UniFaaS: Programming across Distributed Cyberinfrastructure with Federated Function Serving

Yifei Li*, Ryan Chard‡, Yadu Babuji†‡, Kyle Chard†‡, Ian Foster†‡, Zhuozhao Li*

* Dept. of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, China

† Dept. of Computer Science, University of Chicago, Chicago, IL, USA

‡ Data Science and Learning Division, Argonne National Laboratory, Lemont, IL, USA

Motivation

SUSTech Southern University of Science and Technology

- Executing scientific workflows across cyberinfrastructure(CI)
  amortizing queue times, distributed data, specialized accelerator etc.

- When executing distributed scientific workflows
  - **funcX**
    Pros: Easy to build a distributed computing resource pool
    Cons: Independent execution, manual data staging, limitations of input/output size
  - **Parsl**
    Pros: Support the DAG workflow, data staging (e.g. FTP, HTTP)
    Cons: Complicated to execute workflows on distributed CI simultaneously

- What about funcX as an executor of Parsl?
  - Things can be resolved immediately
    easy to program (in Parsl's way), distributed execution

  - Things to be resolved
    data management, performance (scheduling)

Programming and Architecture

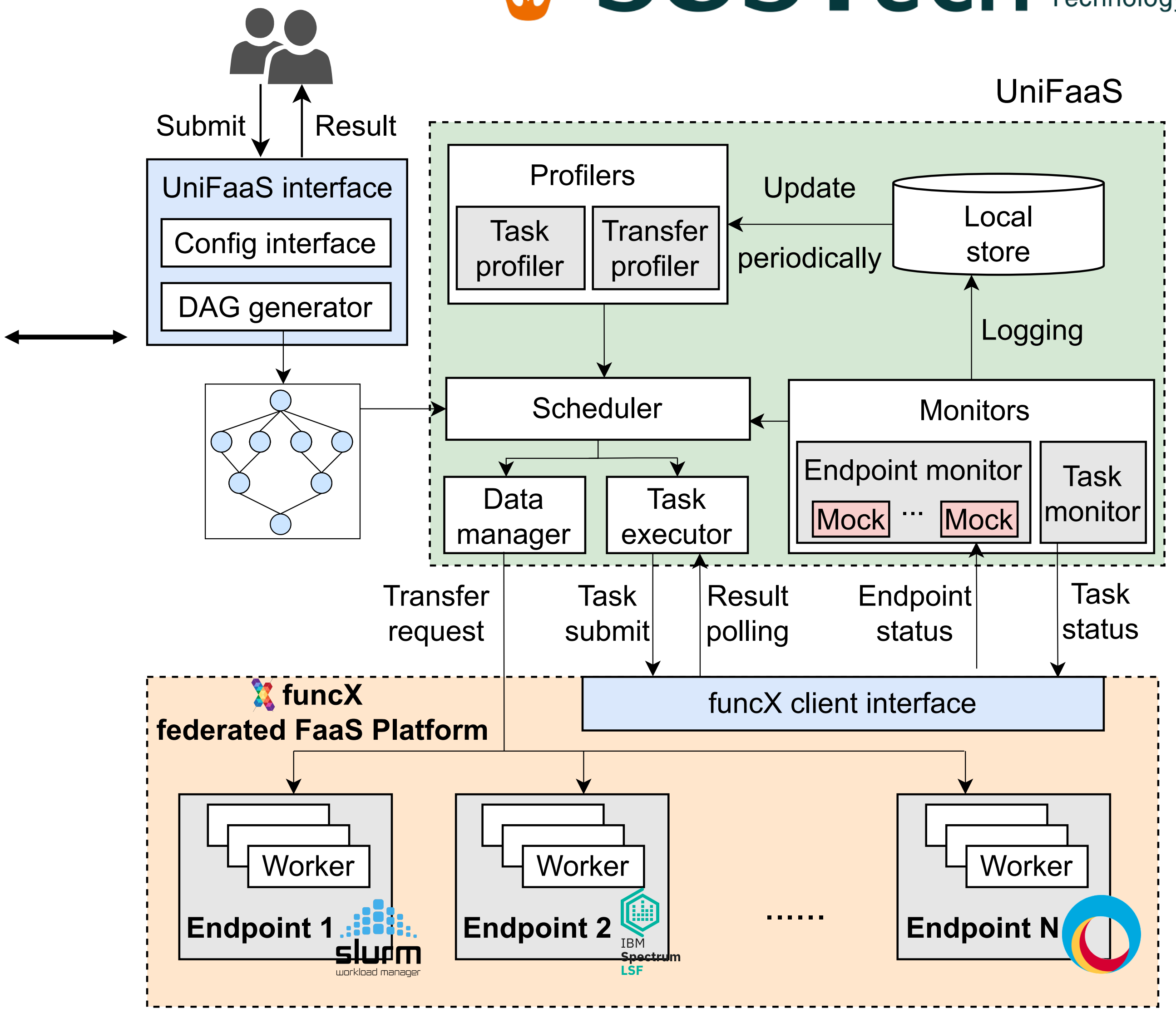
```
@function
def compute_fingerprint(GlobusFile: mol_file):
    from rdkit import *
    import GlobusFile

    mol_path = mol_file.get_remote_file_path()
    molecule = open(mol_path).readline()
    fp = AllChem.GetMorganFingerprint(
        Chem.MolFromSmiles(molecule), 2)

    out_file = GlobusFile.create("fp.txt")
    out_path = out_file.get_remote_file_path()
    open(out_path, 'w').write(fp)
    return out_file
```

