SC24
Atlanta, GA | hpc creates.

Establishing a High-Performance and Productive Ecosystem for Distributed Execution of Python Functions Using Globus Compute

Rachana Ananthakrishnan,[1] Yadu Babuji,[1] Josh Bryan,[1] Kyle Chard,[12] Ryan Chard,[2] Ben Clifford,[3] Ian Foster,[12] Lev Gorenstein,[1] Kevin Hunter Kesling,[1] Chris Janidlo,[1] Daniel Katz,[4] Reid Mello,[1] **J. Gregory Pauloski**,[12] Lei Wang[1]

[1]*University of Chicago,* [2]*Argonne National Laboratory,* [3]*Hawaga,* [4]*University of Illinois Urbana-Champaign*

22 November 2024 — Atlanta, Georgia

THE UNIVERSITY OF CHICAGO

globus

Globus is …

a non-profit service developed and operated by

THE UNIVERSITY OF CHICAGO

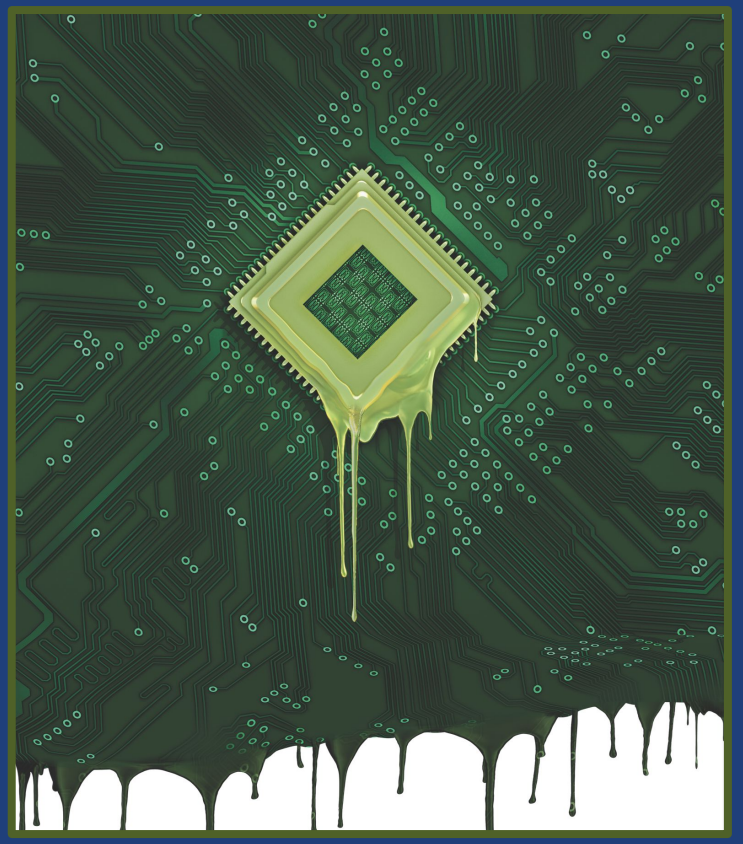
Our mission is to…

increase the efficiency and effectiveness of researchers engaged in data-driven science and scholarship through *sustainable* software

Some context…

General Purpose Computing is morphing…

"…the economic cycle that has led to the usage of a common computing platform, underpinned by rapidly improving universal processors, is giving way to **a fragmentary cycle, where economics push users toward divergent computing platforms driven by special purpose processors.**"



"The Decline of Computers as a General Purpose Technology",
Thompson, N. & Spanuth, S., Communications of the ACM, March 2021

Our data management legacy is morphing…

- From fast, reliable, data transfer …
- … to secure data sharing …
- … and data management automation at scale
- But research flows inevitably include computation

