Parsl: an example of trying to sustain research software

Daniel S. Katz (d.katz@ieee.org, @danielskatz@fosstodon.org)
Chief Scientist, NCSA
Associate Research Professor, School of Computing & Data Science, School of Information Sciences
University of Illinois at Urbana Champaign

With contributions from Parsl team & community, including
Ben Clifford, CQX Limited
Yadu Babuji, Kevin Hunter Kesling, Anna Woodard, and Kyle Chard, University of Chicago

ILLINOIS
NCSA | National Center for Supercomputing Applications

6 December 2024
NCI Informatics Technology for Cancer Research (ITCR)

https://doi.org/10.5281/zenodo.14285239

Parsl: parallel programming in Python

Apps define opportunities for parallelism
- 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 dataflow dependencies. Natural parallel programming!

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

```
pip install parsl
```

```
@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
    print(f.read())
```

```
Hello World!
```

Try Parsl: https://parsl-project.org/binder

ILLINOIS NCSA

https://doi.org/10.5281/zenodo.14285239

Parsl usage

- Parsl used by individual researchers, large research consortia, and in industry, spanning domains such as astrophysics, biology, materials science, and many others
- Impact examples: Parsl used to:
  - Produce the most interconnected simulated sky survey in preparation for analysis of the Vera C. Rubin Observatory Legacy Survey of Space and Time (LSST)
  - Conduct one of the largest single batch imputations ever performed on 474k subjects in the Million Veterans Program
  - Search for potential COVID-19 therapeutics in a search space of 4 billion candidate molecules

