TECHNISCHE UNIVERSITÄT BERLIN

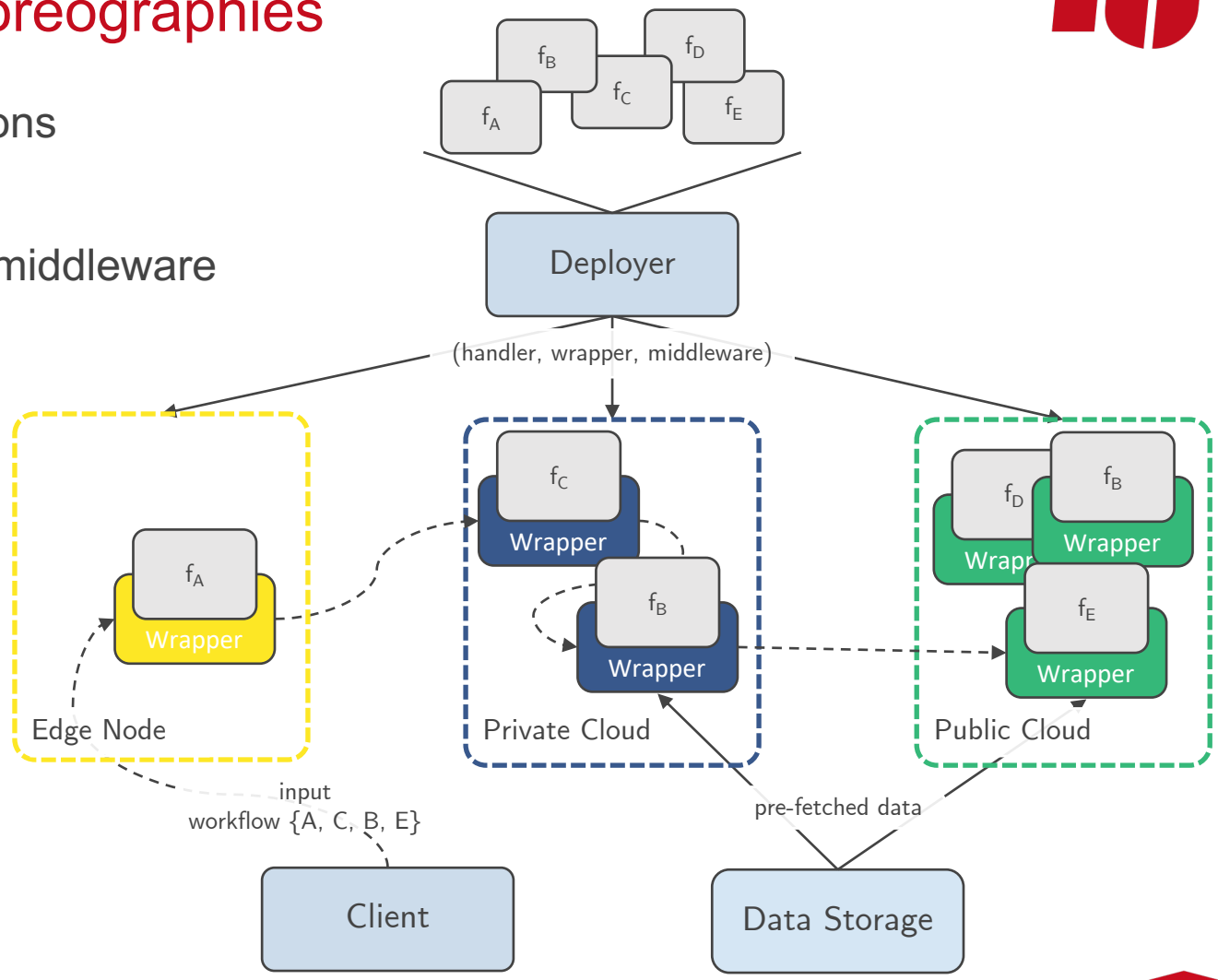
GeoFF: Federated Serverless Workflows with Data Pre-Fetching

Valentin Carl, Trever Schirmer, Tobias Pfandzelter, David Bermbach
Scalable Software Systems

