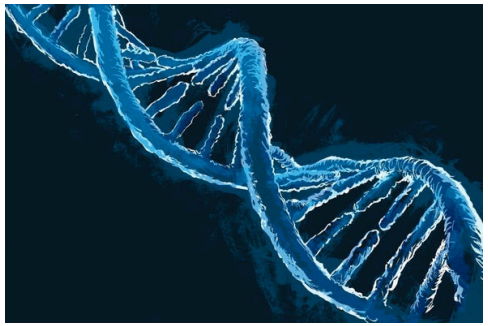
A Study of Orchestration Approaches for Scientific Workflows in Serverless Computing

Abdallah Elshamy, Ahmed Alquraan, and Samer Al-Kiswany

Acronis

Scientific Workflows Overview

- Critical for advances in science
- Noticeable increase in the computation demands



Platforms for Scientific Workflows

AWS Lambda

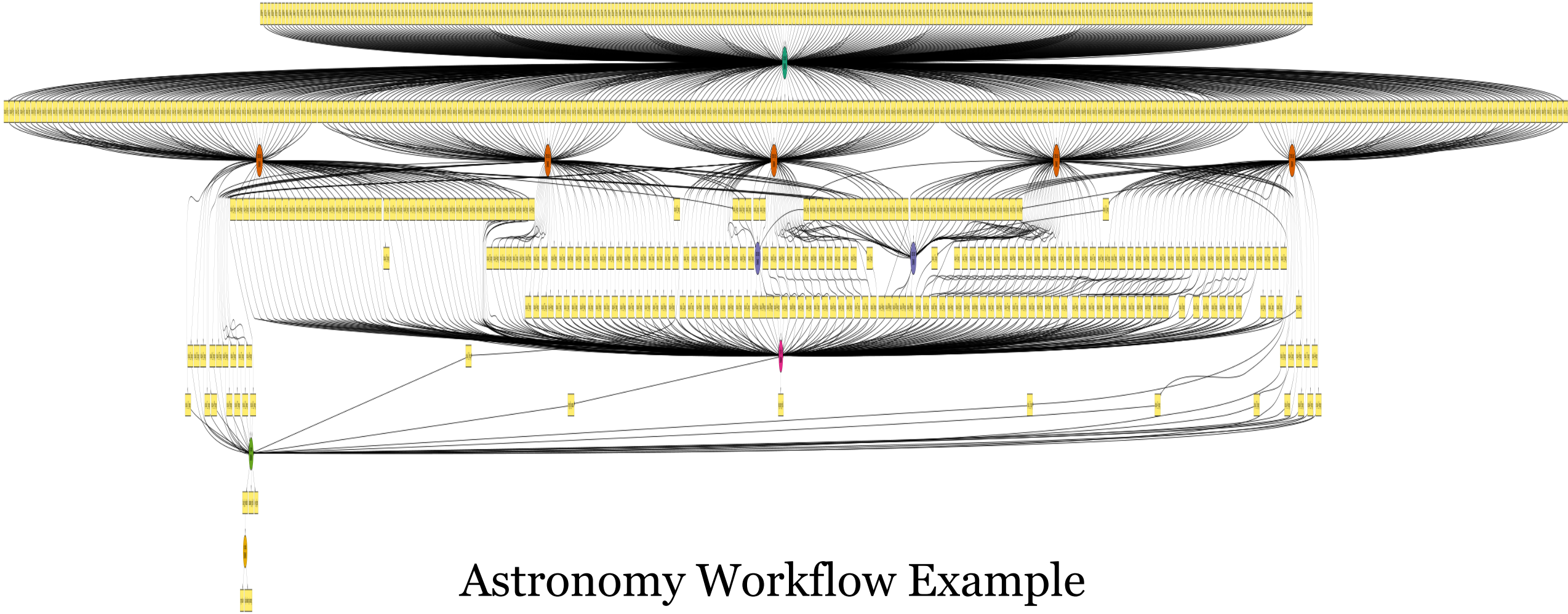
Azure Functions

Google Cloud Functions

- Significant capital and operation cost
- Elasticity & Pay-as-you-go

Is using modern cloud paradigms (serverless) a viable option?