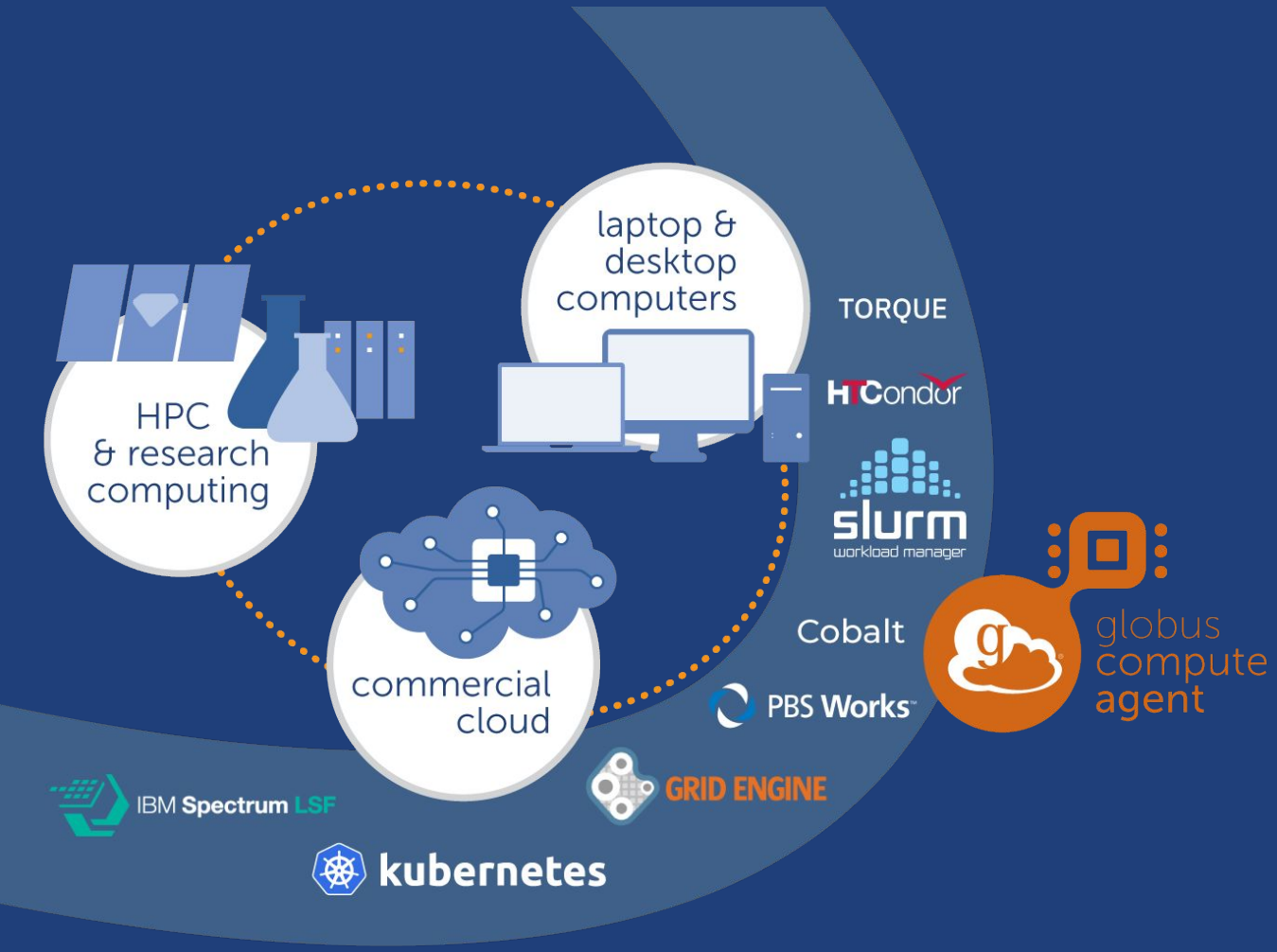
Deliver the same “fire-and-forget” capabilities for computation as we do for data management

In the traditional model of remote computing…

- **Figure out authentication (across multiple domains)**
- **Establish and maintain the right network connections (e.g., SSH)**
- **Interact with resources (configure for job scheduler, wait in queues, scale nodes)**
- **Configure execution environments**
- **Detect, understand, and recover from various failures**
- **…**

Researchers need to overcome the same obstacles every time they move to a new resource

What is Globus Compute?



- **FaaS for any compute resource**
- **Programmatic access to compute resources**
- **“Fire and forget” reliable execution**
- **Consistent user interface across diverse execution systems**

What is Globus Compute?

- **Compute service — Highly available cloud-hosted service for managed function execution**
- **Compute endpoint — Abstracts access to compute resources, from edge device to supercomputer**
- **Compute SDK — Python interface for interacting with the service**

What does it look like to a researcher?

Turn any resource into a FaaS endpoint

- **Python `pip` installable agent**
- **Elastic resource provisioning from local, cluster, or cloud system (via Parsl)**
- **Parallel execution using local fork or via common schedulers**
  - Slurm, PBS, LSF, Cobalt, K8s

Configuring a Globus Compute Endpoint

```
$ pip install globus-compute-endpoint
$ globus-compute-endpoint configure my-endpoint

Created profile for endpoint named <my-endpoint>

    Configuration file: /home/greg/.globus_compute/my-endpoint/config.yaml

Use the `start` subcommand to run it:

    $ globus-compute-endpoint start my-endpoint
```

```
$ globus-compute-endpoint start my-endpoint
Starting endpoint; registered ID: 54460200-b652-4f43-a918-02882fa6114a
```

Configuring a Globus Compute Endpoint

```yaml
engine:
  max_workers_per_node: 1
  type: GlobusComputeEngine
  provider:
    type: LocalProvider
    init_blocks: 1
    max_blocks: 1
    min_blocks: 0
```

…/my-endpoint/config.yaml

```yaml
engine:
  type: GlobusComputeEngine
  max_workers_per_node: 4
  available_accelerators: 4

  provider:
    type: PBSProProvider

    launcher:
      type: MpiExecLauncher
      bind_cmd: --cpu-bind
      overrides: --depth=64 --ppn 1

    account: {{ YOUR_POLARIS_ACCOUNT }}
    queue: debug-scaling
    cpus_per_node: 32
    select_options: ngpus=4
    scheduler_options: "#PBS -l filesystems=home:grand:eagle"
    walltime: 01:00:00
    nodes_per_block: 1
    init_blocks: 0
    min_blocks: 0
    max_blocks: 2
```

