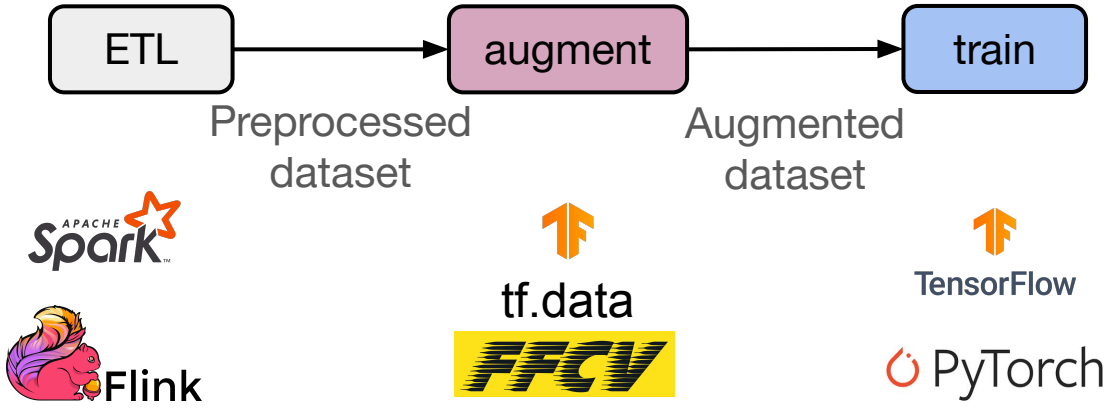
ExoFlow: A Universal Workflow System for Exactly-Once DAGs

Siyuan Zhuang, Stephanie Wang, Eric Liang, Yi Cheng, Ion Stoica

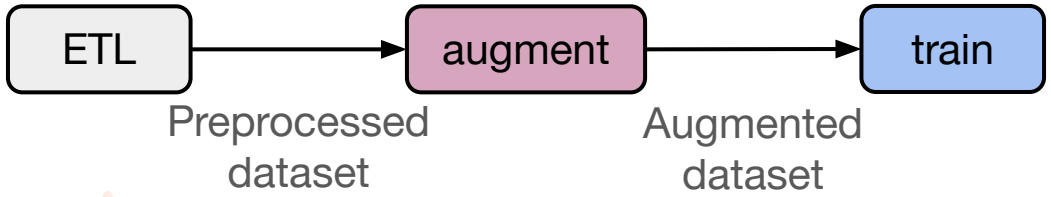

Workflows: Heterogeneous application pipelines

Distributed ML training workflow



Workflows: Heterogeneous application pipelines

<u>Distributed ML training workflow</u>

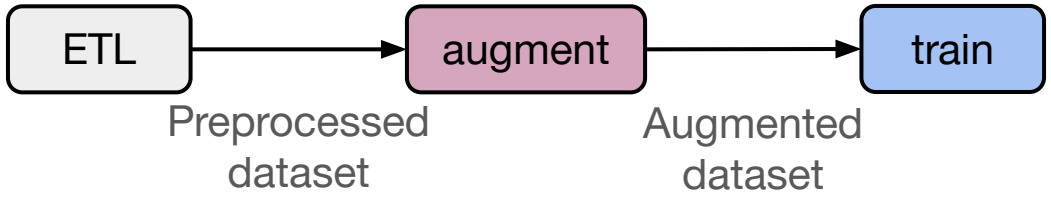


The workflow system handles *orchestration*:
Execution + Failure recovery

- **Automatic:** Event-triggered execution, transparent failover
- **High-performance:** Scalability, minimize application overhead
- **General:** Interoperate with application code and third-party systems

Workflows: Heterogeneous application pipelines

Distributed ML training workflow



Serverless microservices workflows

AWS Step Functions

Azure Durable Functions

Workflows: Heterogeneous application pipelines

<u>Distributed ML training workflow</u>

- Data-intensive, offline
- Idempotent
- (mostly) Deterministic

<u>Serverless microservices workflows</u>

- Latency-sensitive, online
- Affects external state
- Nondeterministic

AWS Step Functions

Azure Durable Functions

Workflows: Heterogeneous application pipelines

The workflow system handles *orchestration*:
Execution + **Failure recovery**

Tradeoff between execution vs. recovery (failover time) overhead.
Current workflow systems choose a <u>single</u> point on this tradeoff space.
→ Different workflow systems for different use cases.

Apache Airflow

Checkpointing:

+ Low performance overhead
- Re-execute more on failure → bad fit for online workloads

Goal: Can we build a **universal** workflow system that enables a **flexible** choice over the tradeoff between recovery and performance?

Exoflow: Decoupling the unit of execution from the unit of recovery.

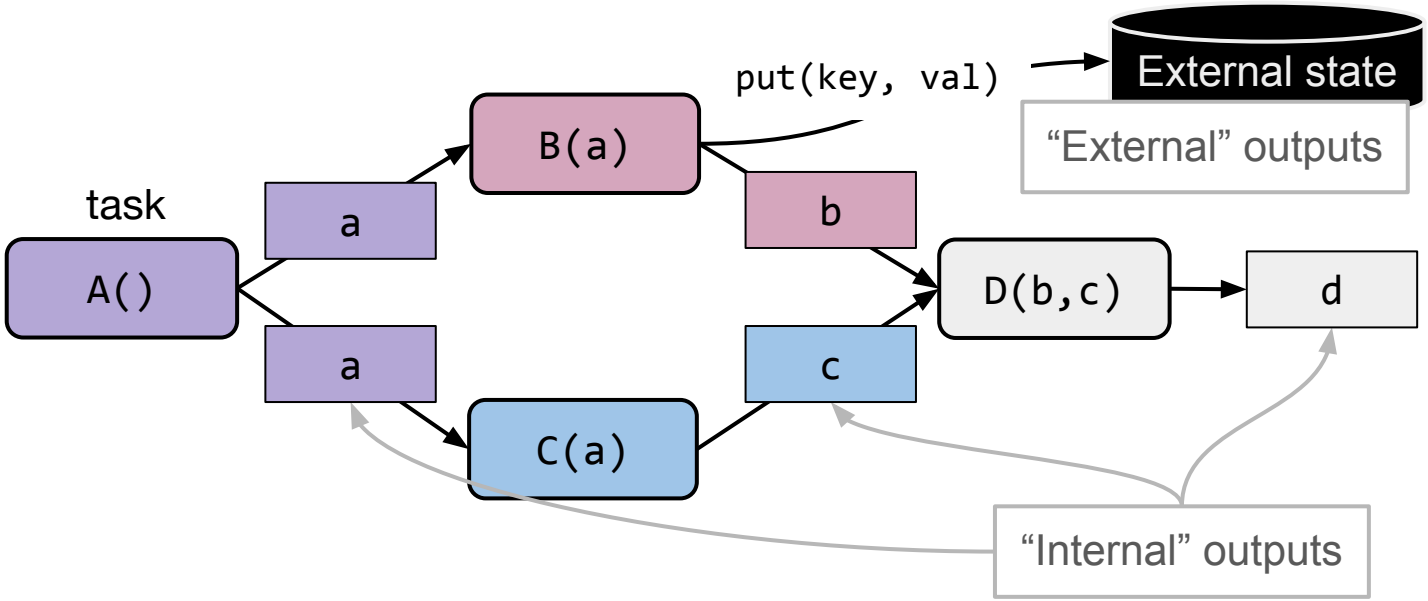
→ **Need application semantics**

Failure recovery for distributed workflows

<u>Exactly-once semantics:</u> Workflow output is equivalent to a failure-free execution.

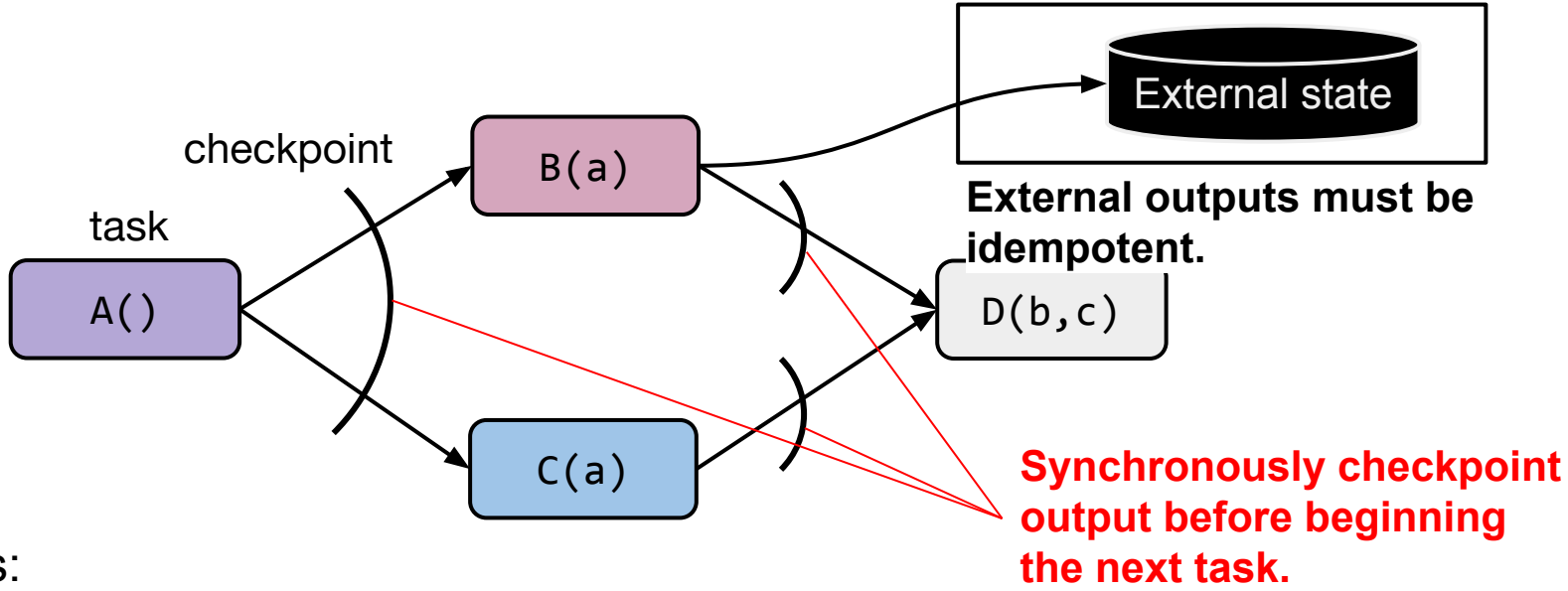
Failure recovery for distributed workflows