Example of a Scientific Workflow



Astronomy Workflow Example

Comparison with Typical Serverless Workflows

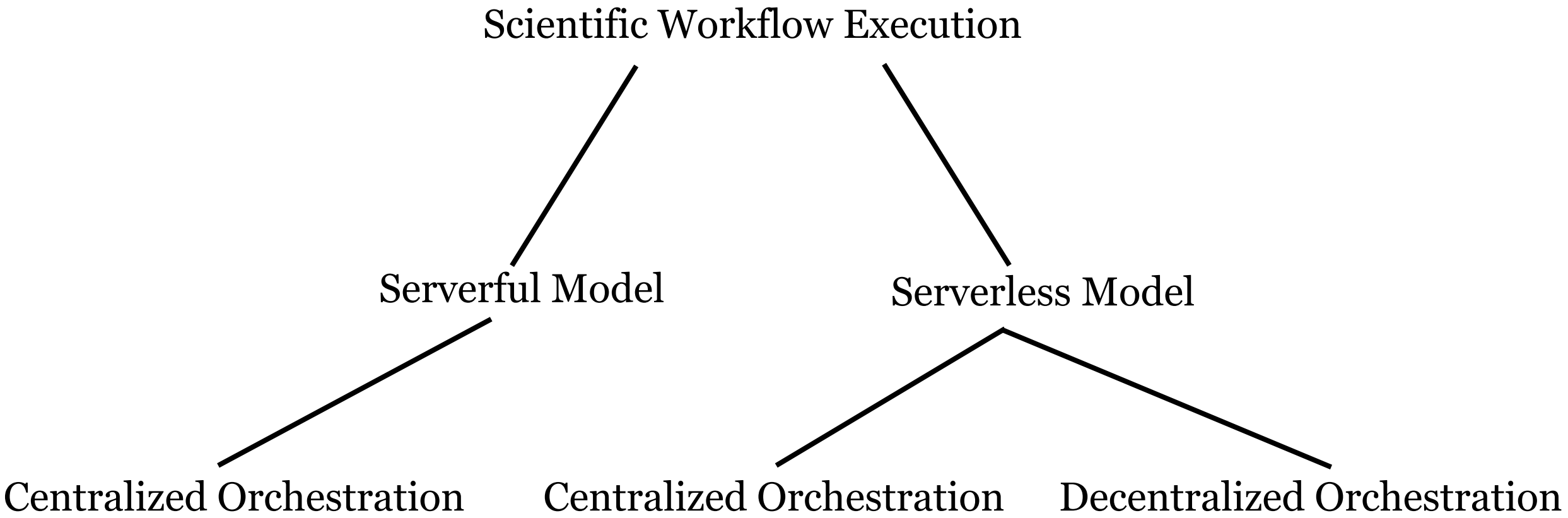
Scientific workflows:

- more data- and compute-intensive
- significantly higher degree of parallelism
- longer execution time
- larger amount of intermediate data
- bursty

GOALS

- What are the different orchestration approaches and their tradeoffs?
- What are the possible (IO/Scheduling) optimizations?
- How can the current serverless ecosystem provide a better support for scientific workflows?

Workflow Orchestration Approaches



Serverful Model

- Traditional execution model
- A set of computing nodes (i.e., workers) are responsible for executing the tasks of a workflow
- A storage service is used to share data between nodes

Serverful Model: Centralized Orchestration

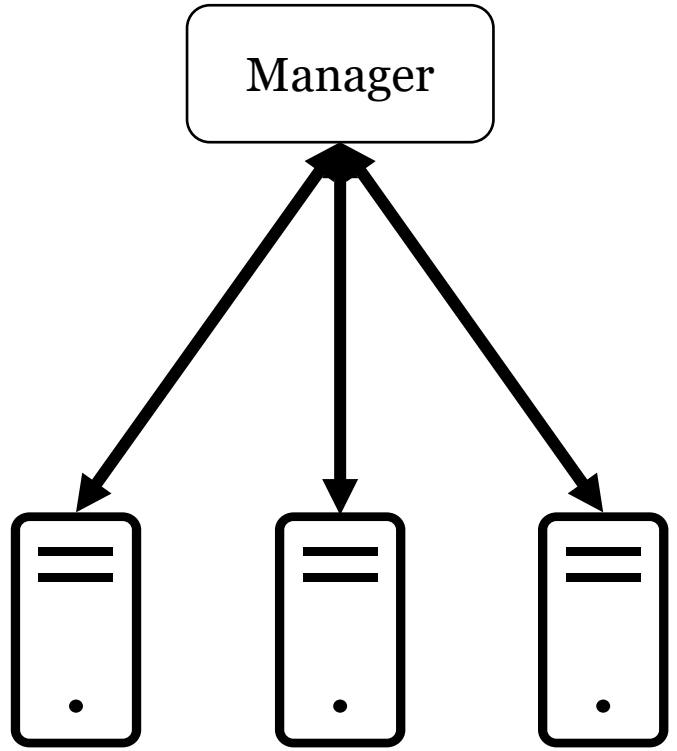
+ More efficient task to worker assignment