← 1 Worker/GPU

← PBS Batch Job

← 1 Manager/Node

← PBS Job Params

Executing workloads with Globus Compute

- **Invoke Python functions as tasks**
  - Select endpoint
  - Define (optionally register) function
  - Execute task with input arguments
- **Globus Compute stores tasks in the cloud**
- **Endpoints fetch waiting tasks (when online), run the task, and return the results (or errors)**
- **Results stored in the cloud; users retrieve results asynchronously**

Instantiate the Globus Compute Executor

```
from globus_compute_sdk import Executor

endpoint_id = "4b116d3c-1703-4f8f-9f6f-39921e5864df"
gce = Executor(endpoint_id=endpoint_id)
```

Execute a function on the remote endpoint

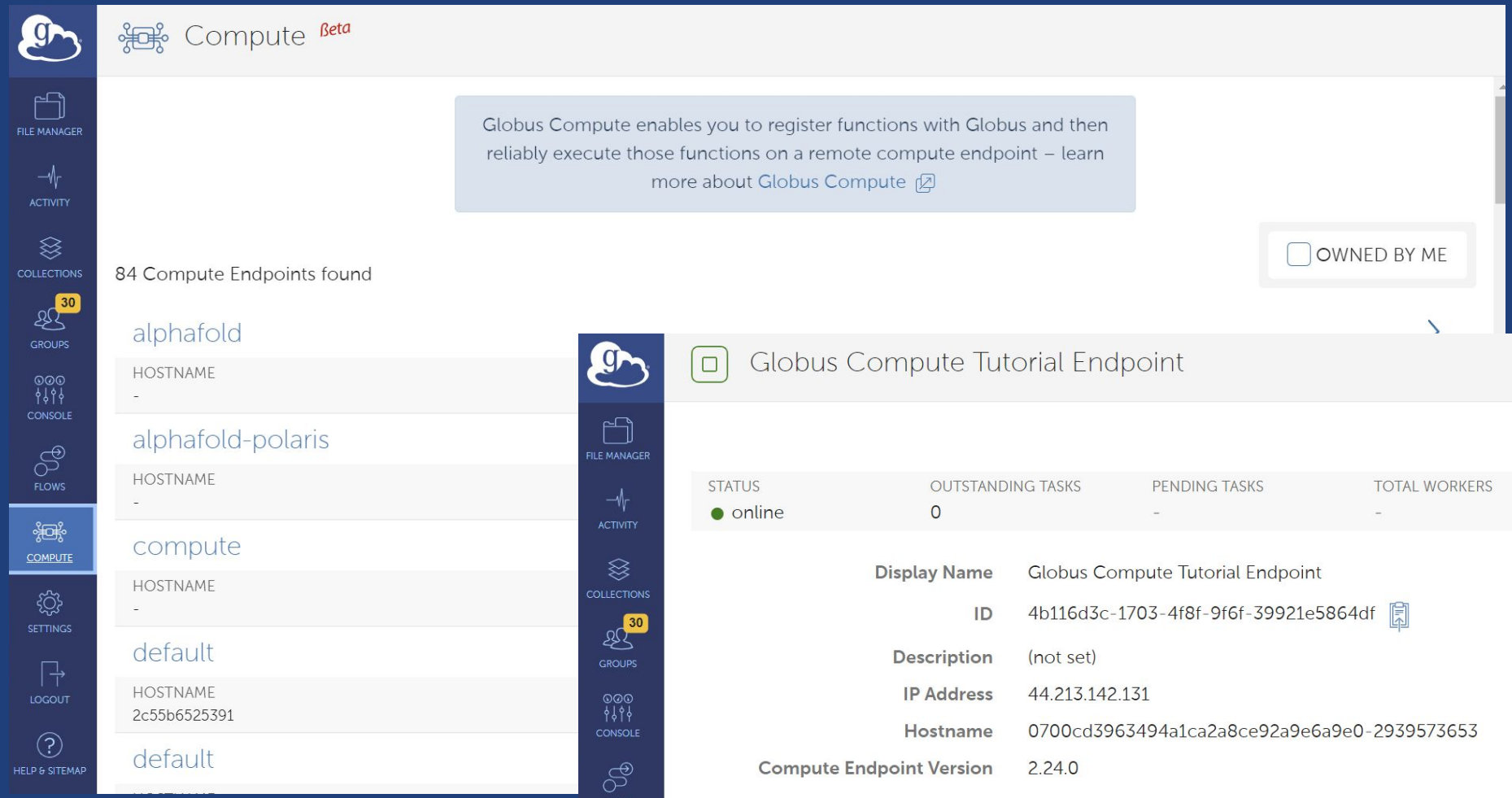
```
def hello_world(name):
    return f"Hello {name}"

future = gce.submit(hello_world, "World")
```