<u>Exactly-once semantics:</u> Workflow output is equivalent to a failure-free execution.



Failure recovery for distributed workflows

<u>Exactly-once semantics:</u> Workflow output is equivalent to a failure-free execution.

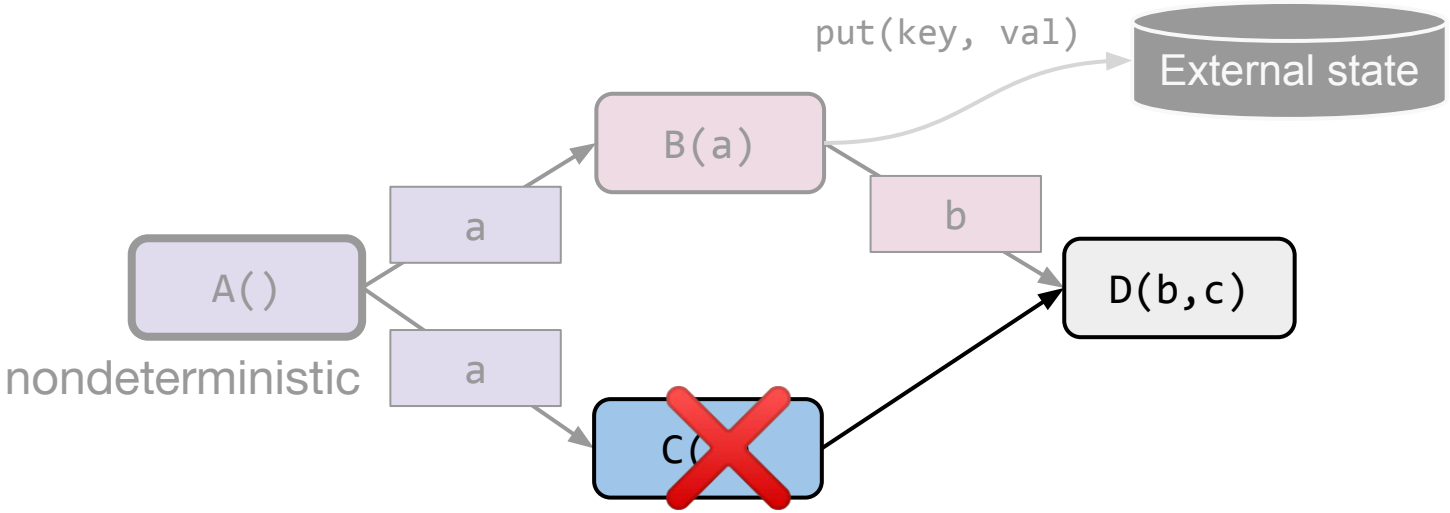


Problems:

1. Tasks may be nondeterministic.
2. Tasks may have external outputs visible to others.

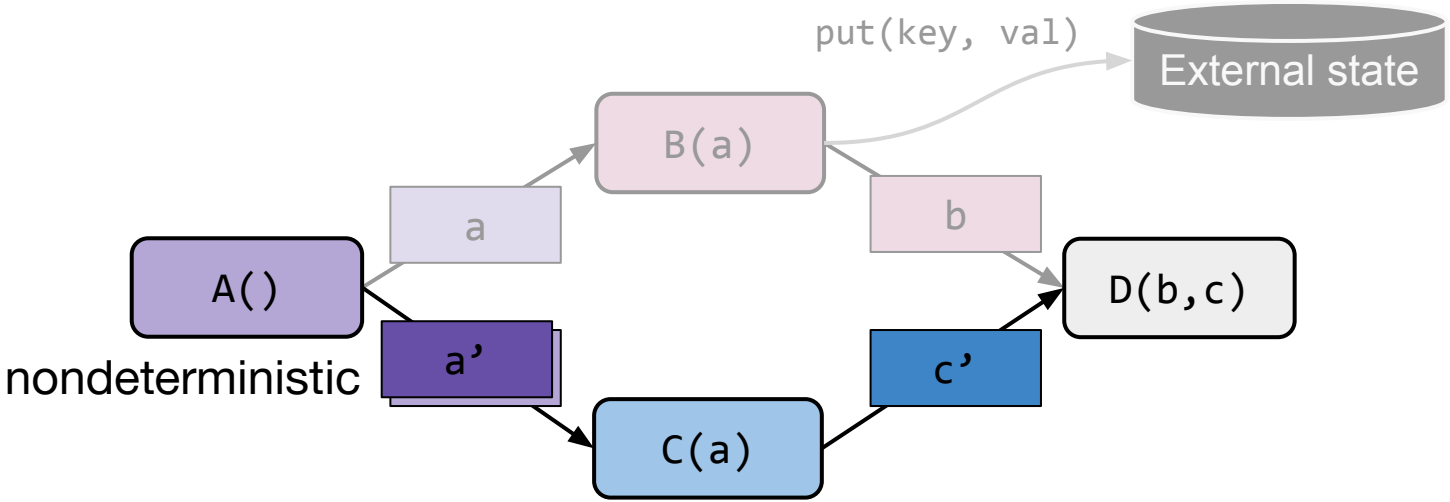
Nondeterminism and idempotence

What happens if we don't synchronously checkpoint?



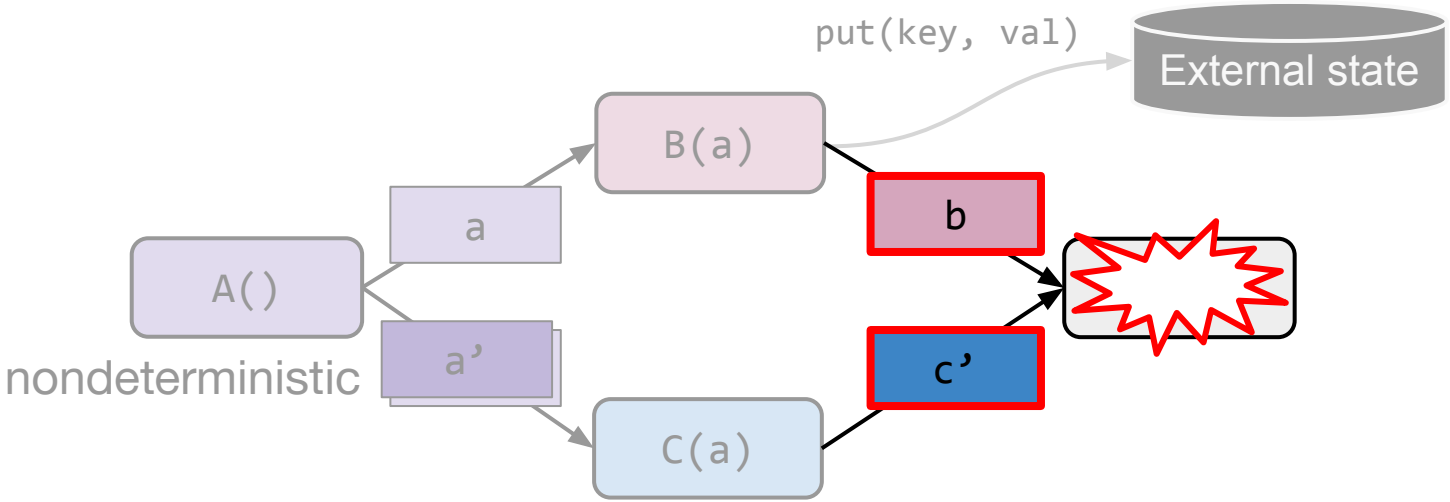
Nondeterminism and idempotence

What happens if we don't synchronously checkpoint?