+ Better resource utilization

+ Shorter end-to-end execution time

- Responsibility for managing resources

- Harder to auto-scale resources



Serverless Model

- Dynamic scaling of resources
- A storage service is used to share data between workers

AWS Lambda

Azure Functions

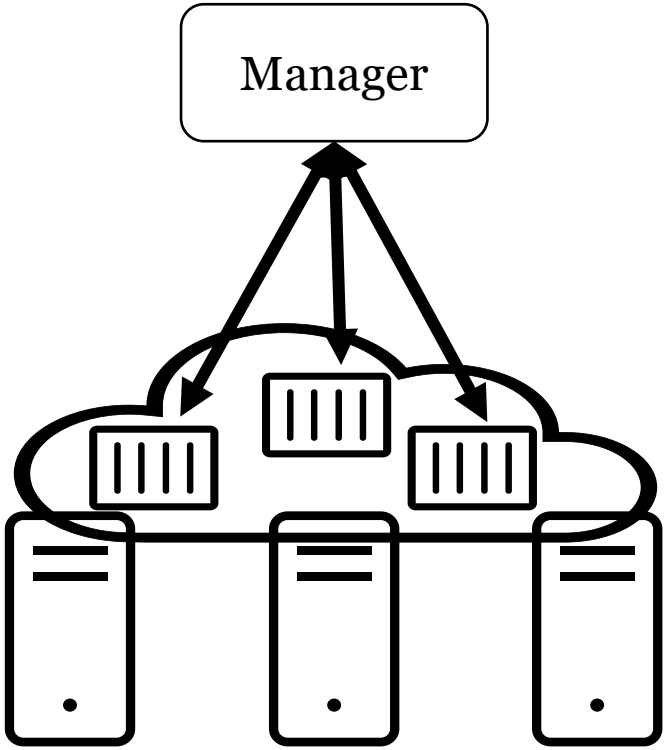
Google Cloud Functions

Serverless Model: Centralized Orchestration

+ Simple

- The workflow manager is a stateful, long-running process.

- The workflow manager cannot be deployed as serverless function or container.