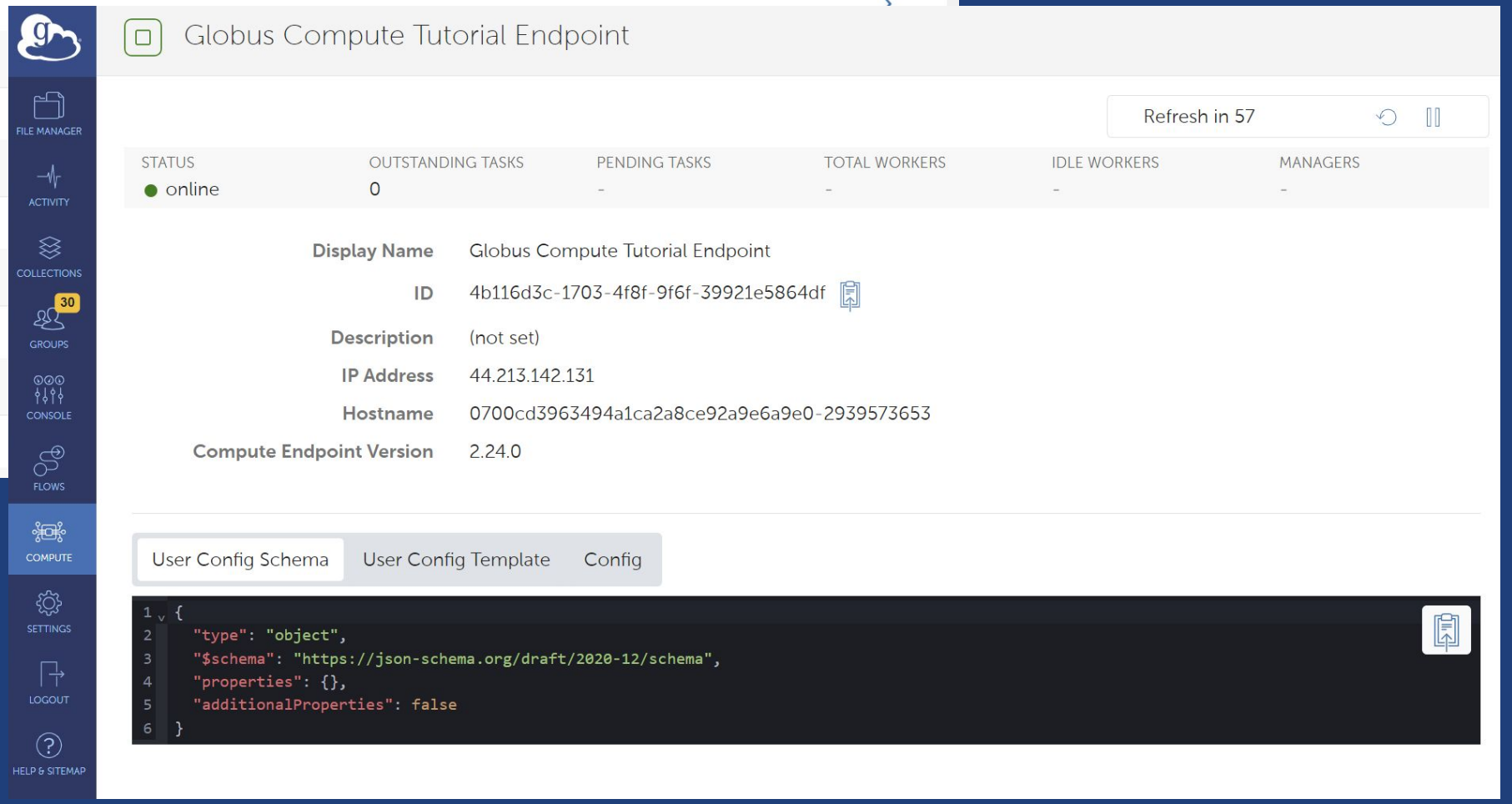
Retrieve the results

```
print(future.result())   # Hello World
```

https://jupyter.demo.globus.org/hub/user-redirect/lab/tree/globus-jupyter-notebooks/Compute_Introduction.ipynb