Nondeterminism and idempotence

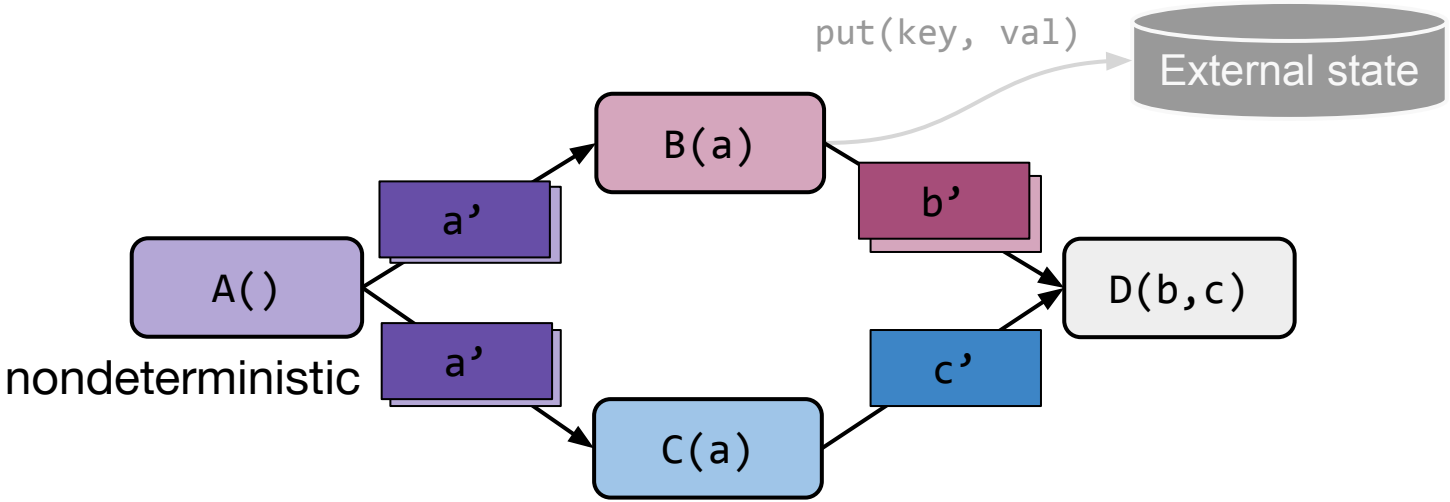
What happens if we don't synchronously checkpoint?



Rollback of **internal** outputs: Drop previous outputs and rerun the task

Nondeterminism and idempotence

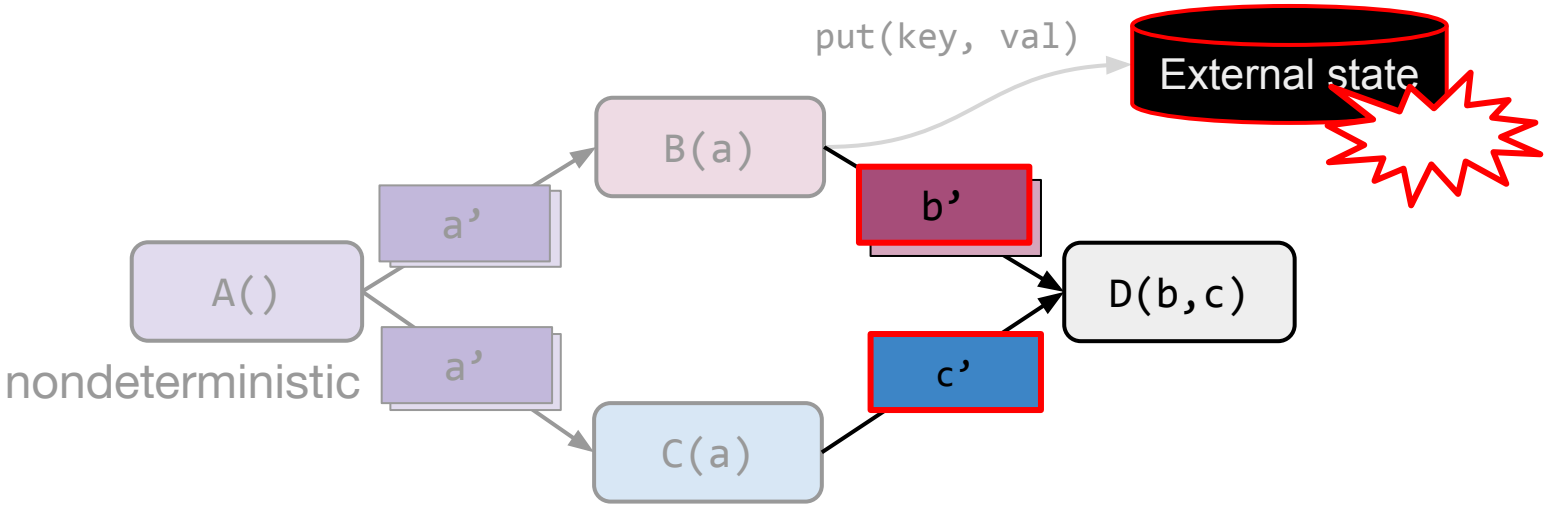
What happens if we don't synchronously checkpoint?



Rollback of **internal** outputs: Drop previous outputs and rerun the task

Nondeterminism and idempotence

What happens if we don't synchronously checkpoint?

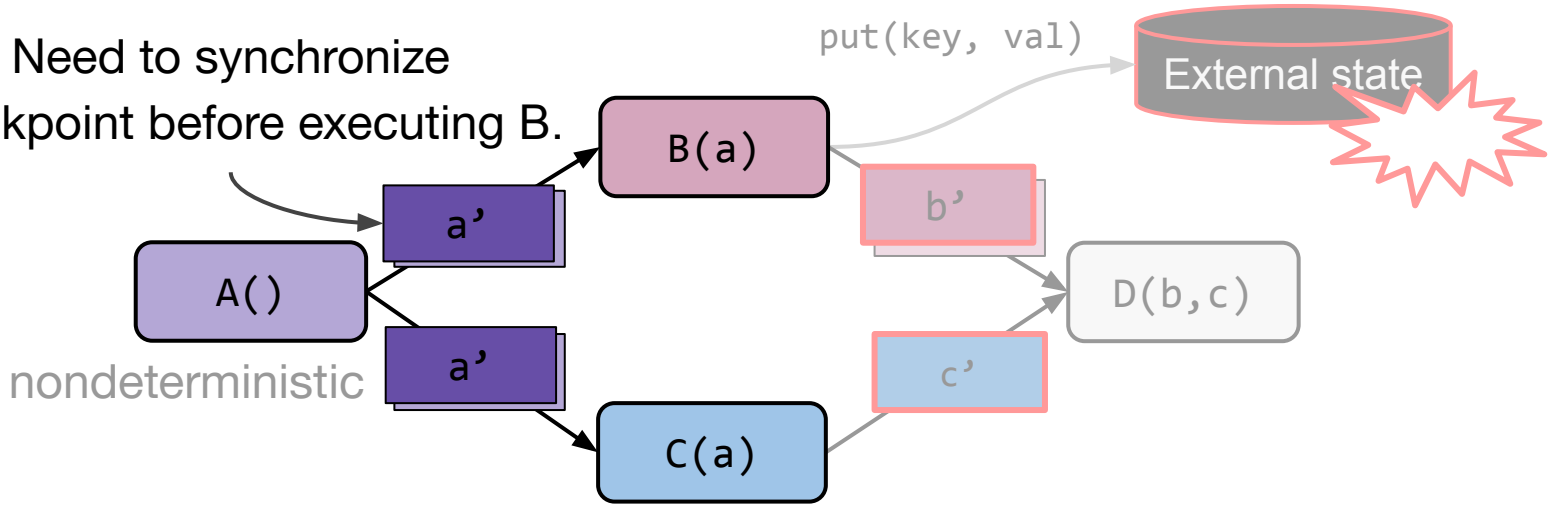


Rollback of **external** outputs: ???

Nondeterminism and idempotence

What happens if we don't <u>synchronously checkpoint</u>?

Need to synchronize checkpoint before executing B.



Rollback of **external** outputs: ???

Current workflow systems use a one-size-fits-all approach

Assume the worst:

1. Tasks may be nondeterministic.
2. Task may have external outputs.

Synchronously checkpoint outputs.
Simple and correct, but:

- Unnecessary depending on task semantics
- Slow if data is large

Goal: Can we build a **universal** workflow system that enables a **flexible** choice over the tradeoff between recovery and performance?

Exoflow: Decoupling the unit of execution from the unit of recovery.

Exoflow: Key ideas

How to decouple unit of recovery from unit of execution?

How to lower the execution overhead of recovery?