: a shim layer to wrap data and R/W ops.

: a decorator like @python_app in Parsl



UniFaaS architecture

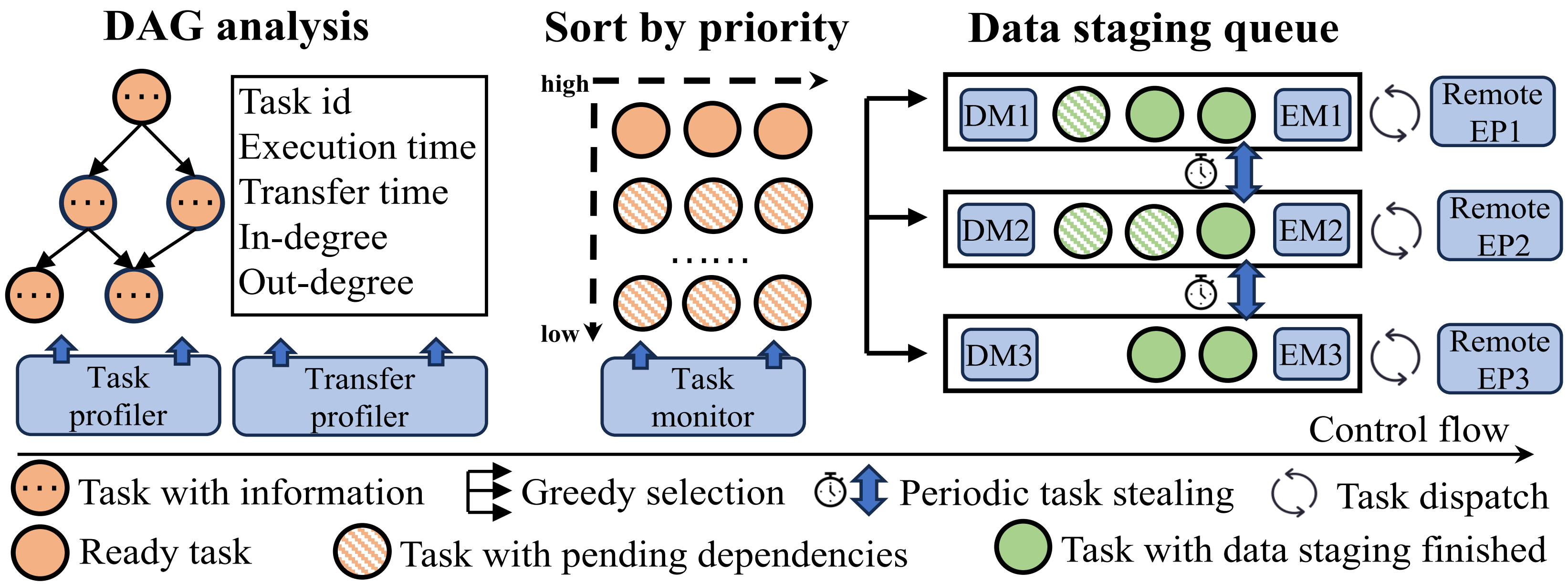
UniFaaS Scheduling

Goal: to minimize workflow's makespan

Challenges: varying data staging time, dynamic resource capacity.

Intuition:

- Data staging problem: start it as early as possible
- Dynamic resource capacity : real-time scheduling



Dynamic heterogeneity-aware scheduling (DHA in short)

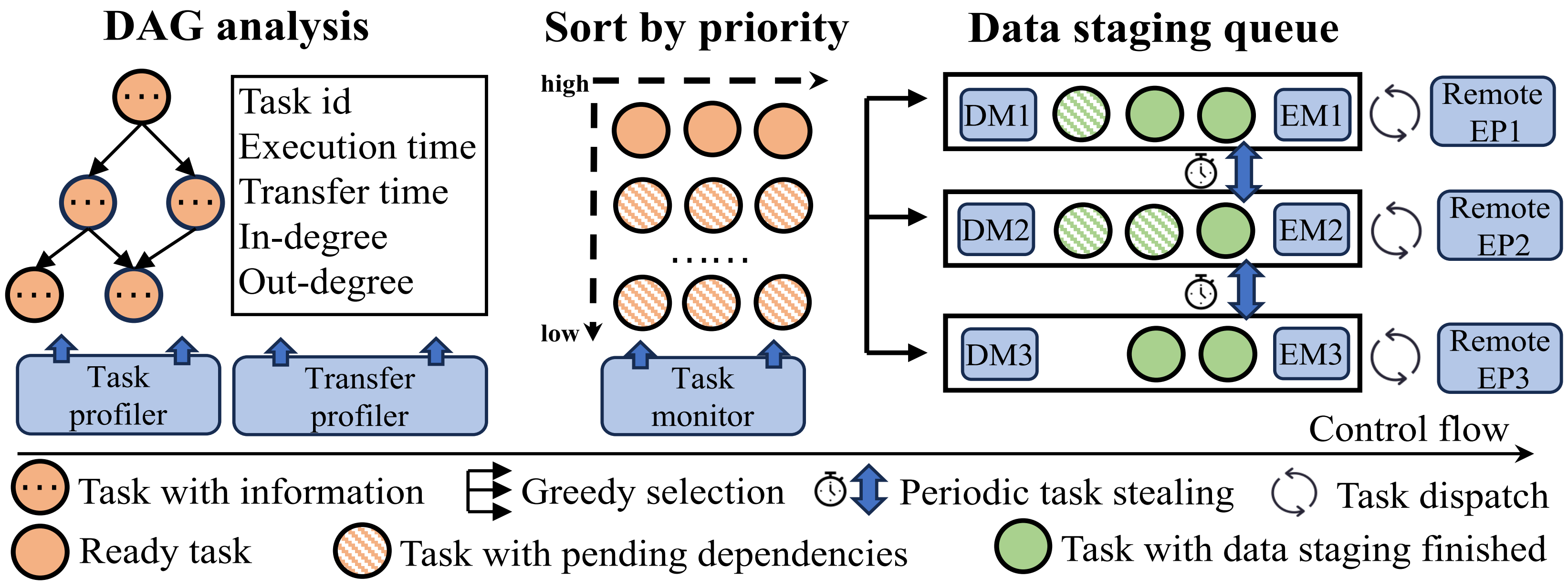
UniFaaS Scheduling

Goal: to minimize workflow's makespan

Challenges: varying data staging time, dynamic resource capacity.