Discovering & monitoring endpoints

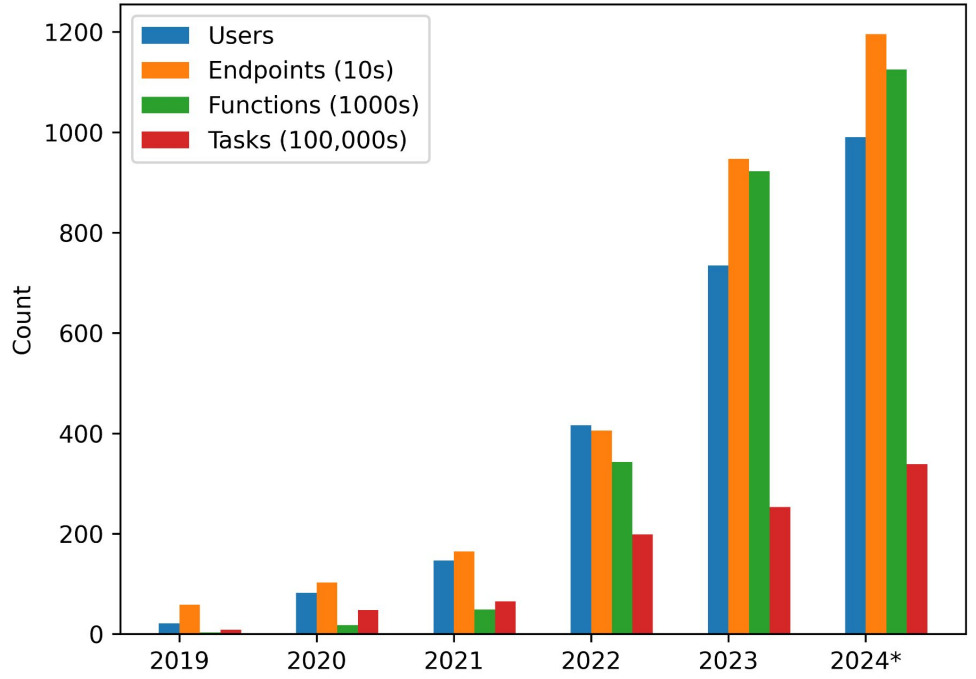




A fast growing user base

Biggest users in one of three categories:

- **Remote (bag-of-tasks) execution**
- **Research automation**
- **Platform for building other services**



>35M tasks, >1M functions, >12K endpoints

This paper is about…

eliminating barriers of use through new features in Globus Compute

Four existing challenges

1. **Executing shell commands**
2. **Invoking MPI codes**
3. **Multi-user Endpoints**
4. **Out-of-band data transfer**

Shell Commands

Globus Compute is used for more than just Python…

- **Users often write functions that call other languages, scripts, and binaries**
- **Easier to integrate into scripts than invoking commands over SSH**

Shell Commands

`ShellFunction`

- Better abstraction for command execution
- Per-task sandboxing
- Per-task walltime limits
- Captures stdout, stderr, & return code

```python
from globus_compute_sdk import Executor
from globus_compute_sdk import ShellFunction

# Command is formatted with kwargs when invoked
sf = ShellFunction("echo '{message}'")

with Executor(endpoint_id="...") as ex:
    for msg in ["hello", "hola", "bonjour"]:
        future = ex.submit(sf, message=msg)
        shell_result = future.result()
        print(shell_result.stdout)
```

```python
sf = ShellFunction("sleep 2", walltime=1)
future = executor.submit(sf)
print(future.returncode)  # Returns 124
```

Invoking MPI Codes

Shell commands but harder…

Want to support heterogenous MPI app shapes in single batch job

Invoking MPI Codes

`MPIFunction`

- Extends ShellFunction (output & sandboxing)
- Resource specification (nodes & ranks per node)
- Uses MPI launcher to execute MPI apps
- Uses GlobusMPIEngine

```
from globus_compute_sdk import MPIFunction

func = MPIFunction("hostname")

for n in range(1, 2):
    print(f'Ranks per node{n}')
    executor.resource_specification = {
        "num_nodes": 2,
        "ranks_per_node": n,
    }
    future = executor.submit(func)
    mpi_result = future.result()
    print(mpi_result.stdout)
```

```
Ranks per node: 1
exp-14-08
exp-14-20
Ranks per node: 2
exp-14-08
exp-14-20
exp-14-08
exp-14-20
```

Invoking MPI Codes

`GlobusMPIEngine`

- MPI-aware version of default GlobusEngine
- Dynamically partition batch job based on user-defined task requirements