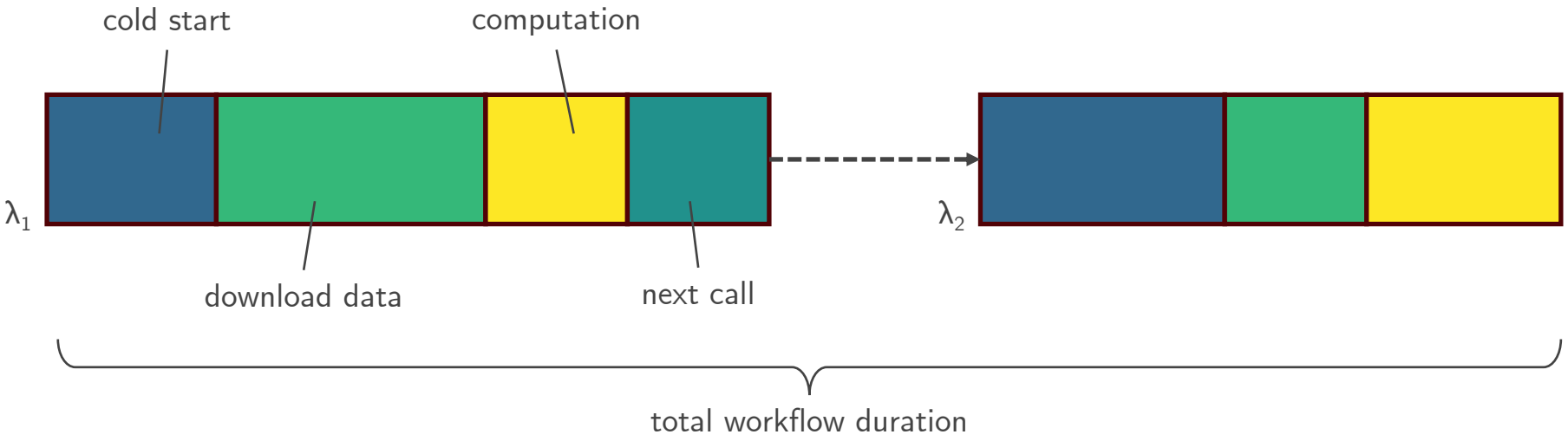
3S

Federated Serverless Workflow Choreographies

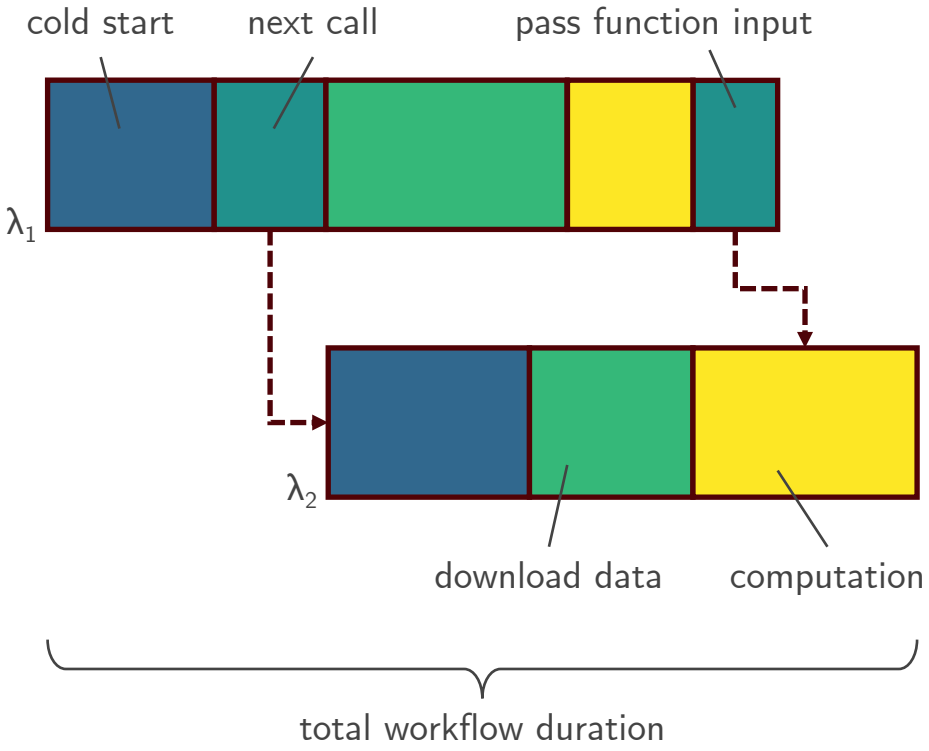
- Developers write platform-independent functions and a deployment specification
- Functions are deployed with a choreography middleware in wrapper
- Workflows are executed across platforms



