm, Torque, Condor, Cobalt



A variety of execution models

- High throughput (HTEX)
  - General-purpose pilot job model, 2k nodes, O(M) tasks
- Extreme scale (EXEX)
  - MPI-based pilot jobs, >250k workers on 8k Blue Waters nodes
- Low latency (LLEX)
  - Using ZeroMQ, <5 ms

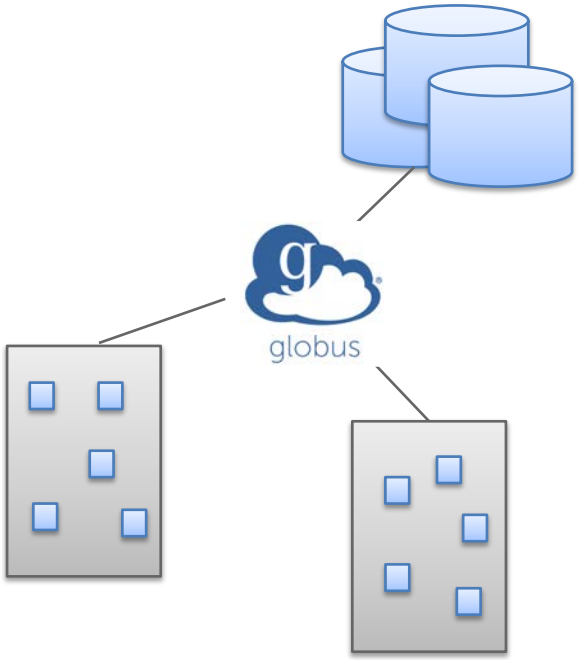
- New execution models can be added



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
- Compliments node-specific staging and caching models

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



Parsl feature summary

- Parsl's implicit dataflow model allows intuitively expressing parallelism that is then made possible at execution time via an executor
  - Expressed directly in Python
  - Can be used to implement a range of workflow models
- Parsl integrates with the scientific ecosystem
  - Development and execution of scalable applications in Jupyter
  - Use of common SciPy libraries
  - Integration with Globus
- In Parsl, code is separate from the specification of computing resources and data location: this makes Parsl scripts portable and scalable
- Parsl has a number of other important features:
  - app caching, checkpointing, elasticity, container support, data transfer, and more

Parsl project summary

- Initially funded by NSF, $3m over 3 years (stretching to 4)
- 2.5 core developer FTEs, PI, co-PIs, chemistry & education application developers, undergraduate & graduate students
- Open source, intended as open community, including library of reusable workflows
- Interesting milestones
  - First outside user
  - First outside user who didn't contact us
  - First outside contributor
  - First outside contributor who didn't contact us
- Some success with purely external contributions to code, more success with collaborating projects (e.g., DESC, Thain group @ ND)

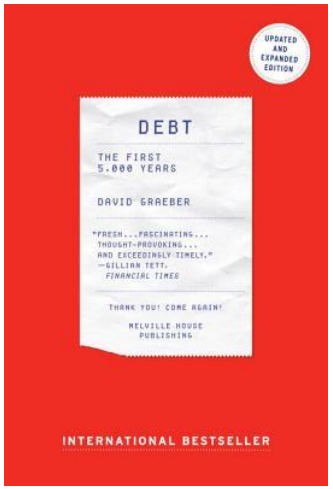
- What happens next? How we make Parsl sustainable?

Software sustainability for whom?

- (Parsl) Users
  - The capacity of the software to endure
  - Will the software (Parsl) will continue to be available in the future, on new platforms, meeting new needs?
- (Parsl) Funders
  - My definition while an NSF program officer:
  - “If I give you funds for this (Parsl) now, how will you keep it going after these funds run out?”
  - “… without coming back to me for more funds”
- (Parsl) Managers
  - Focused on people, not software
  - How do I keep the (Parsl) team going?
- (Parsl) Developers (& Maintainers)
  - Often focused on resources, not software
    - How do I get the resources needed to keep my (Parsl) software alive and up-to-date?
    - And keep myself supported / employed?
  - Counterpart
    - How do I make keeping my software alive and up-to-date use less resources?

“Equations” of software sustainability

- Software sustainability ≡ sufficient Δ software state
  - Sufficient to deal with: software collapse, bugs, new features needed
- Δ software state = (human effort in – human effort out - friction) * efficiency
  - Software stops being sustained when human effort out > human effort in over some time
- Human effort ⇆ $
  - All human effort works (community open source)
  - All $ (salary) works (commercial software, grant funded projects)
  - Combined is hard, equation is not completely true, humans are not purely rational
- Δ software state $\overset{?}{\rightarrow}$ users choose to volunteer effort or $
  - Development choices might take this into account



Debt: The First 5,000 Years
by David Graeber

Parsl sustainability methods

- Seek contributors
  - Keep code simple, well-maintained
  - Keep documentation simple, well-maintained
  - Be friendly – answer questions, encourage contributions of all kinds
  - Encourage users to contribute
- Seek support
  - Will seek new core funding
  - Adding Parsl components/effort to other proposed projects
  - Encourage SC centers to provide support (staff)
  - Explore institutional support from UC/ANL/UIUC
- Seek users/projects
  - Encourage users to talk to SC centers about Parsl
  - Encourage users to talk about Parsl in papers/press
  - Make feature/support choices based on future impact on Parsl sustainability

Summary

- Parsl as an example of an open source project
  - Started with funding and a core team, and a lot of experience (e.g. from Swift)
  - Will need to expand to be a community project and consider how to bring in new resources (funds or people)
- Software sustainability means different things to different groups of people
  - Persistence of working software
  - Persistence of people (or funding)
- Can define sustainability as
  - Inflow of resources is sufficient to do the needed work
  - Some resources can be turned into human effort
- No sustainability silver bullet
  - But many strategies that can be attempted to find resources (people, funds) and reduce needed work

Acknowledgements

- Parsl (http://parsl-project.org) team: Yadu Babuji, Kyle Chard, Ben Clifford, Ian Foster, Lukasz Lacinski, Zhuozhao Li, Connor Pigg, Michael Wilde, Anna Woodard, Justin Wozniak
- Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. Wozniak, I. Foster, M. Wilde, K. Chard, "Parsl: Pervasive Parallel Programming in Python," HPDC, 2019.
- Y. Babuji, A. Brizius, K. Chard, I. Foster, D. S. Katz, M. Wilde, J. Wozniak, "Introducing Parsl: A Python Parallel Scripting Library," 2018. https://doi.org/10.5281/zenodo.853491
- Sustainability discussions with: Neil Chue Hong and the UK SSI; Rob Haines and Caroline Jay at U. Manchester
- Sustainability keynote by James Howison at RSE2018
- Feedback from Matt Turk, James Howison, Dan Sholler
- D. S. Katz, "Scientific Software Challenges and Community Responses," 2015. https://www.slideshare.net/danielskatz/scientific-software-challenges-and-community-responses

ILLINOIS

NCSA | National Center for
Supercomputing Applications