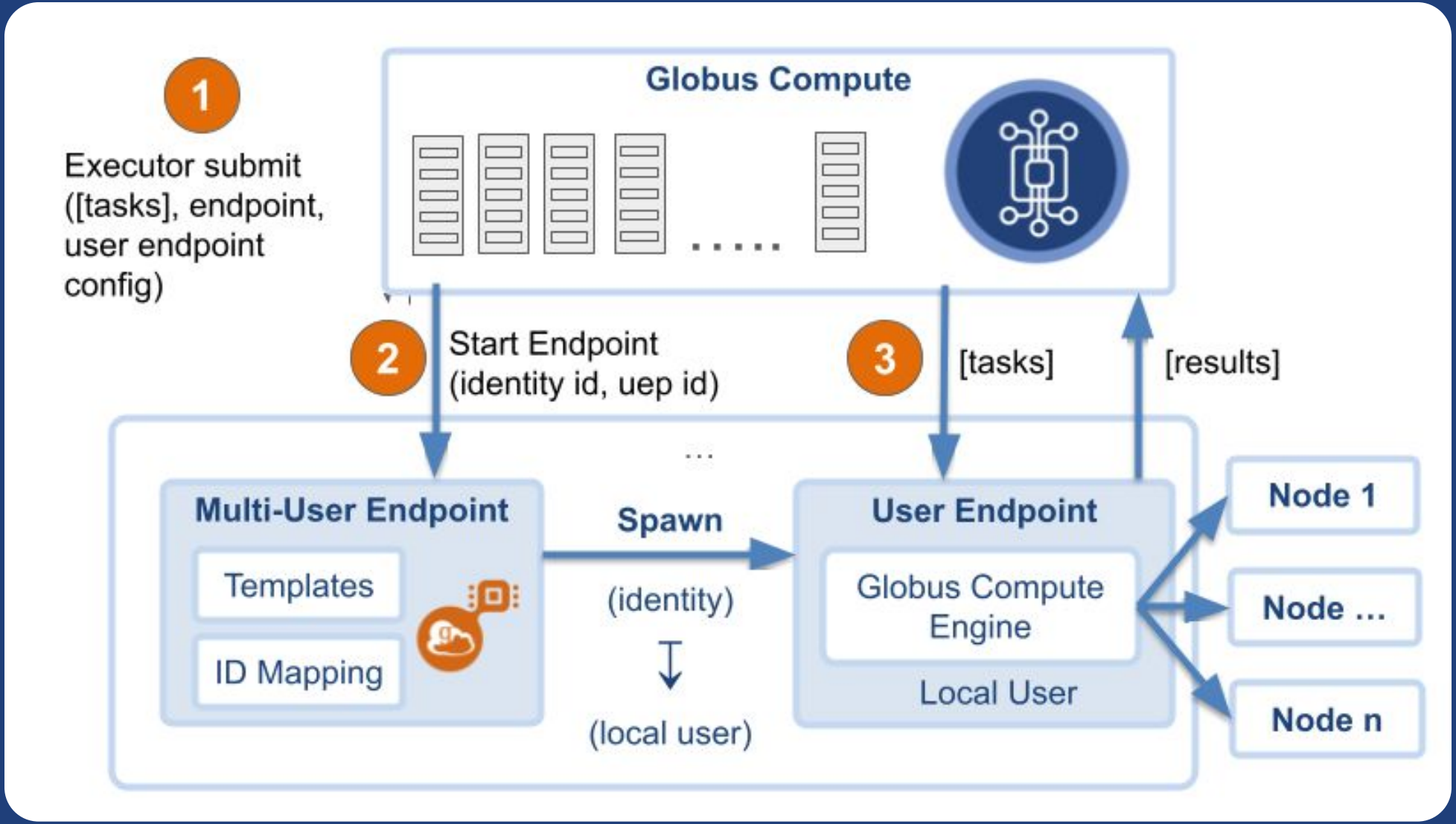
```yaml
# Configuration for a Slurm based HPC system
display_name: SlurmHPC
engine:
  type: GlobusMPIEngine
  mpi_launcher: srun

  provider:
    type: SlurmProvider

    launcher:
      type: SimpleLauncher

    # Specify # of nodes per batch job that
    # will be shared by multiple MPIFunctions
    nodes_per_block: 4
```

Multi-User Endpoints

Single-user endpoints…

- Accessible only by user that created them
- Static configuration: reconfiguration requires restart
- Users run many endpoints on a single resource
- Difficult for administrators to provide support

Multi-User Endpoints



Multi-User Endpoints — Admin Perspective

1. **Installation**
   a. Available on pip, RPM, and Deb repositories
   b. `globus-compute endpoint configure {name} --multi-user`
2. **Identity Mapping: Globus User to local UID?**
   a. Default mapping for single domain users
   b. Pattern-based mappings
   c. Callouts to external scripts/programs (e.g., query database or LDAP).
3. **Template Configuration**

Multi-User Endpoints — Admin Perspective

Admin-defined Jinja templates

```
engine:
  type: GlobusComputeEngine
  nodes_per_block: {{ NODES_PER_BLOCK }}

  provider:
    type: SlurmProvider
    partition: cpu
    account: {{ ACCOUNT_ID }}
    walltime: {{ WALLTIME|default("00:30:00") }}

    launcher:
      type: SrunLauncher
```

- User required values and default values
- Optional property schema for guiding users
- Useful to users—define endpoint parameters from your code
- Optional permitted functions list for admin-created functions

Multi-User Endpoints — User Perspective

1. **Find an endpoint UUID:** via resource-specific documentation or Globus Compute web app
2. **Run your code:**

```
uep_conf = {
  "NODES_PER_BLOCK": 64
  "ACCOUNT_ID": "314159265",
  "WALLTIME": "00:20:00"
}

with Executor(endpoint_id="<UUID>") as gce:
    gce.user_endpoint_config = uep_conf
    fut = gce.submit(hello_world)
    res = fut.result()
```

Multi-User Endpoints — Goals

1. **Lower Barriers of Use:** Shift configuration complexities to administrators
2. **Improved Access Control:** Administrators have granular control over user access permissions and resources
3. **Efficient Resource Utilization:** Optimize endpoint configurations for common classes of users
4. **Improved User Experience:** Users no longer need to manage endpoints