Workflow **without** Pre-Fetching

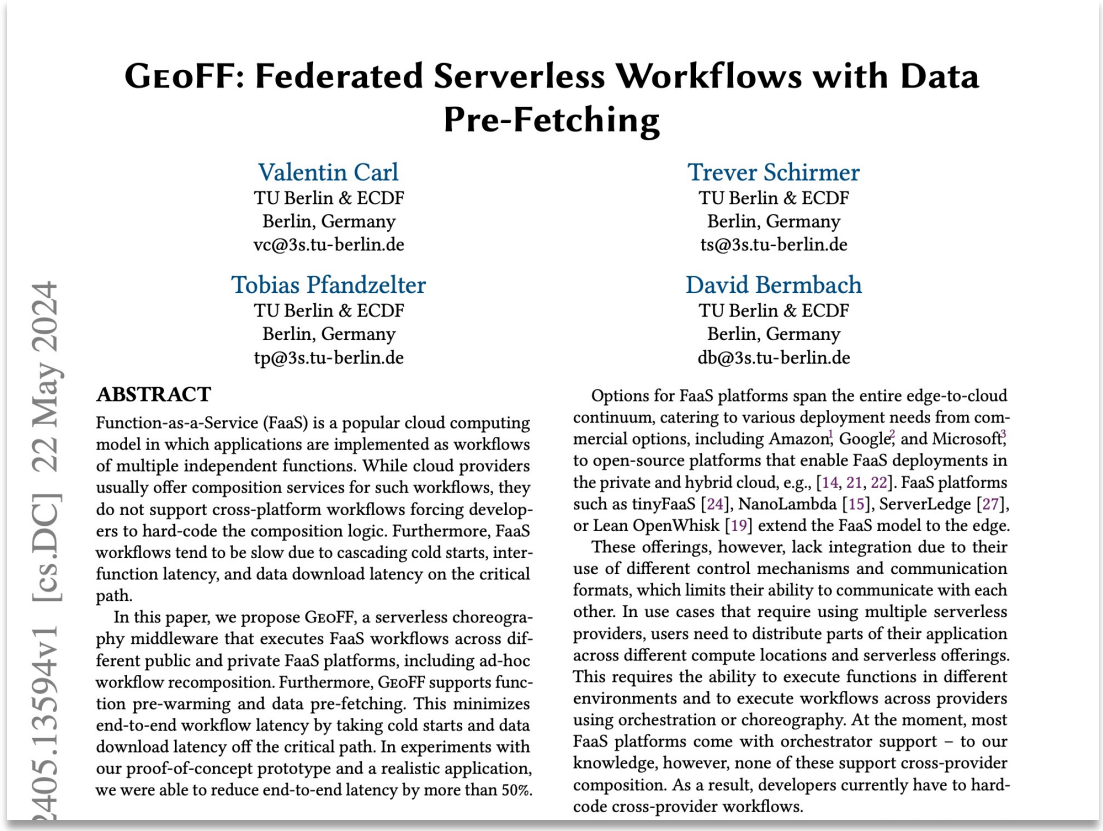


Workflow **with** Pre-Fetching



Evaluation and Results

- Experiments with document processing workflow on AWS Lambda, Google Cloud Functions, and tinyFaaS[1]
- Improved total workflow duration across all experiments
- Source: https://github.com/valentin-carl/federation
- Full results in our paper (under review)



GeoFF: Federated Serverless Workflows with Data Pre-Fetching

Valentin Carl
TU Berlin & ECDF
Berlin, Germany
vc@3s.tu-berlin.de

Trever Schirmer
TU Berlin & ECDF
Berlin, Germany
ts@3s.tu-berlin.de

Tobias Pfandzelter
TU Berlin & ECDF
Berlin, Germany
tp@3s.tu-berlin.de

David Bermbach
TU Berlin & ECDF
Berlin, Germany
db@3s.tu-berlin.de

ABSTRACT

Function-as-a-Service (FaaS) is a popular cloud computing model in which applications are implemented as workflows of multiple independent functions. While cloud providers usually offer composition services for such workflows, they do not support cross-platform workflows forcing developers to hard-code the composition logic. Furthermore, FaaS workflows tend to be slow due to cascading cold starts, inter-function latency, and data download latency on the critical path.

In this paper, we propose GeoFF, a serverless choreography middleware that executes FaaS workflows across different public and private FaaS platforms, including ad-hoc workflow recomposition. Furthermore, GeoFF supports function pre-warming and data pre-fetching. This minimizes end-to-end workflow latency by taking cold starts and data download latency off the critical path. In experiments with our proof-of-concept prototype and a realistic application, we were able to reduce end-to-end latency by more than 50%.

Options for FaaS platforms span the entire edge-to-cloud continuum, catering to various deployment needs from commercial options, including Amazon[1], Google[2], and Microsoft[3], to open-source platforms that enable FaaS deployments in the private and hybrid cloud, e.g., [14, 21, 22]. FaaS platforms such as tinyFaaS [24], NanoLambda [15], ServerLedge [27], or Lean OpenWhisk [19] extend the FaaS model to the edge.

These offerings, however, lack integration due to their use of different control mechanisms and communication formats, which limits their ability to communicate with each other. In use cases that require using multiple serverless providers, users need to distribute parts of their application across different compute locations and serverless offerings. This requires the ability to execute functions in different environments and to execute workflows across providers using orchestration or choreography. At the moment, most FaaS platforms come with orchestrator support – to our knowledge, however, none of these support cross-provider composition. As a result, developers currently have to hard-code cross-provider workflows.

https://arxiv.org/abs/2405.13594

[1] Tobias Pfandzelter and David Bermbach. 2020. tinyFaaS: A Lightweight FaaS Platform for Edge Environments. In Proceedings of the Second IEEE International Conference on Fog Computing (ICFC '20). 17–24.