https://doi.org/10.5281/zenodo.14285239

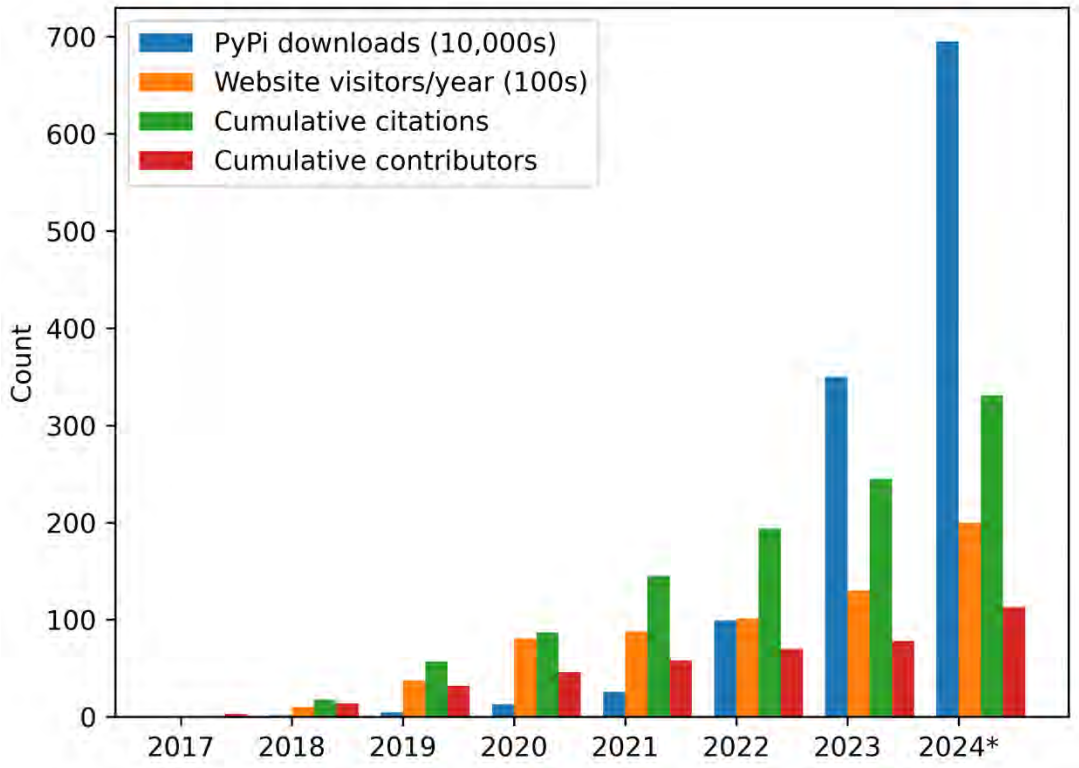
Parsl's external stakeholders

- Direct users: use Parsl for science/etc.
  - E.g., LSST DESC
- Platforms: platform developers use Parsl as a component of a platform/application used by end users
  - E.g., QC Archive, Globus Compute (was funcX)
- Cyberinfrastructure providers: support Parsl on their HPC/etc. system
  - E.g., Argonne, NERSC
- Linked contributors: link naturally complementary components with Parsl
  - E.g., Parsl provides interesting ways to describe related tasks, and Work Queue provides interesting ways to schedule those tasks: WorkQueue -> Parsl WorkQueue executor
- Funders
  - E.g., NSF, CZI, DOE, collaborating projects

https://doi.org/10.5281/zenodo.14285239

Parsl history

- Initially supported by an NSF SI2 award from 2016-2022 (5 years + 1-year NCE)
- Released version 1.0 in 2020, now releasing weekly
  - Focus since v1.0 mostly maintenance rather than adding new features
- Initial funded development team
  - 2-4 people/FTEs per year
- Current funding (new NSF award, CZI award, contributions from projects that require Parsl) supports
  - ~1 FTE/year maintenance and development
  - ~0.5 FTE/year community management



* Through ~1 Aug 2024

https://doi.org/10.5281/zenodo.14285239

Sustainability: Balancing resources & work

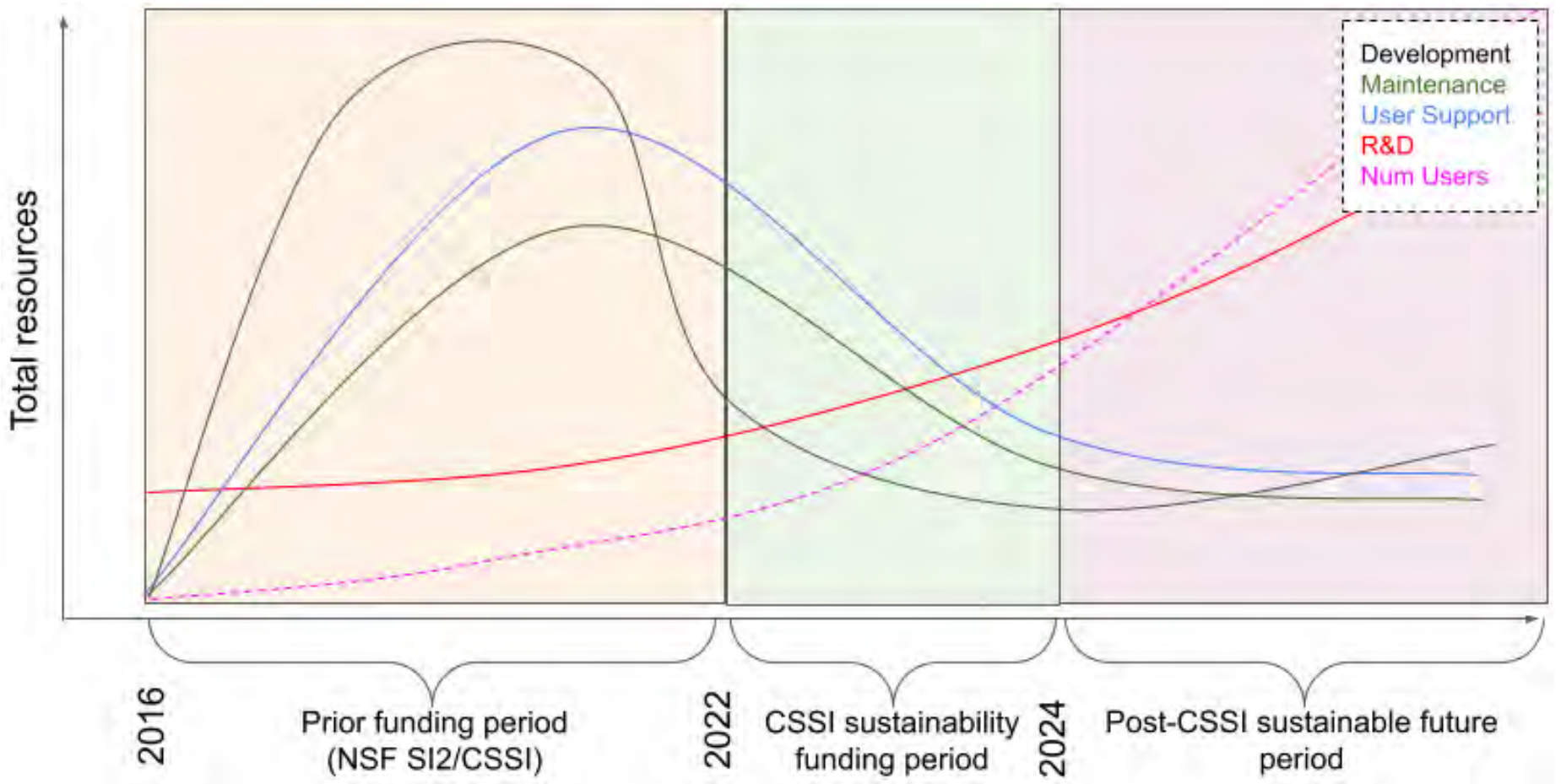
- Parsl resources
  - Grants
  - External funding from projects that depend on Parsl
  - Volunteer (in-kind) effort from groups that develop tools that use Parsl
  - Companies that use Parsl in their services
- Funded Parsl team does the core work, such as
  - Managing the community
  - Reviewing code contributions
  - Fixing bugs
  - Supporting users
  - Developing new features
  - Releasing new versions of the software
- Provided/volunteers resources can add features to Parsl and support some limited number of use cases
  - But aren’t currently sufficiently coordinated or aligned to fully support Parsl's core needs over multiple years