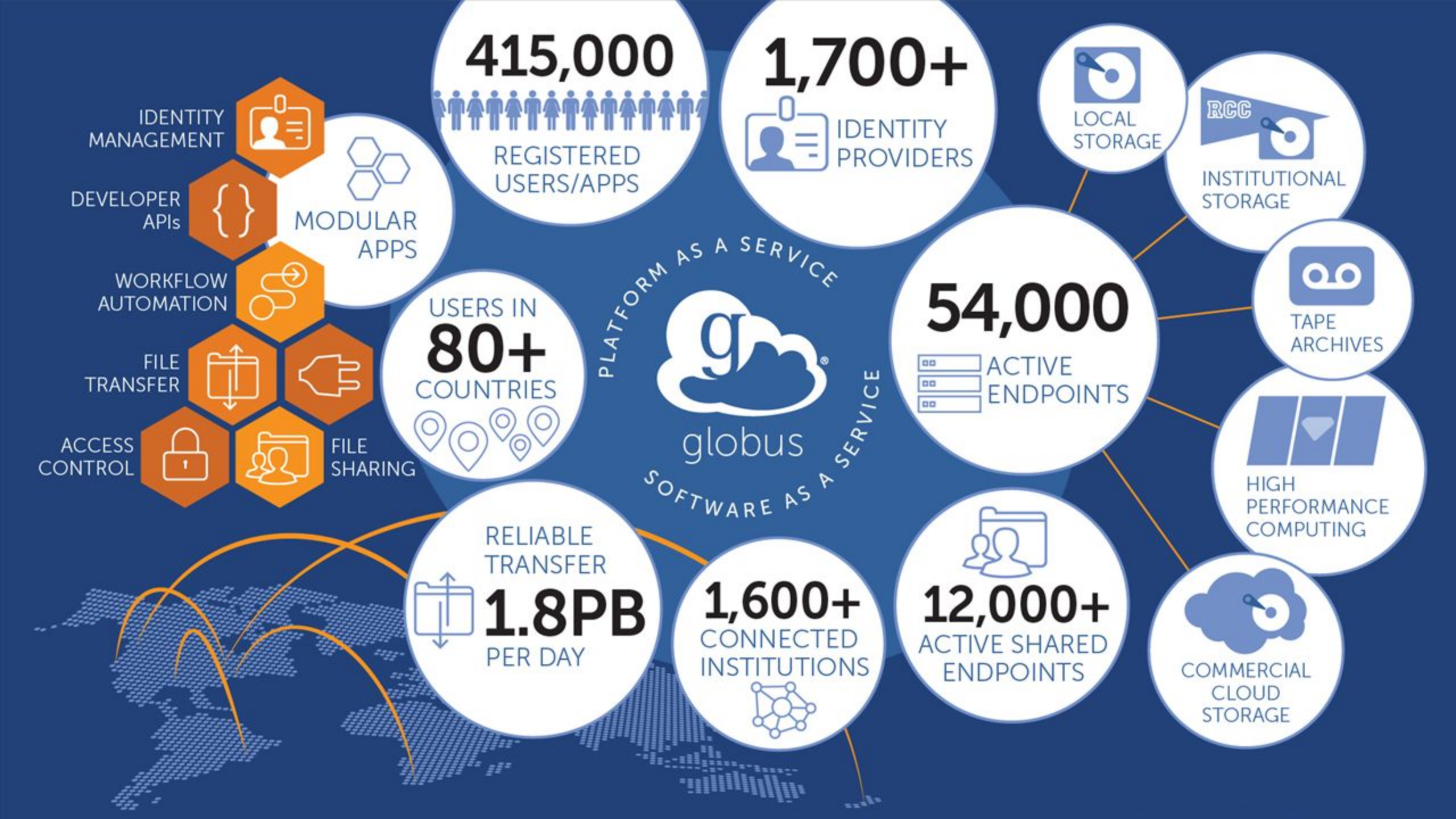
Out-of-band Data Transfer

Globus Compute API has a few limitations:

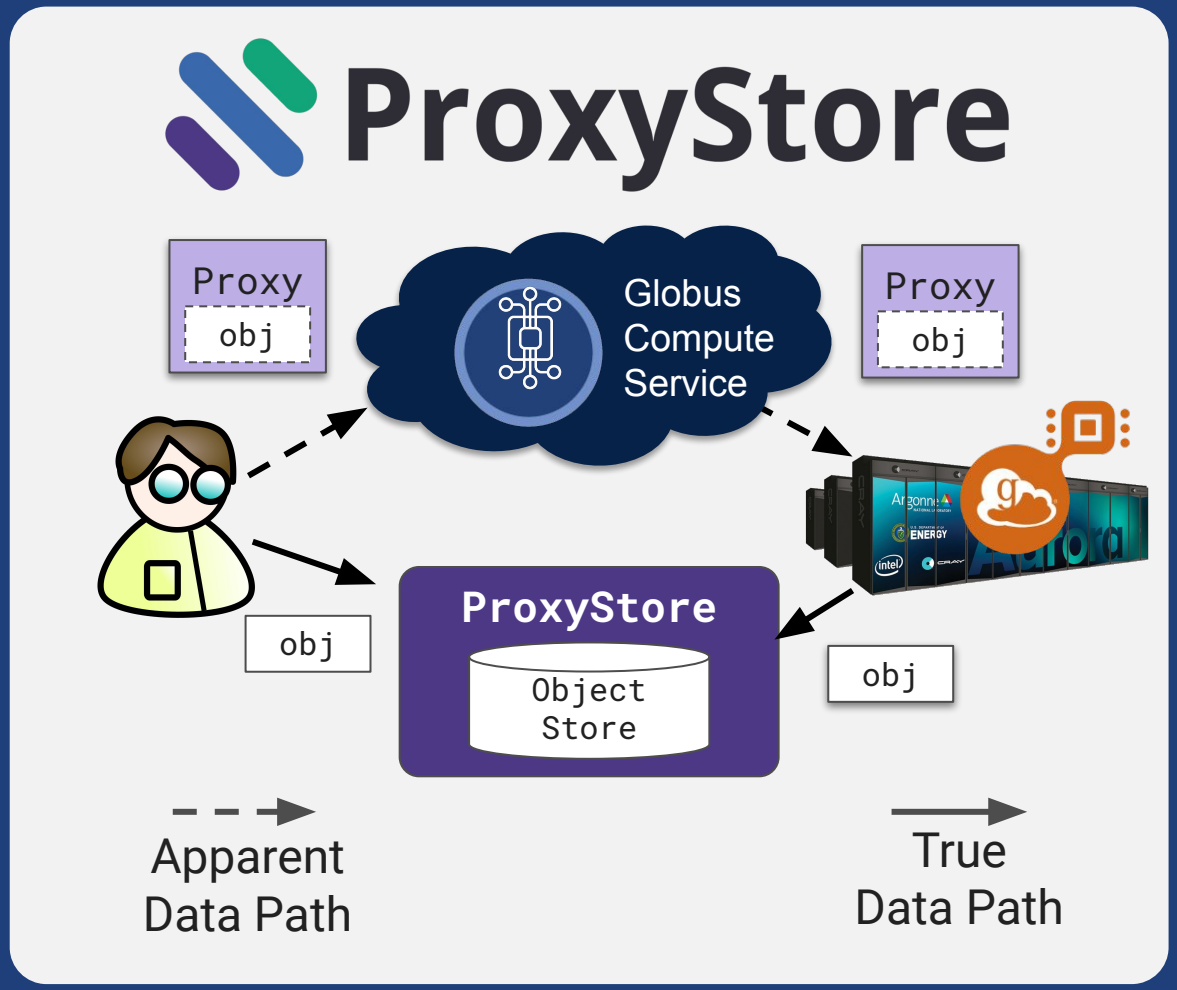
- Rate limiting (20 requests/10 seconds)
- Task TTL (2 weeks)
- Data Limits (10 MB each for payload and result)

We describe two solutions:

- Globus Transfer
- ProxyStore


415,000
REGISTERED USERS/APPS
1,700+
IDENTITY PROVIDERS
IDENTITY MANAGEMENT
DEVELOPER APIs
MODULAR APPS
WORKFLOW AUTOMATION
FILE TRANSFER
ACCESS CONTROL
FILE SHARING
USERS IN 80+ COUNTRIES
PLATFORM AS A SERVICE
globus
SOFTWARE AS A SERVICE
LOCAL STORAGE
RCC
INSTITUTIONAL STORAGE
54,000
ACTIVE ENDPOINTS
TAPE ARCHIVES
HIGH PERFORMANCE COMPUTING
RELIABLE TRANSFER
1.8PB
PER DAY
1,600+
CONNECTED INSTITUTIONS
12,000+
ACTIVE SHARED ENDPOINTS
COMMERCIAL CLOUD STORAGE

Out-of-band Data Transfer



Python library for distributed data flow management

- Represent and efficiently move objects in federated applications
- Object proxy: ***pass-by-reference*** and ***pass-by-value***

Out-of-band Data Transfer

Why pass-by-proxy?

- Proxy is self-contained = No changes to function code
- Avoid 10 MB payload limit
- Use more performant transfer method (TCP, RDMA, Object Stores, P2P)

```python
from globus_compute_sdk import Executor
from proxystore.connectors import UCXConnector
from proxystore.store import Store

def foo(value: Bar) -> None:
    assert isinstance(value, Bar)

with Executor(endpoint_id="<UUID>") as gce:
    with Store('demo', UCXConnector(...)) as store:
        value = Bar(...)
        proxy = store.proxy(value)
        gce.submit(foo, proxy)
```

https://docs.proxystore.dev/main/guides/globus-compute/

Summary

Globus Compute: “Fire & Forget” computations

Reduce barriers of access through:

- **Native execution of shell commands**
- **Integrated MPI support**
- **Multi-user Endpoints**
- **Out-of-band data transfer mechanisms**

Thank you, funders...

U.S. DEPARTMENT OF
ENERGY

THE UNIVERSITY OF
CHICAGO

NIST
National Institute of
Standards and Technology
U.S. Department of Commerce

Argonne
NATIONAL LABORATORY

Any questions?

Docs: https://globus-compute.readthedocs.io/en/latest/
GitHub: https://github.com/globus/globus-compute
Slack: https://funcx.slack.com/