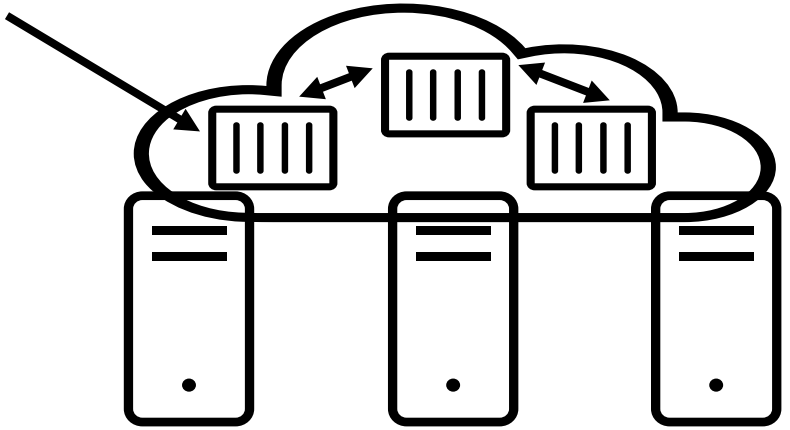


Serverless Model: Decentralized Orchestration

+ Purely serverless approach

- Considerable development effort

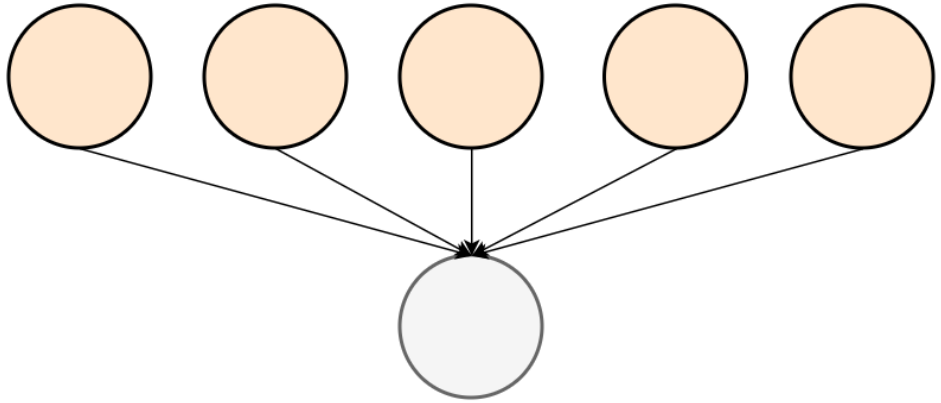
- Complicates handling some workflow patterns



Challenges in Reduce Stages in Decentralized Orchestration

- It is hard to know when the last parallel task finishes.

- The only way to infer that parallel tasks are done is by checking their output files.



Evaluation

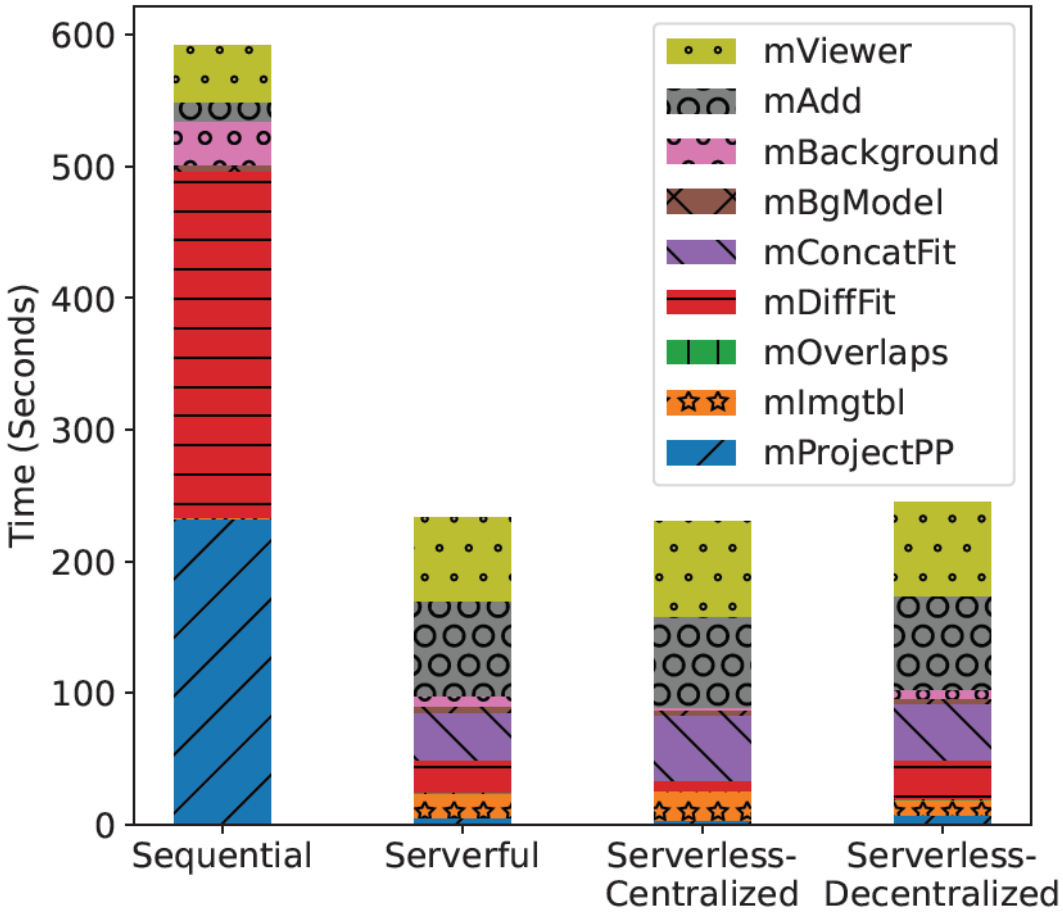
- What is the effect of the orchestration approach on the end-to-end execution time?
- How effective are IO and scheduling optimization?
- What is the effect of cold starts on the performance?

