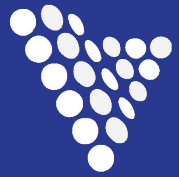


ParslFest 2023

Globus Compute Executor

Reid Mello - reid@globus.org

THE UNIVERSITY OF CHICAGO

Argonne NATIONAL LABORATORY

The Executor class

Part of the Globus Compute SDK

```
from globus_compute_sdk import Executor
```

Preferred approach to submitting tasks and collecting results

```
with Executor(endpoint_id="...") as gce:
    fut = gce.submit(func, *args, **kwargs)
    print(fut.result())
```

Subclass of concurrent.futures.Executor

- Typically used as a context manager
- `.submit()` returns a Future
- Automatically wait for results

The ComputeFuture class

Subclass of [concurrent.futures.Future]

- `.result()` waits until upstream services return the result
- `.done()` returns a boolean indicating whether the result is ready

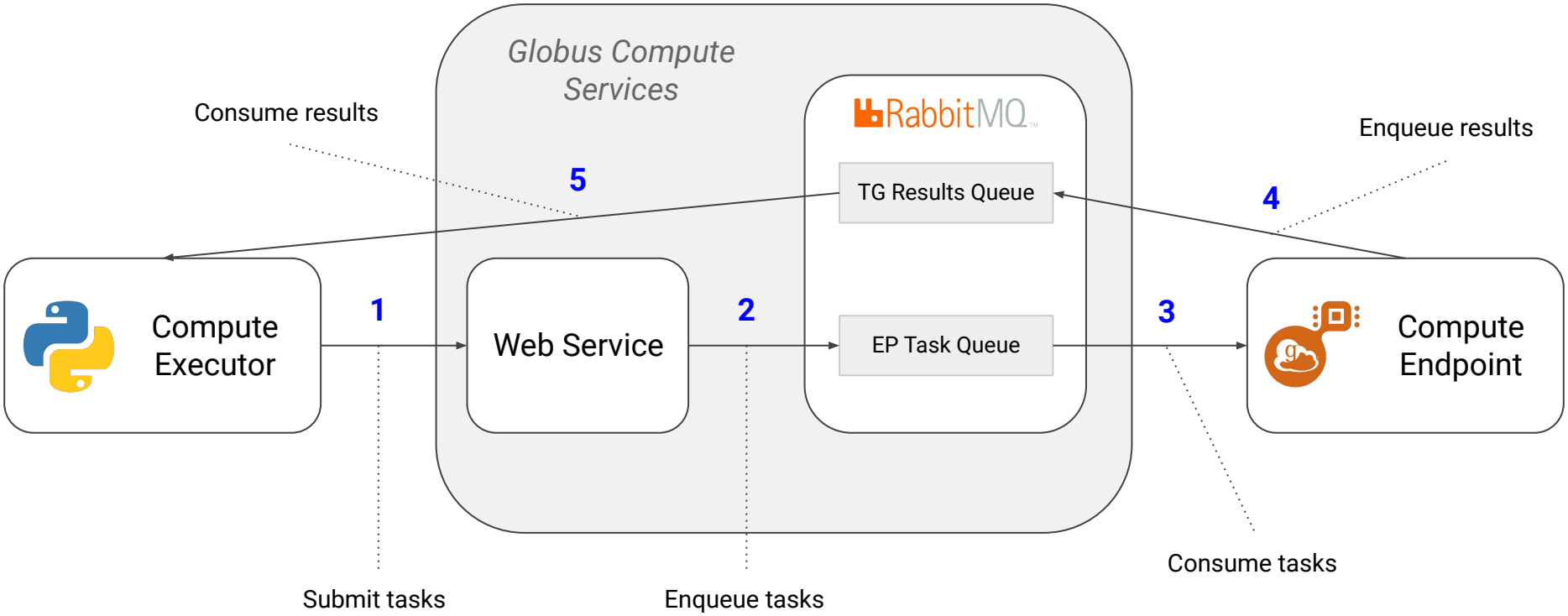
```
# Blocking
print(f"Result: {fut.result()}")

# Non-blocking
print(f"Status: {fut.done()}")
```

What's the difference? **ComputeFuture** objects are associated with tasks

```
>>> print(fut.task_id)
3223b5fd-6fe4-4c78-bfe4-ec0950048559
```

A peak under the hood



An example workflow

Create some functions

Submit tasks to an endpoint running on Polaris (ALCF)

Submit tasks to an endpoint running on Midway (UChicago), using the former results as arguments

```
from concurrent.futures import as_completed
from globus_compute_sdk import Executor

def func1(x: int) -> int:
    ...
def func2(x: int) -> int:
    ...

polaris_ep_id = "..."
with Executor(polaris_ep_id) as gce:
    polaris_futs = [
        gce.submit(func1, i) for i in range(100)
    ]

    midway_ep_id = "..."
    with Executor(midway_ep_id) as gce:
        midway_futs = []
        for f in as_completed(polaris_futs):
            midway_futs.append(
                gce.submit(func2, f.result())
            )
```

Any questions?

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