https://doi.org/10.5281/zenodo.14285239

Past, current, and sustainable resources

Current NSF & CZI awards aim at making Parsl *sustainable*:

Resources (from various sources) will be able to perform all project activities well into the future



https://doi.org/10.5281/zenodo.14285239

Path to sustainability

- At a high level, we want to follow the successful sustainability model of AstroPy, yt, etc.
- Work on community, governance, funding streams, innovation, training, outreach/engagement
- Work with other related community (software sustainability, workflows)
- Capture, document, and share sustainability lessons

- Also reduce technical debt
  - The easier it is to do things, the less resources are needed to do them

https://doi.org/10.5281/zenodo.14285239

Project stages

https://doi.org/10.5281/zenodo.14285239

Parsl coding

- Parsl started as an idea in 2016, based on previous project Swift (http://swift-lang.org)
  - Basically asked if we wanted to do the same thing – a simple tool (language/runtime) for fast, easy scripting on big machines – today, what would we do
  - One person did some exploration/proof-of-concept – it worked
    - w/ 4 people managing and "helping"
  - Build the initial usable system, was the main developer, did things the way they wanted
  - Once a second developer became active, needed to agree on and define/document processes
  - As we moved to a more open community project (2-4 FTEs/year of developers funded, and 73 contributors), these processes became more important

https://doi.org/10.5281/zenodo.14285239

Parsl processes

- How to make design/architecture decisions?
- What code style to use?
- What testing is sufficient?
- What documentation is sufficient?
- How to engage with and support users?
- What properties do contributions and changes need to have?
- How do contributions and changes get accepted?
- How to encourage/develop contributors?
- How to mix CS research and software product development?
- Who writes papers and who is listed as co-authors?

- All of the answers have changed over the life of the project

https://doi.org/10.5281/zenodo.14285239

Changing developer work

- Current needs (e.g., maintenance, outreach, and support) differ from earlier in the project, leading to a need for new types of contributions
- Development activities include
  - Maintain the Parsl codebase
    - Including adding additional tests to improve code quality
  - Respond to issues, including supporting deployment on different computing resources
    - As community grows and diversifies, range of use cases and range of challenges also grow
  - Review contributed code
    - Leads us to develop minimal requirements on contributed code, starting with a pre-coding discussion and including plans for future maintenance and support of contributed code