Evaluation Setup

- Alternatives:
  - Serverful-Centralized
  - Serverless-Centralized
  - Serverless-Decentralized
- Storage:
  - Distributed file system (CephFS v16.2.1)
  - Only uses RAM disks (50 GB)
- Workflow:
  - Montage application ("2MASS J" dataset)
  - Size 4 and location "M 101"(4034 tasks)
- Serverless Environment:
  - Knative v1.8.0 on a Kubernetes environment v1.24.6
  - No limit on CPU and RAM
- Setup:
  - 11 CloudLab nodes (Wisconsin data center)
  - Two Intel Xeon Silver 10-core CPUs
  - 196 GB of RAM
- Average of 15 trials
  - std dev < 9% (15% for cold starts)

Execution time

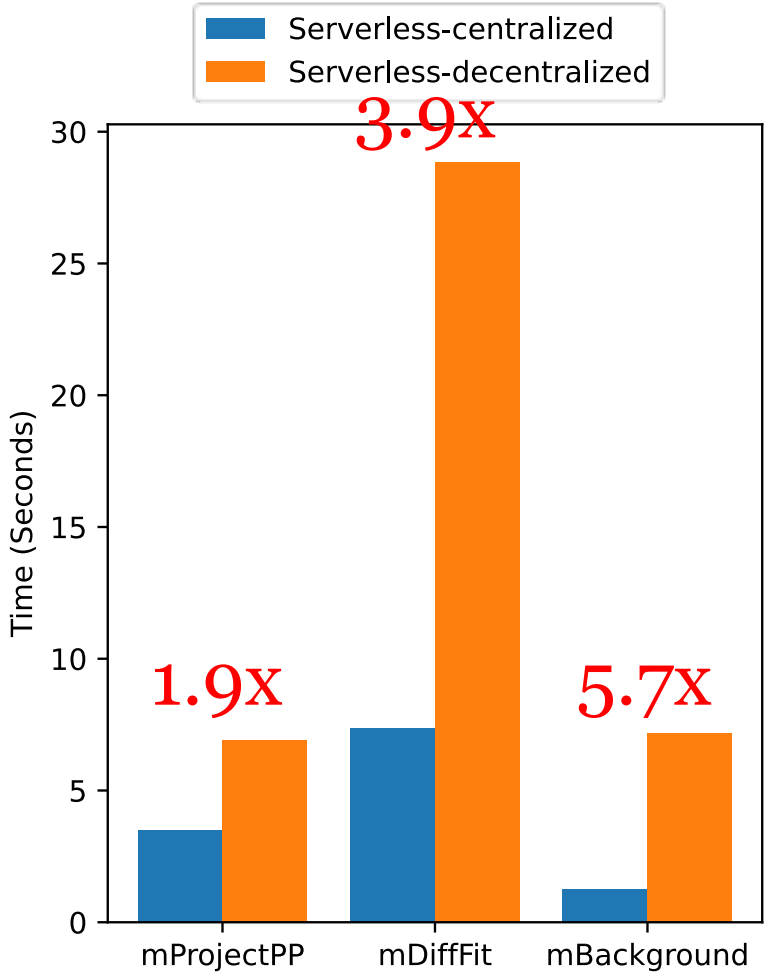
Serverless-centralized and serverless-decentralized alternatives achieve a comparable performance to the serverful-centralized alternative.