Choose recovery technique based on **annotations** for task semantics

Pass application data via **first-class references**

Avoid unnecessary checkpoint synchronization

Avoid unnecessary output materialization and/or copying

Related work

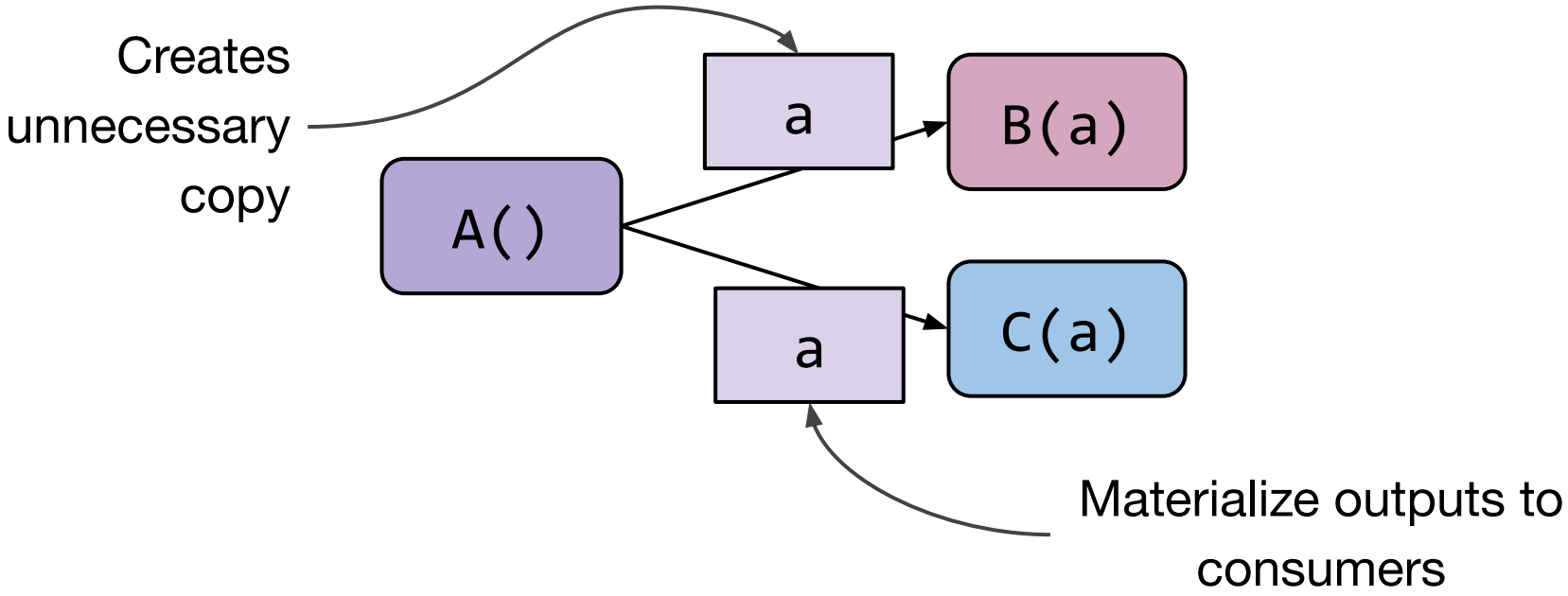
Current workflow systems couple unit of execution with unit of recovery.

- Execution unit: Task + idempotent APIs for external outputs
- Pick a single recovery method:
  - Data workflows: sync checkpoint+retry (Airflow, Kubeflow)
  - Serverless workflows: retry (AWS Step Functions), retry+rollback (Aft), or log+replay (Durable Functions, Beldi, BokiFlow)

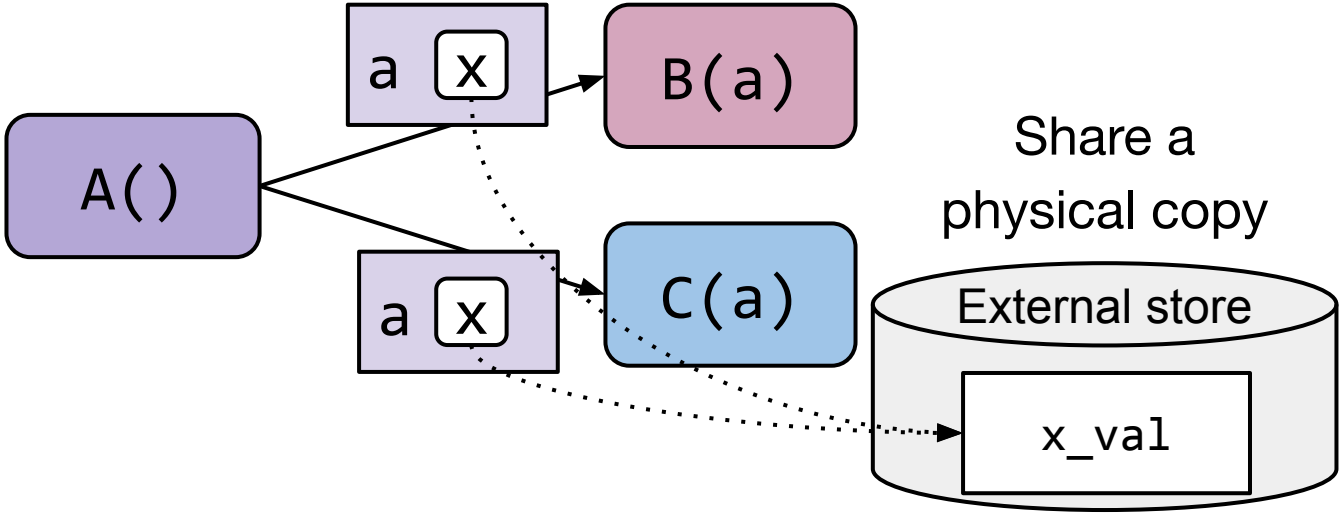
Systems that decouple unit of execution (messages) from unit of recovery (rollback prefix): Falkirk Wheel, DARQ.

- Lack references to abstract data movement
- Lack annotations to further improve recovery flexibility and efficiency

Challenge: Existing workflow systems must synchronously checkpoint all internal outputs



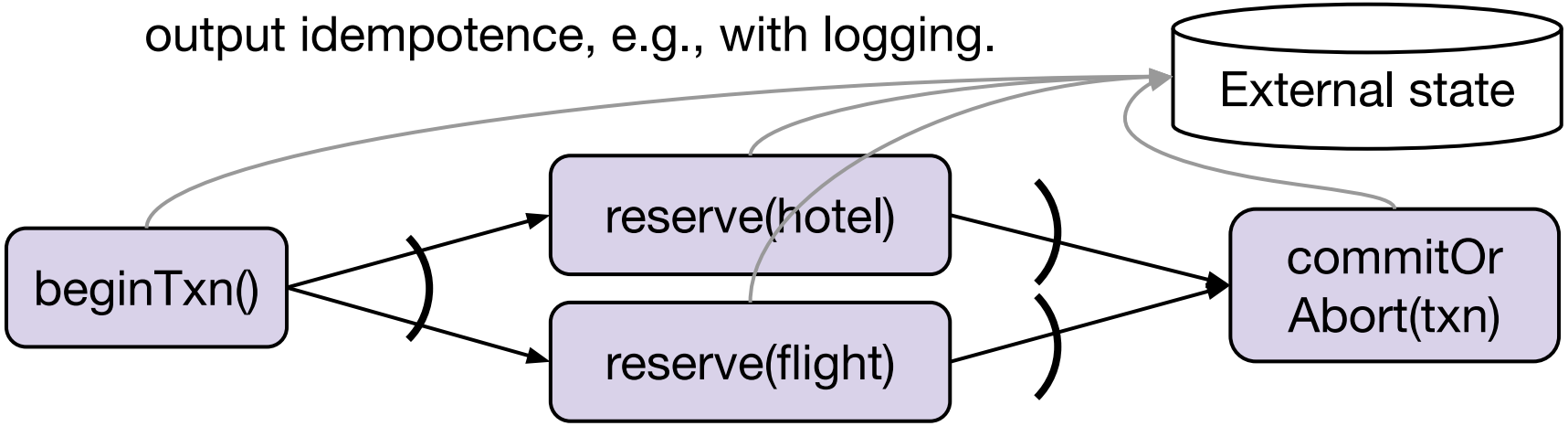
Solution: Use first-class references for more efficient data movement



Key idea: Pass by **reference** lets the execution backend decide how to pass the physical value.