https://doi.org/10.5281/zenodo.14285239

Internal Parsl developer roles

1. Research programmer
   - Main job: prototype ideas very quickly
   - Focus on what the software can do, more than how users will use it
   - Essential to bootstrap new research software project, develop & test initial ideas; might take shortcuts that harm project's later sustainability (adding technical debt)
   - Stage: Initial development & new features in later stages
2. Software developer
   - Main job: development professional-class research software
   - Focus on software itself & its users
   - Dedicated to making the software as useful as possible; making it clean & beautiful; increasing simplicity, compatibility, future maintainability; reducing technical debt
   - Stage: Important to have involved in all but the initial stage of the project (where process may impede fast innovation)

https://doi.org/10.5281/zenodo.14285239

Internal/external Parsl developer roles

3. User/developer
   - Main job: a scientist or disciplinary researcher who also adds features relevant to their work
   - Focus on their own usage of the software
   - Typically write code elsewhere for their real job (research); not dedicated to Parsl development, but a power user of Parsl who can understand bugs and fix them
   - May take shortcuts that harm project's future sustainability
   - Stage: all but initial stage
4. Collaborating developer
   - Main job: responsible for a collaborative project
   - Focus on their software working with or being "part of" Parsl
   - In their own project, can may be any other developer type; may follow different coding/software engineering style, potentially leading to integration and culture challenges
   - Defining the interface to these developers is a key challenge for sustainability, as ideally, they will become committed to Parsl's success as important to their own project's
   - Stage: most helpful after the project has become initially proven and has demonstrated some success (stages 3 & 4)