Overhead of the Orchestration Approach

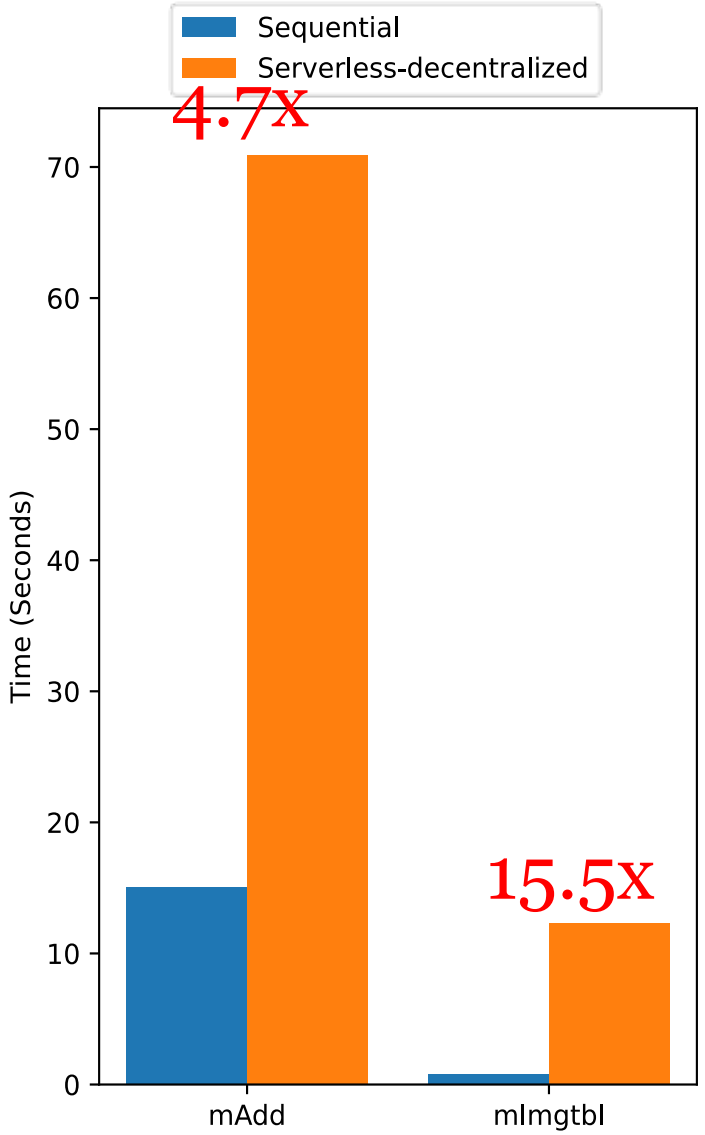
- A significant slowdown in the parallel stages for the serverless-decentralized alternative compared to the serverless-centralized.

- This is due to using the storage for orchestration in the parallel stages.



Effect of Distributed Storage

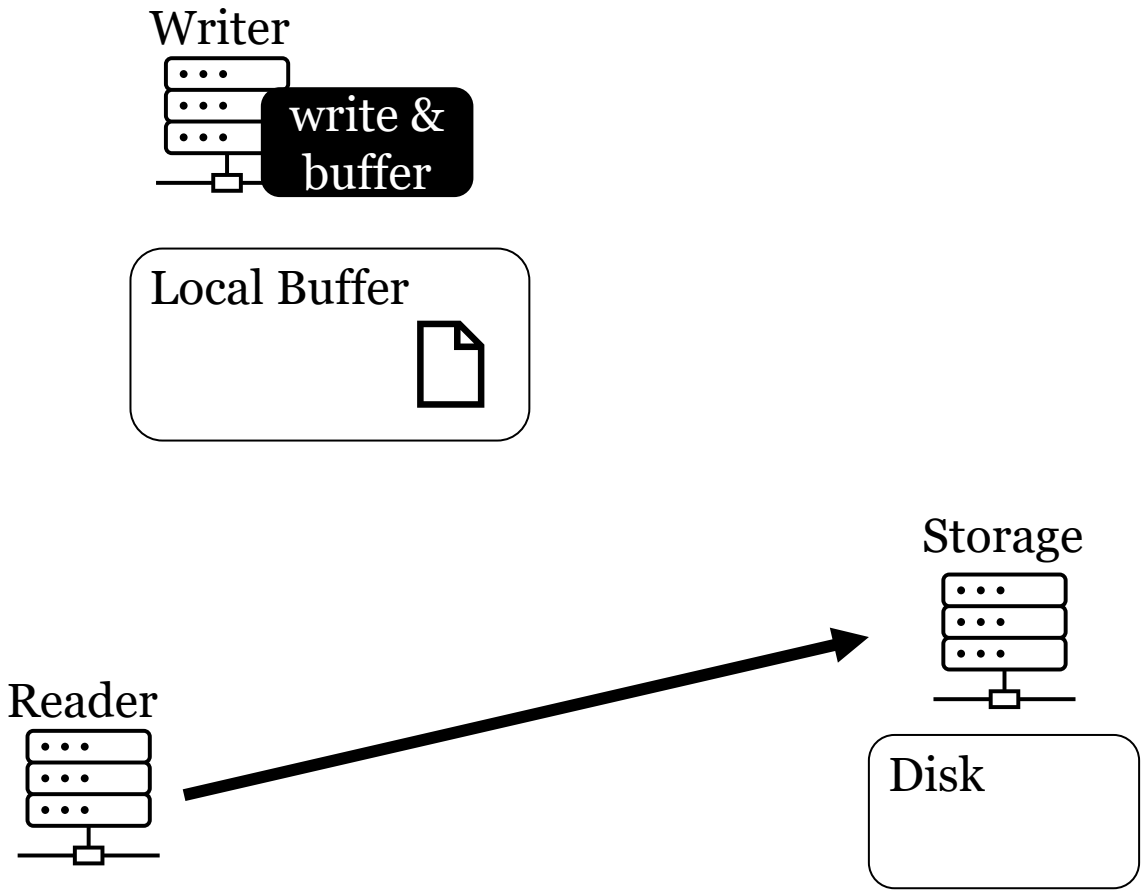
Serverful and serverless alternatives have low performance in the reduce stages.