Intuition:

- Data staging problem: start it as early as possible
- Dynamic resource capacity : real-time scheduling



Dynamic heterogeneity-aware scheduling (DHA in short)

UniFaaS Scheduling

No prior knowledge

- Locality-aware scheduling for dynamic resource capacity
  schedule based on real-time status (real-time)

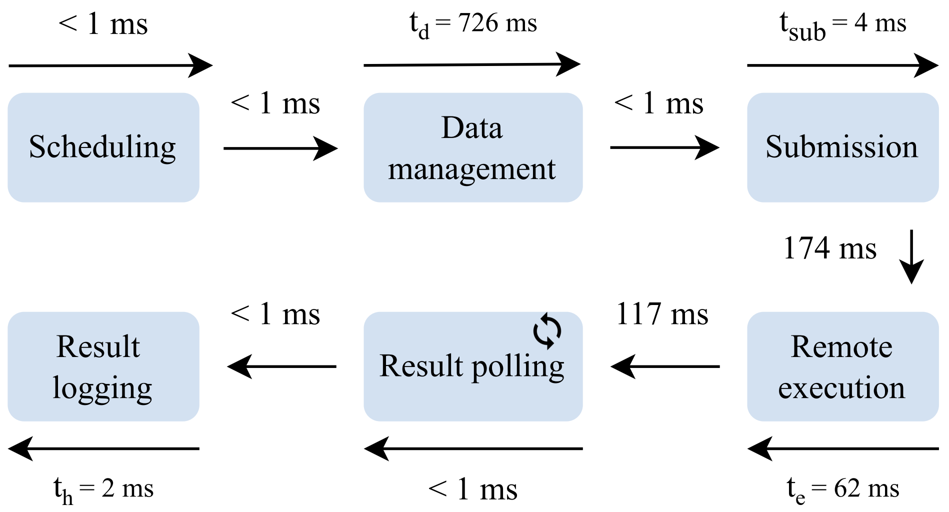
- Capacity-aware scheduling for static resource capacity
  schedule when the DAG enters our system (offline)

SUMMARY OF THE SCHEDULING ALGORITHMS.

	Capacity	Locality	DHA
Scheduling type	Offline	Real-time	Hybrid
Dynamic DAG supported	✗	✓	✓
Dynamic resource supported	✗	✓	✓
Knowledge required	✗	✗	✓

Experiment

Latency



OVERHEAD OF DIFFERENT ALGORITHMS.