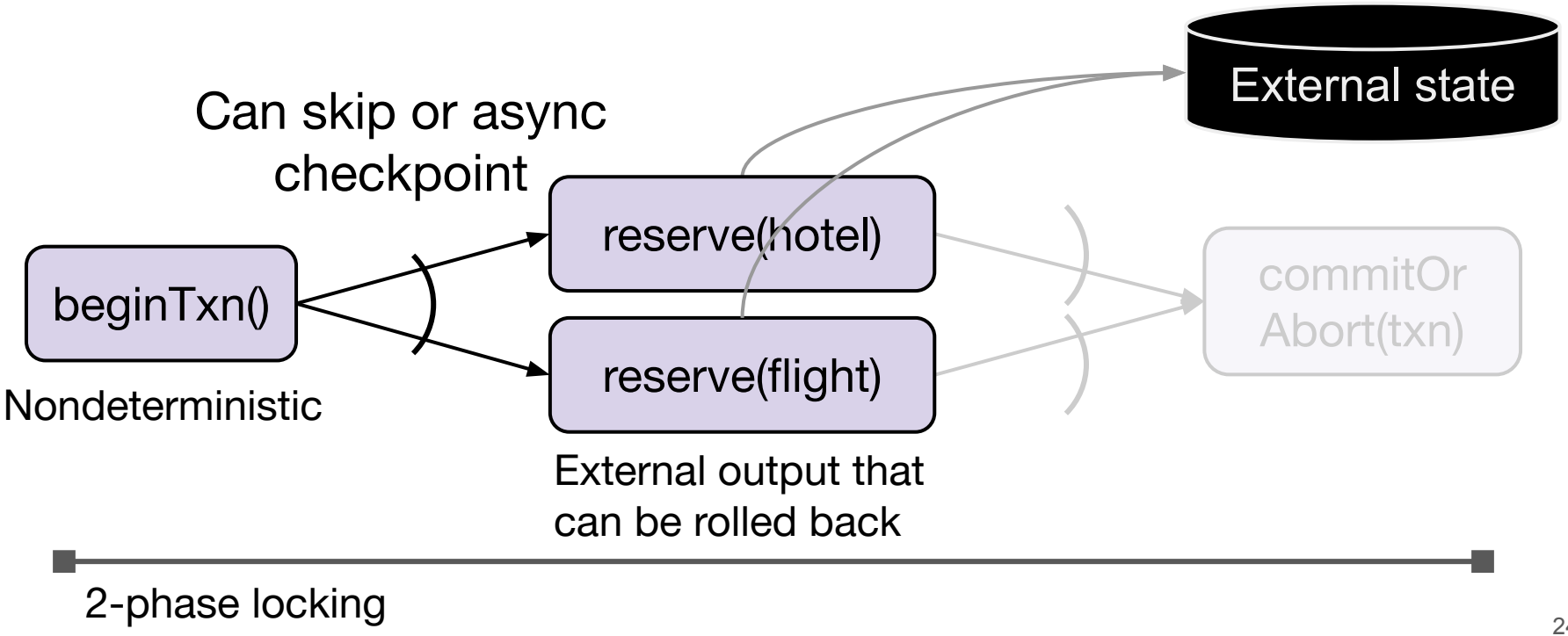
Challenge: Correctness for external outputs

Serverless workflow systems can provide external output idempotence, e.g., with logging.

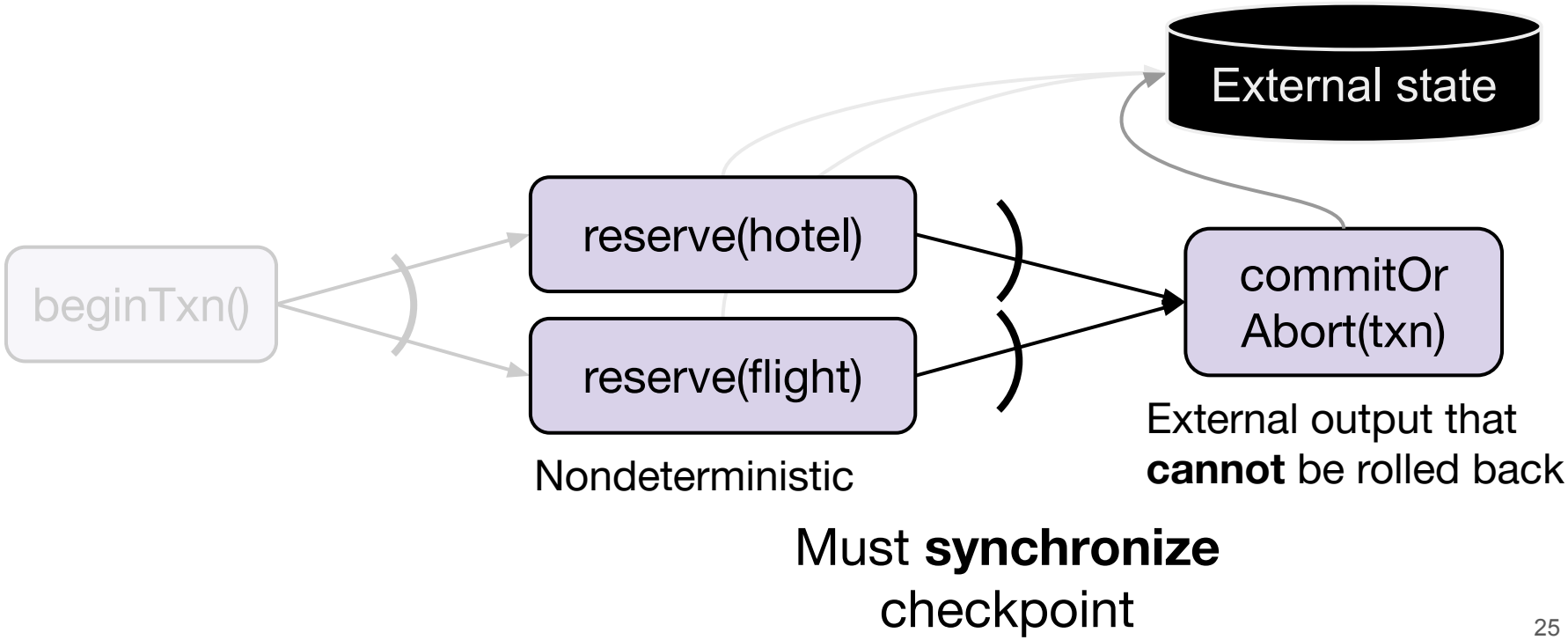


If tasks are nondeterministic, must checkpoint outputs *before* downstream tasks start.

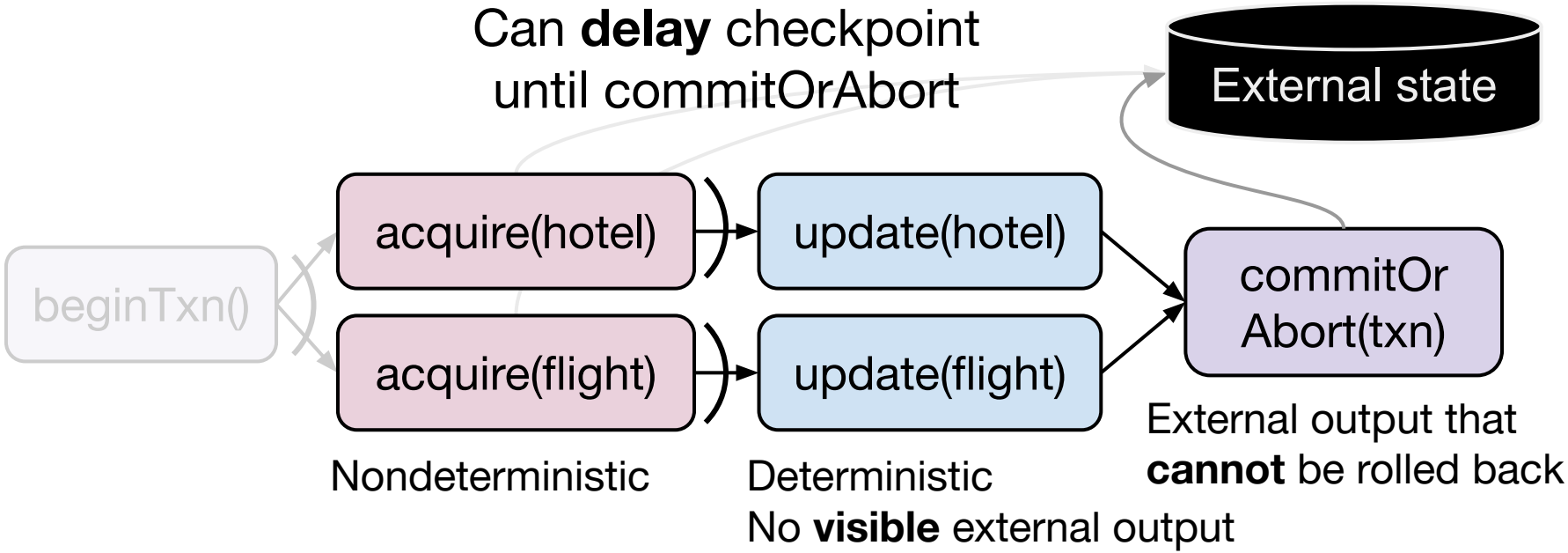
Solution: Task annotations to capture semantics