Behavior of CephFS

- CephFS uses file system privileges.

- Changing file ownership imposes a high overhead.



Locality Optimizations

Prefetching file privileges

Move the operation of changing file ownership from the reduce stage to the parallel stage

- Overlaps the process of acquiring file privileges by the reducer with parallel tasks that are still running
- Improves concurrency

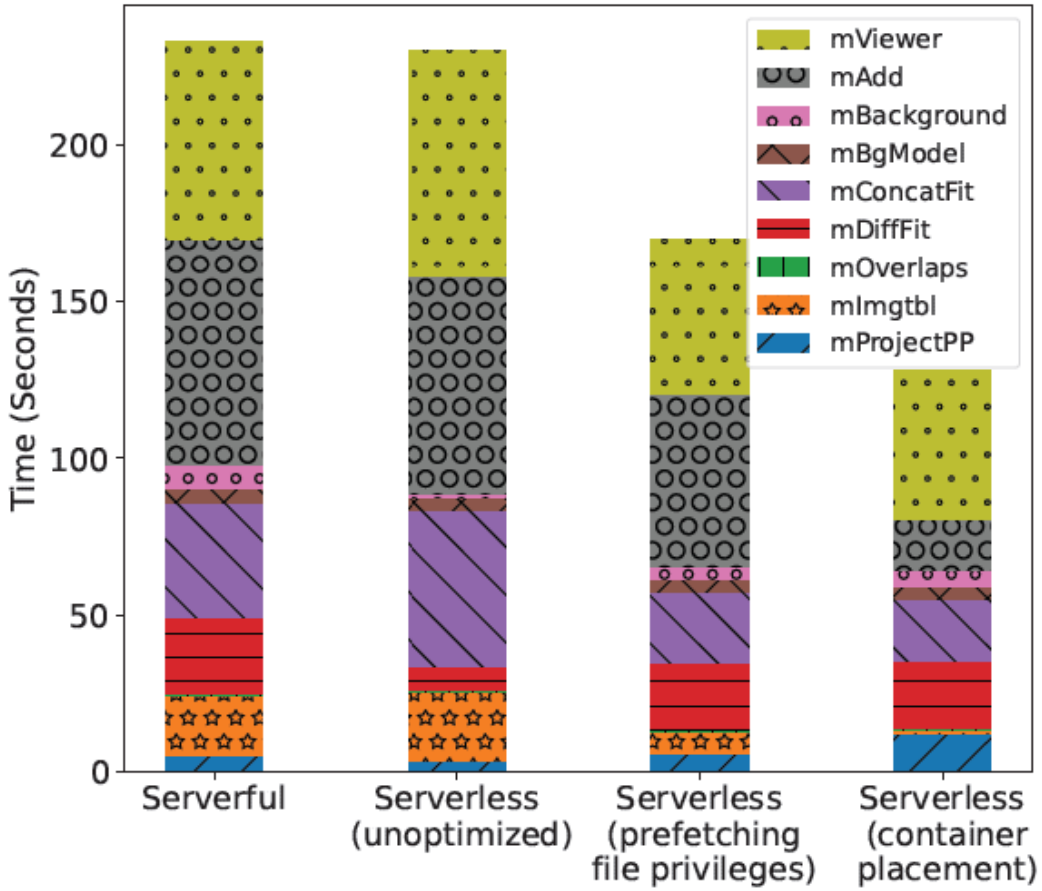
Container placement

Improve locality through container placement

- Containers of tasks that share a large amount of data are placed on the same node
- Avoids changing file ownership
- Improves data locality