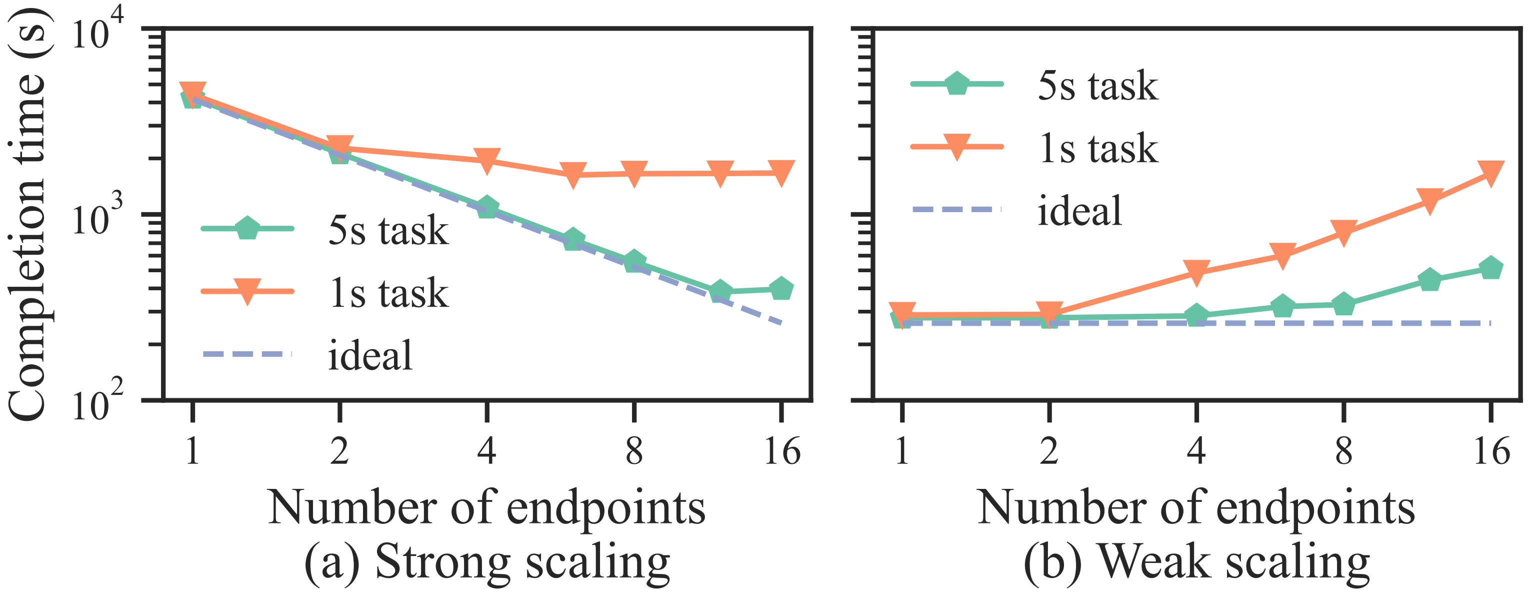
Scheduling algorithm	Overhead (s)
Capacity	1.72×10^{-4}
Locality	3.00×10^{-3}
DHA	3.46×10^{-3}

One "hello world" task with a 1 MB input totally costs 1087 ms.

All algorithms have a modest overhead.

Experiment

Scalability

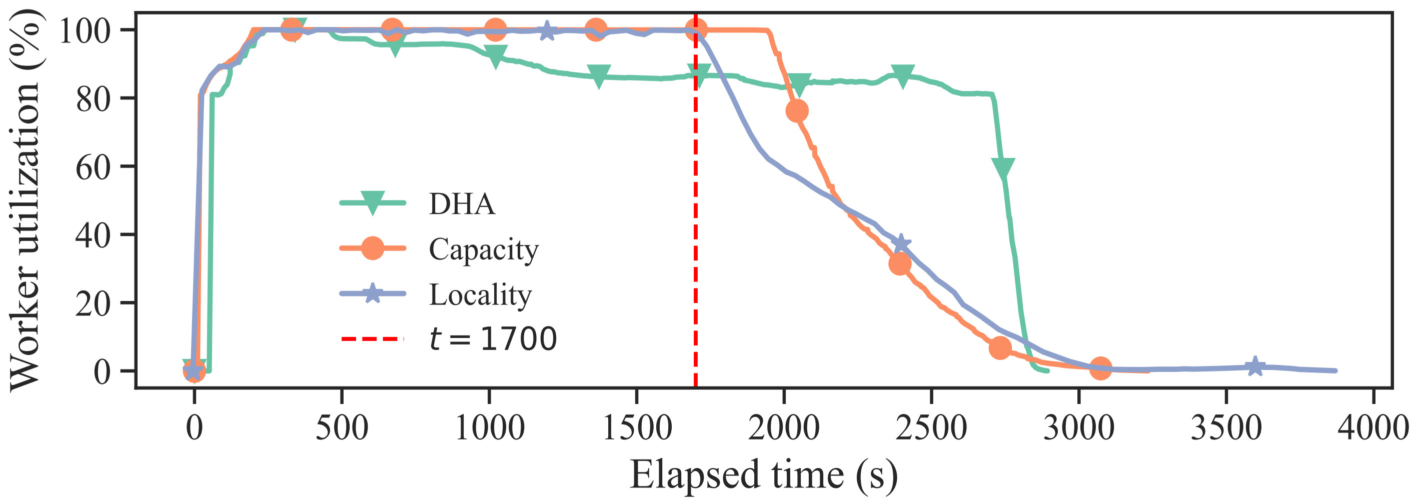


Scalability of 5-second tasks is close to the ideal for up to 12 endpoints
longer-duration tasks, better scaling