Solution: Task annotations to capture semantics

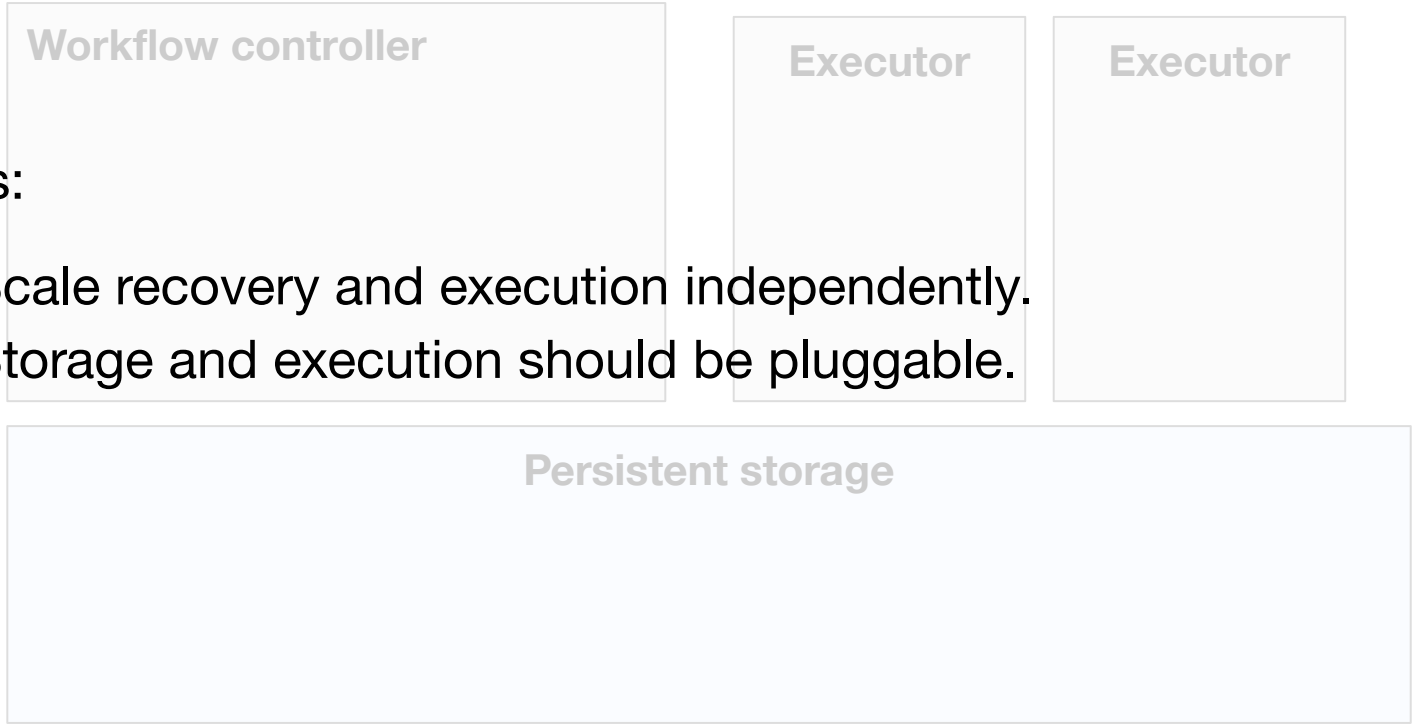


Solution: Task annotations to capture semantics



Key idea: Specify semantics *before* execution, lower overhead *during*.

Exoflow Architecture



Goals:

1. Scale recovery and execution independently.
2. Storage and execution should be pluggable.

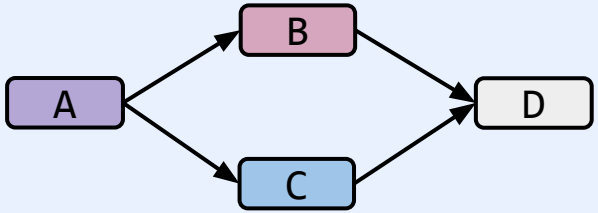
Exoflow Architecture

Workflow controller

Executor

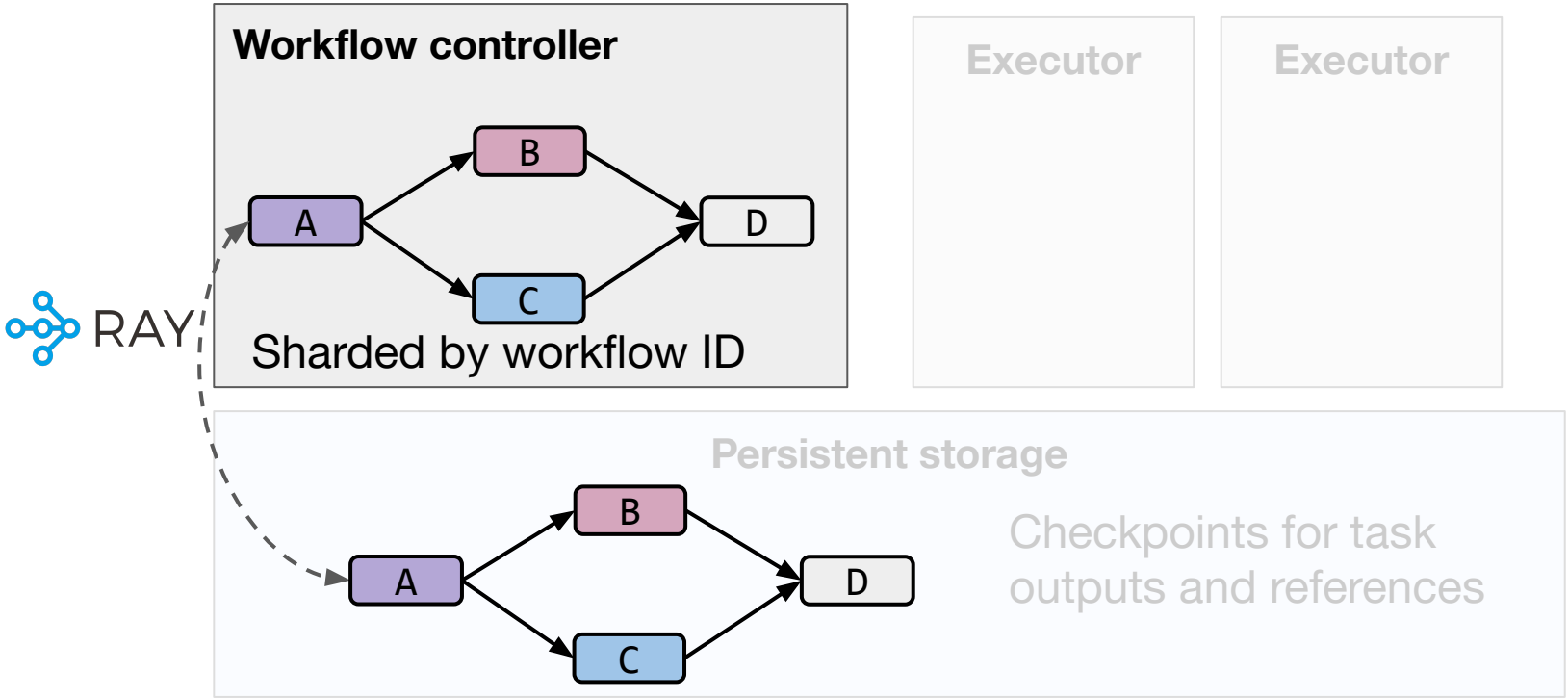
Executor

Persistent storage

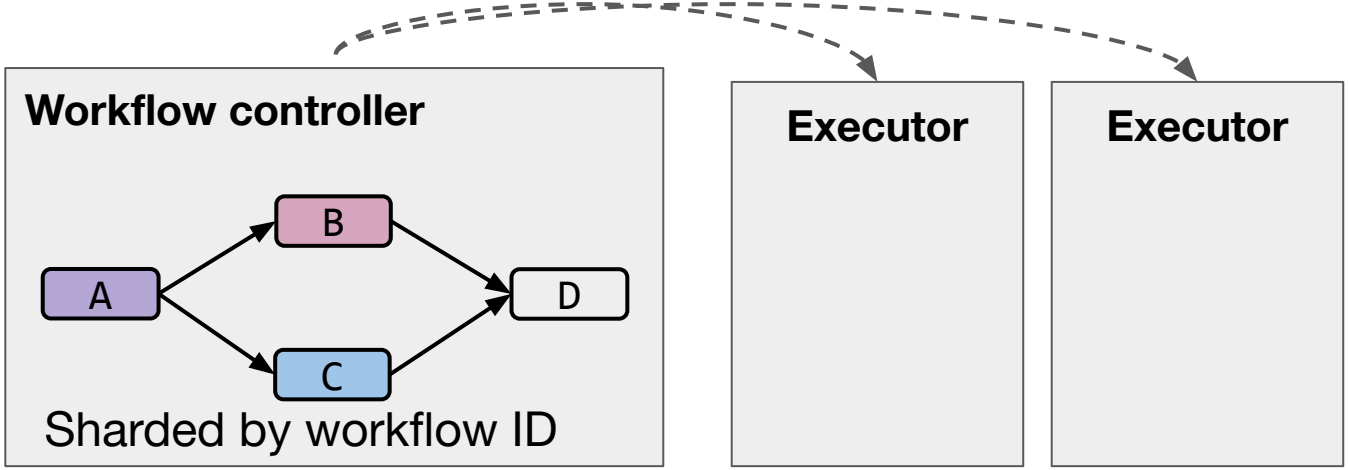


Checkpoints for task outputs and references

Exoflow Architecture

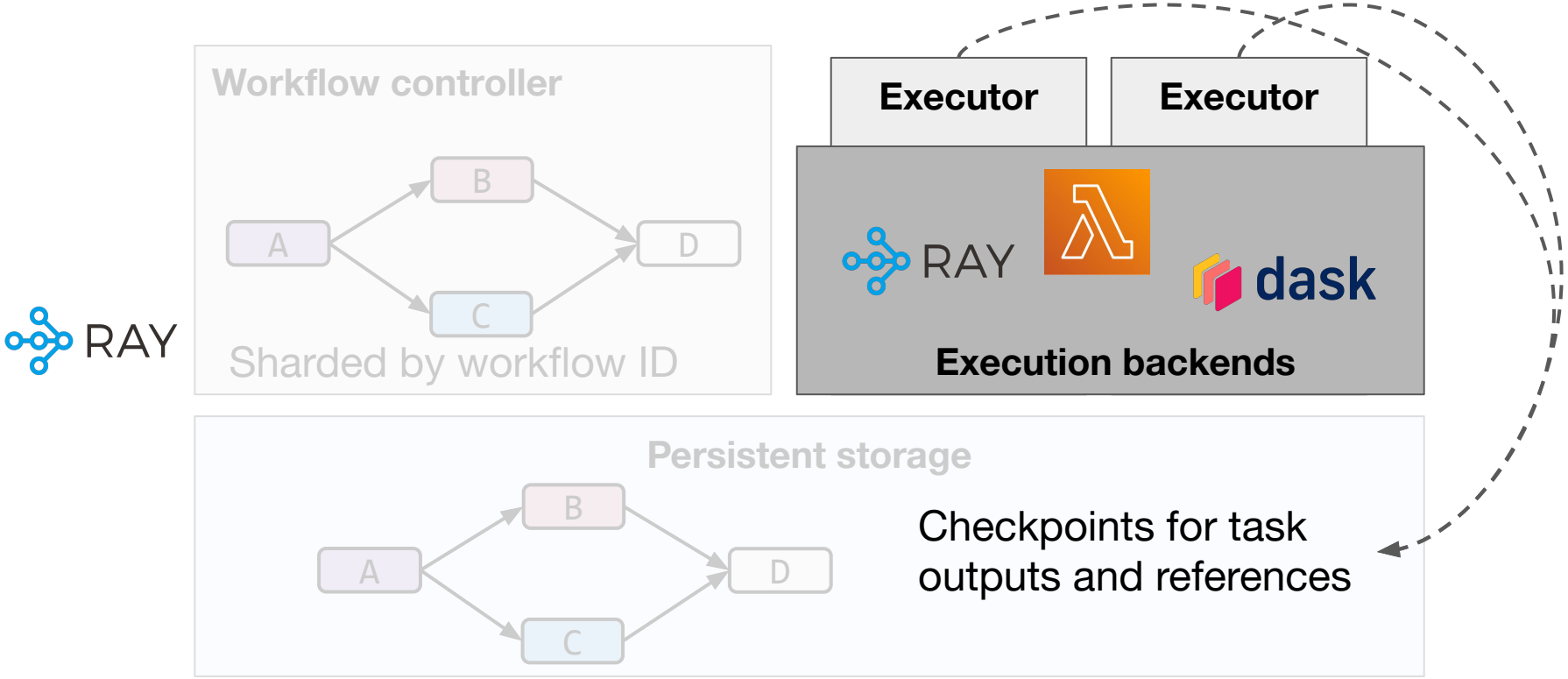


Exoflow Architecture

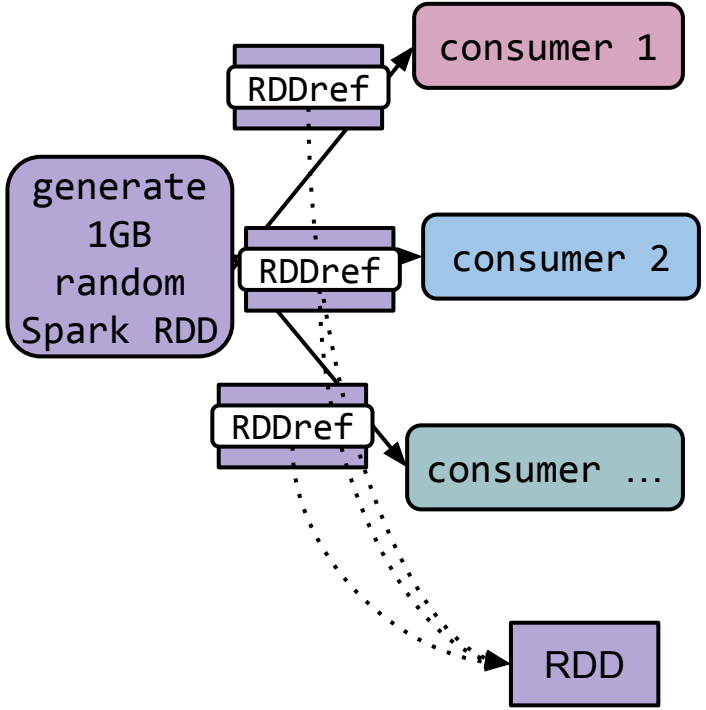


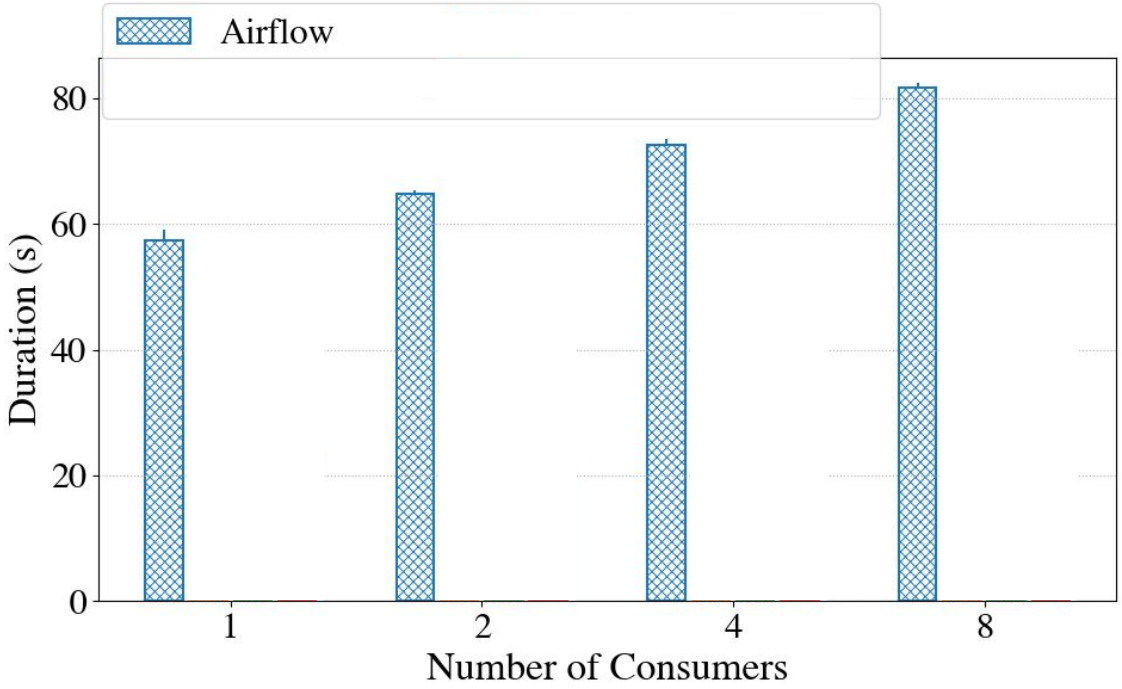
RAY

Exoflow Architecture

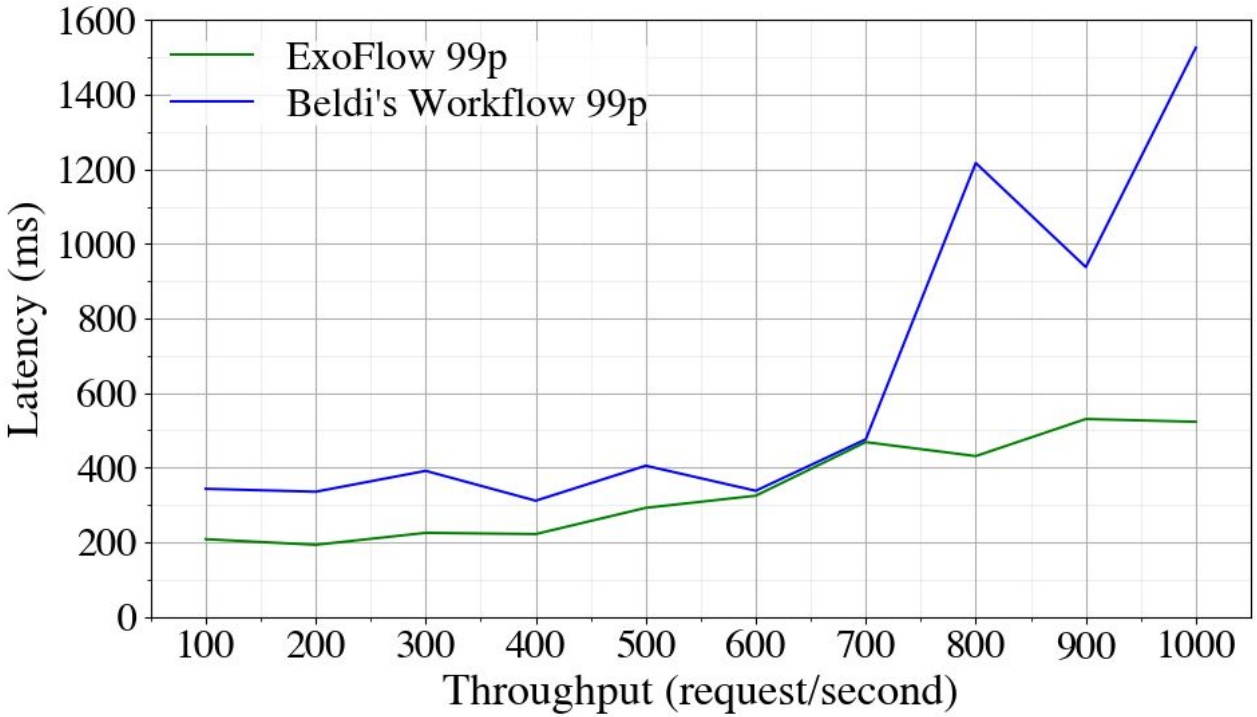


Microbenchmark: Simulating an ETL workflow



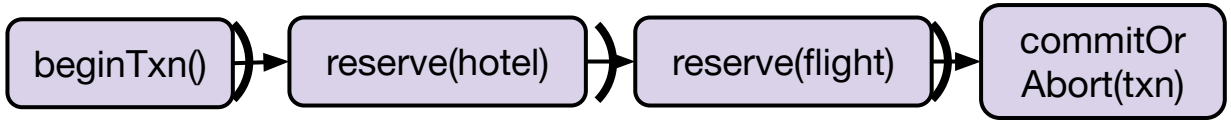


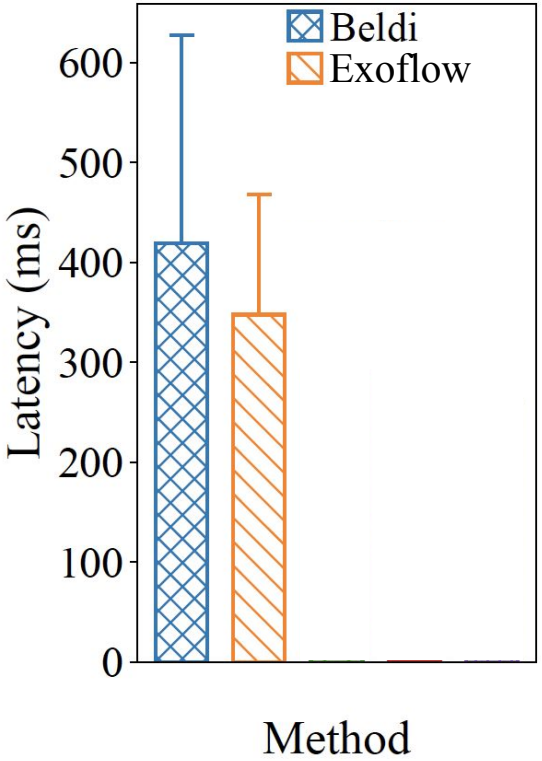
Benchmark: Microservice workflows[1] on AWS Lambdas



[1] Yu Gan et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. *ASPLOS 2019*.

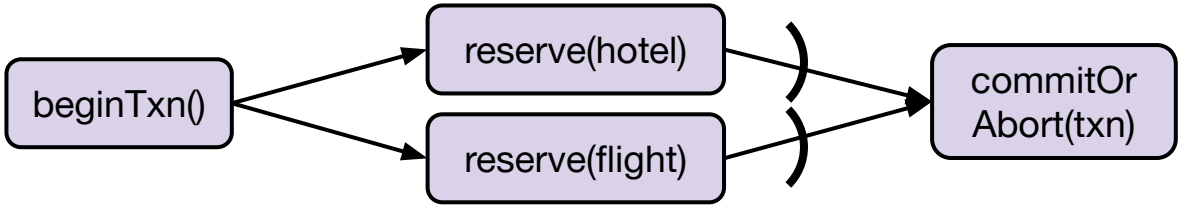
Benchmark: Microservice workflows[1] on AWS Lambdas

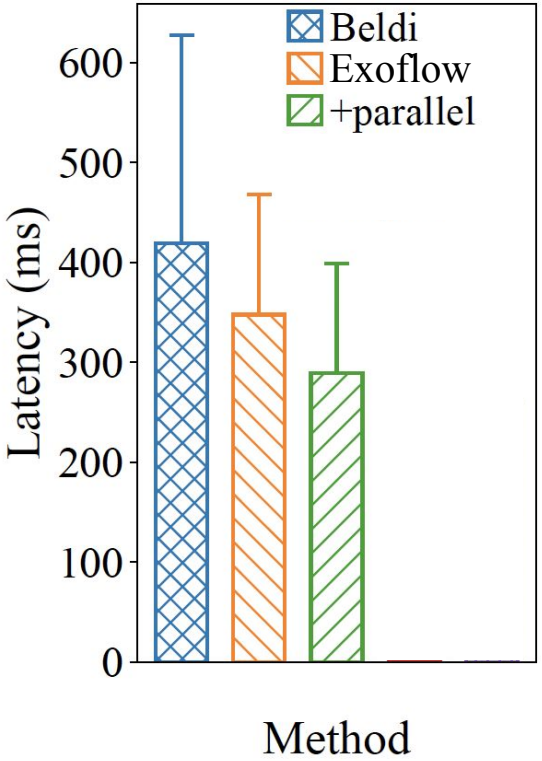




[1] Yu Gan et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. *ASPLOS 2019*.