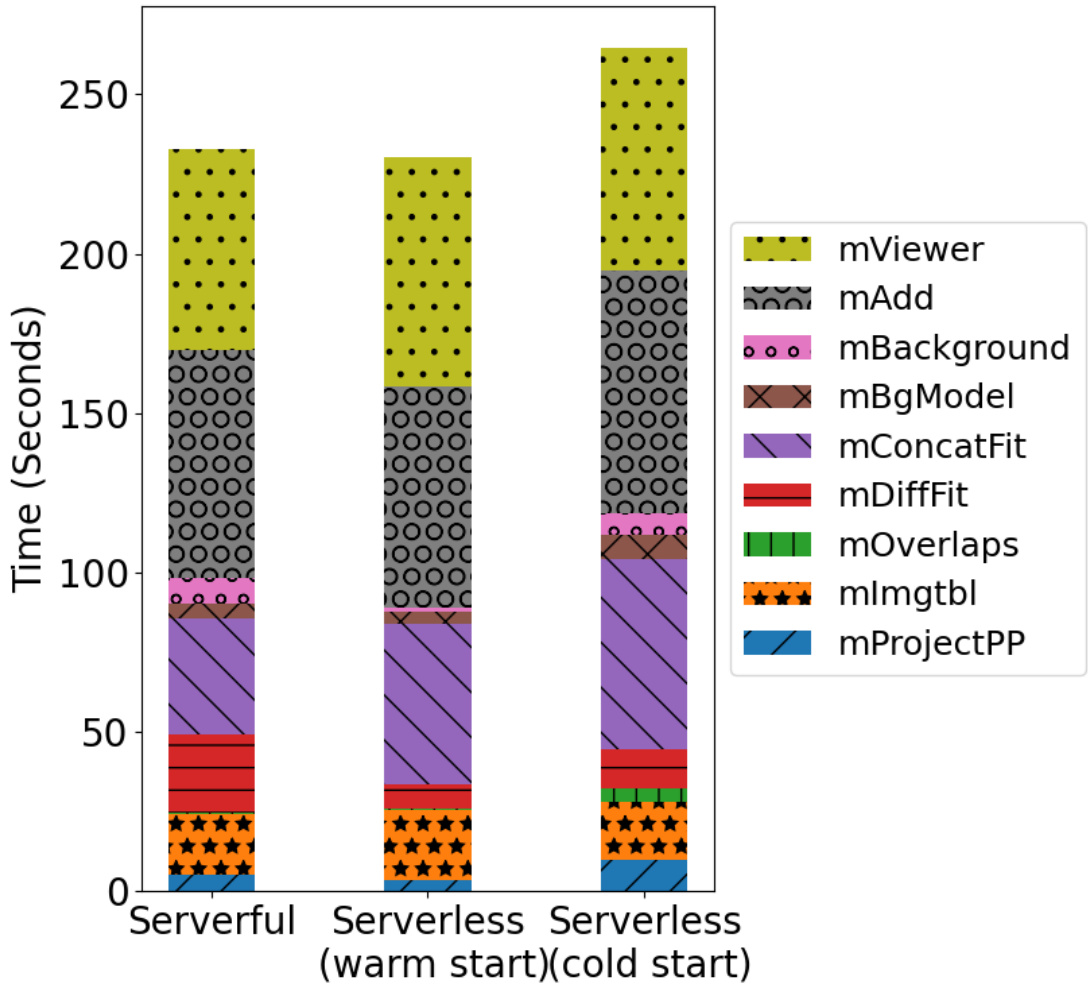
Effect of Locality Optimizations

- Prefetching file privileges reduces the end-to-end execution time by 26%.
- Container placement reduces the end-to-end execution time by 44%.



Effect of Cold Starts

- Cold starts increased the total execution time by 14.89%.
- The effect of cold starts on some of the stages is non-noticeable.
- This is because these stages run for a long time.



Discussion

- Serverless computing is a viable approach for scientific workflows performance wise.
- Current serverless platforms have implicit assumptions about the workloads (timeouts).
- It is critical to coordinate the scheduling decisions and the file system mechanisms.
- Optimizations may have a negative effect.
- Decentralized orchestration brings cost savings and new challenges.
- Workflow burstiness adds a new challenge for serverless.

Future Work

- Extend our study with other workflows that have different patterns.
- Create a framework that leverages the insights we find in our study.
- Similar evaluations for resource utilization and cost.
- Similar evaluations with a larger cluster.