Experiment

Case study

1. DHA has the best performance and highest worker utilization.
2. Improved performance by 22.99%, while utilizing only an additional 19.48% of resources.



Experiment	Makespan (s)	Transfer size (GB)
Capacity	3,240	4.86
Locality	3,882	53.46
DHA	2,898	44.94
Baseline: HPC-I only	3,763	0

Execute the drug screening workflow under static resource capacity.

Questions?

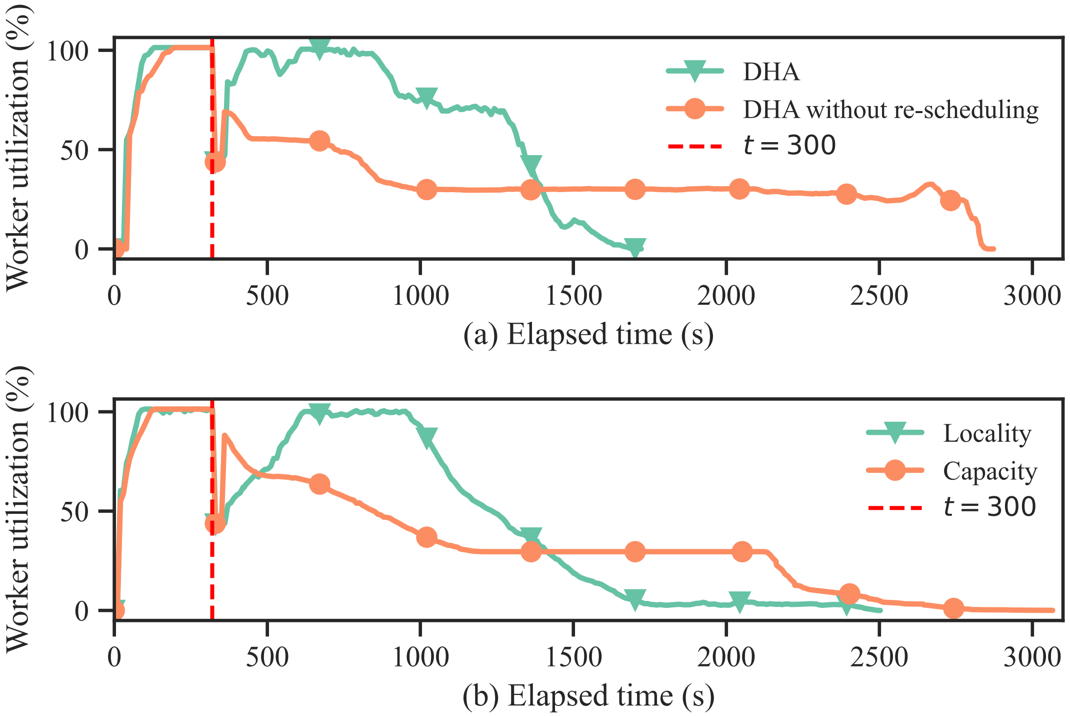
Yifei Li
12232396@mail.sustech.edu.cn

Experiment

Southern University of Science and Technology

Case study: dynamic capacity

1. DHA has the best performance.
2. Locality is better than DHA without re-scheduling.



Experiment	Makespan (s)	Transfer size (GB)
Capacity	3,070	3.54
Locality	2,507	58.93
DHA without re-scheduling	2,880	55.36
DHA	1,727	43.32

Execute the drug screening workflow under dynamic resource capacity.