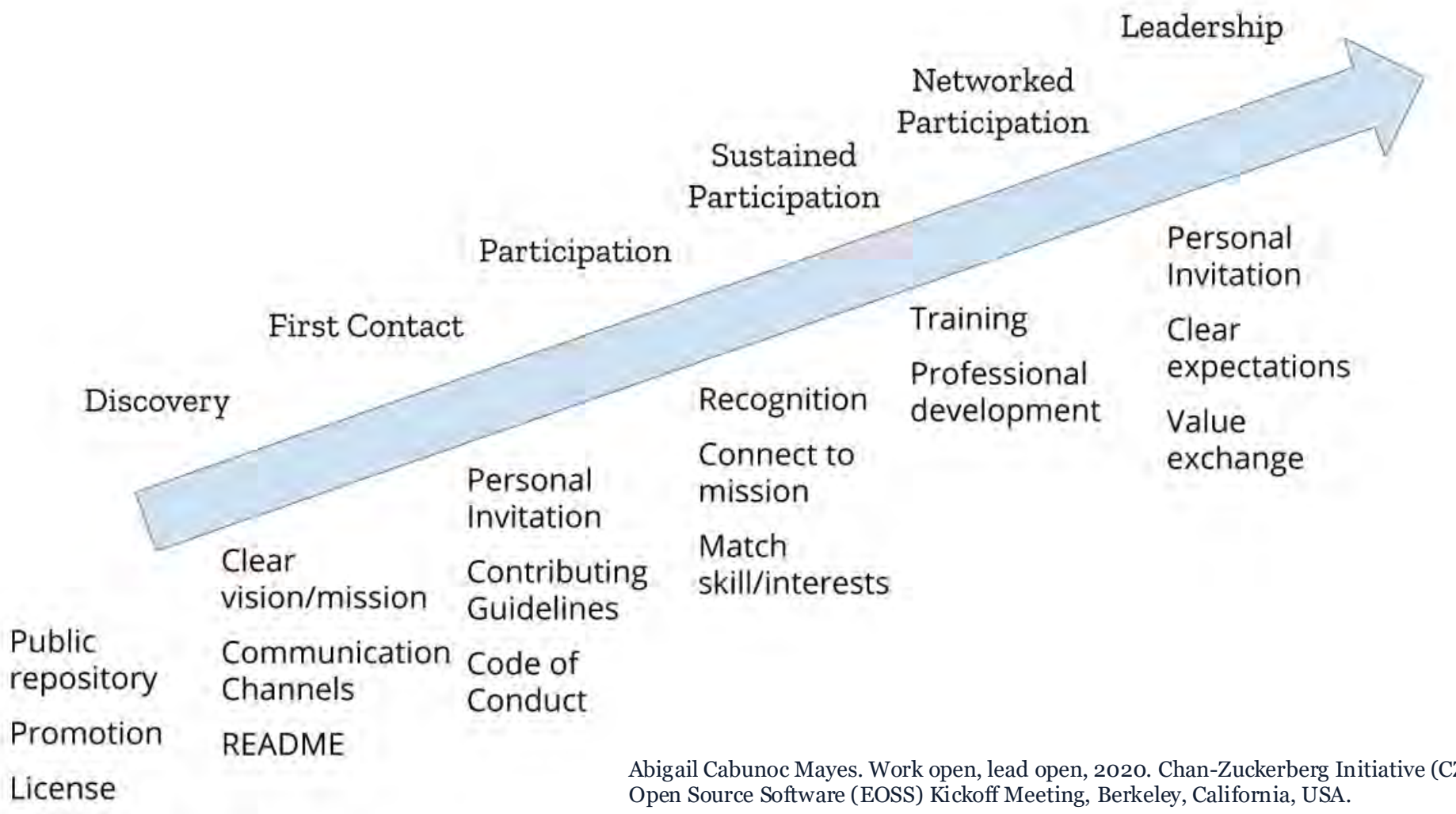
https://doi.org/10.5281/zenodo.14285239

Changing community work

- In addition to developers, members of the community sometimes do (kind of in order of where community contributions actually happen)
  - Answer user questions
  - Share experiences (configs) for specific computing platforms
  - Share expertise in applying Parsl to different scholarly disciplines
  - Support outreach activities (e.g., presenting tutorials, hosting summer students, developing training materials for various domains)
  - Coordinate the yearly user meeting
  - Apply for funding
  - Advertise success stories in blogs
  - Manage social media
- Parsl community manager does/coordinates this

https://doi.org/10.5281/zenodo.14285239

Community



Abigail Cabunoc Mayes. Work open, lead open, 2020. Chan-Zuckerberg Initiative (CZI) Essential Open Source Software (EOSS) Kickoff Meeting, Berkeley, California, USA.

https://doi.org/10.5281/zenodo.14285239

NumFOCUS

- Parsl joined NumFOCUS in Nov 2024
- Process involved applying, being accepted, then signing a financial sponsorship agreement
- U Illinois agreed to transfer ownership of its IP to its employees to transfer it to NumFOCUS after a short period; U Chicago took about 9 months to discuss and agree
- We see this as part of our sustainability plan
  - Ownership of the project in a neutral place to encourage others to take on leadership and governance of the project
  - A mechanism to hire staff outside the US when needed, and to contract for specific work items to the best available person, regardless of their affiliation

https://doi.org/10.5281/zenodo.14285239

Where we are now

- Good news
  - Community growth – at least in part due to community manager
    - Lots of contributors
    - Lots of users
  - Parsl code has gotten better – at least in part due to core maintainer
    - Moving to more plug ins to reduce what the core code has to do
    - Removing old code that isn't used, doesn't work, etc.
    - More and better tests
- Less good news
  - Unclear how to sustain community manager and core developer
    - This core community and maintenance work is hard/impossible to rely on volunteers to do, at least for a project of Parsl's size

https://doi.org/10.5281/zenodo.14285239

Lessons

- Sometimes choices are just choices, made for the sake of having made a choice – once made, these can be hard to change, but changes should be considered regularly
- Going from one to two developers is a big step, and an opportunity to consider changes
- Research software sustainability is a hard problem with no simple answers
- The existence of different types of developers (RSEs) and their utility during different phases of Parsl, particularly when moving from a project funded by a grant to one supported by a mix of sources, emerged during the project
- This seems to match other projects' experiences, but isn't really tested yet
- Boundaries between types of people and roles (developer, RSE, community member, user) are fuzzy

https://doi.org/10.5281/zenodo.14285239

Acknowledgements

- Parsl has been supported by NSF (1550588, 2209919, 2209920) and the Chan Zuckerberg Initiative, and DOE, collaborating projects, and a community of developers, maintainers, and users

https://doi.org/10.5281/zenodo.14285239