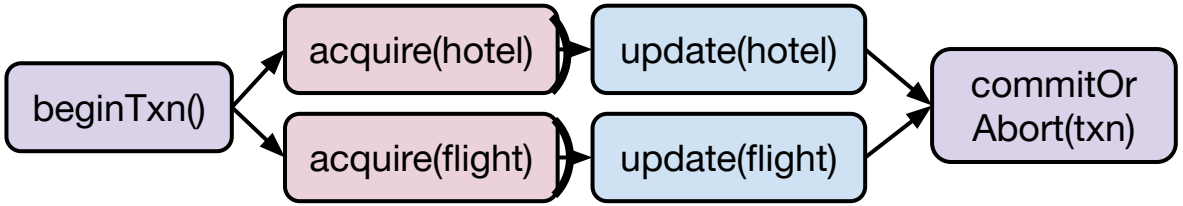
Benchmark: Microservice workflows[1] on AWS Lambdas



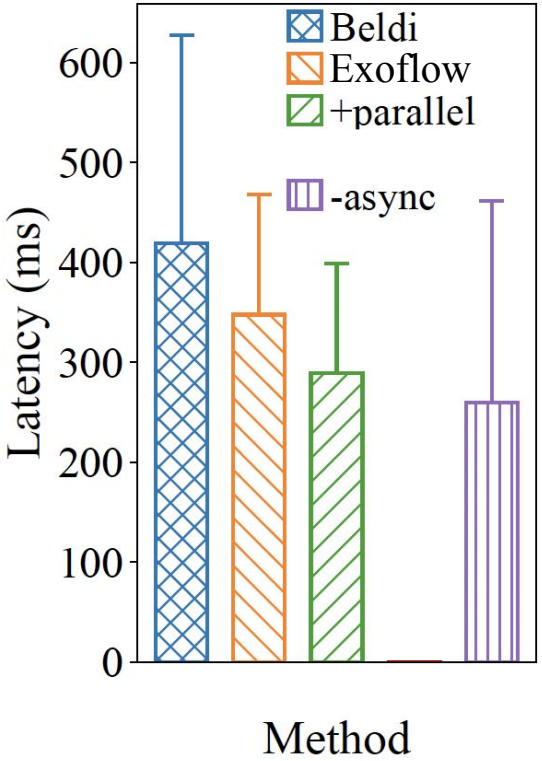


[1] Yu Gan et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. *ASPLOS 2019*.

Benchmark: Microservice workflows[1] on AWS Lambdas



"+async" **delays** checkpoint
until commitOrAbort



[1] Yu Gan et al. An open-source benchmark suite for microservices and their hardware-software implications for cloud & edge systems. *ASPLOS 2019*.

See the paper for…

Microbenchmarks and a wider variety of workloads

- Serverless transactions
- End-to-end ML pipeline
- Online-offline graph processing

Details on:

- References and annotations API
- Execution and recovery protocols

Conclusion

Exoflow: A universal workflow system that decouples the unit of execution from the unit of recovery.

Task annotations

First-class references

Give applications **flexibility** in performance vs. recovery.

Code: github.com/suquark/exoflow

Email: siyuan@cs.berkeley.edu (Siyuan)
swang@cs.berkeley.edu (Stephanie)

RAY

sky UC Berkeley