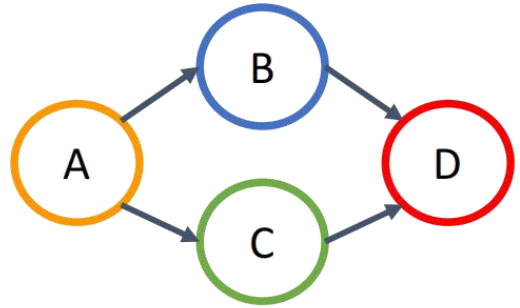
Doing More with Less: Orchestrating Serverless Applications without an Orchestrator

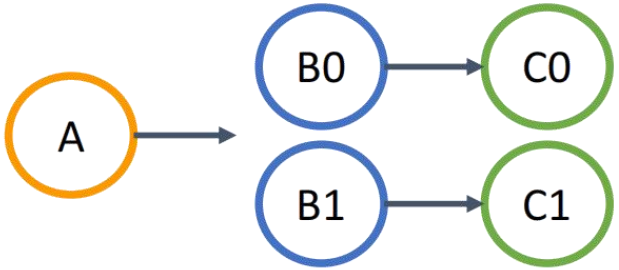
David H. Liu and Amit Levy, Princeton University; Shadi Noghabi and Sebastian Burckhardt, Microsoft Research

2024.6.14

How to build large applications consisting of many functions ?

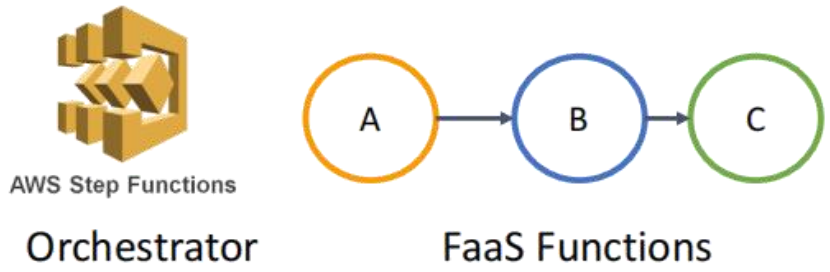
- Stateful patterns over stateless functions
  - E.g., aggregating the results of multiple functions
- At-least-once function execution
  - Multiple diverging outputs
  - Wasted runtime costs





Orchestrators

- Additional services to existing infrastructures
- Higher-level programming interfaces
- User-deployed or provider-hosted
- Logically centralized controller
  - Invoke functions and receive function results
  - Centralization simplifies stateful patterns and execution guarantees

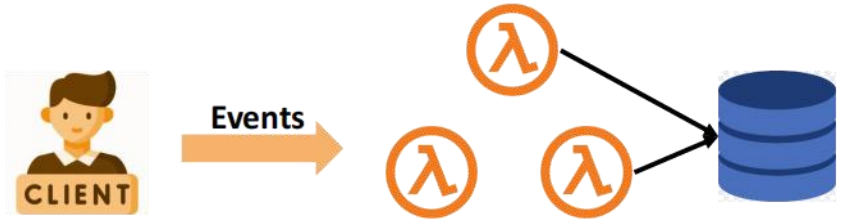


Orchestrators

- Compromise original benefits of serverless
  - E.g., freedom from server management, fine-grained billing
- Hosting costs of resources and engineering
- Limit application performance and programmability
  - Application-specific functionalities and optimizations

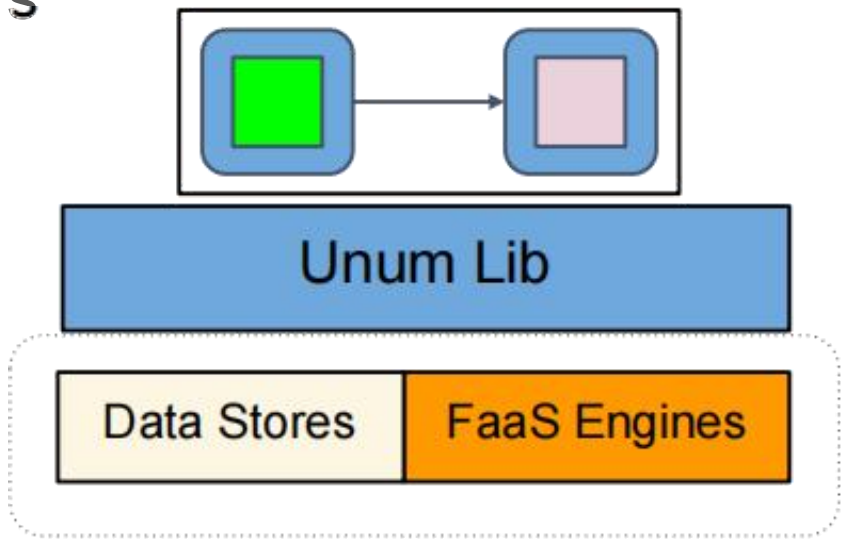
Application-Level Orchestration

- Original serverless benefits
- Automatically improve with underlying systems
- Providers: freedom from hosting
- Users:
  - Ability to implement application-specific functionalities and optimization
  - Easy migration and alleviate vendor lock-in

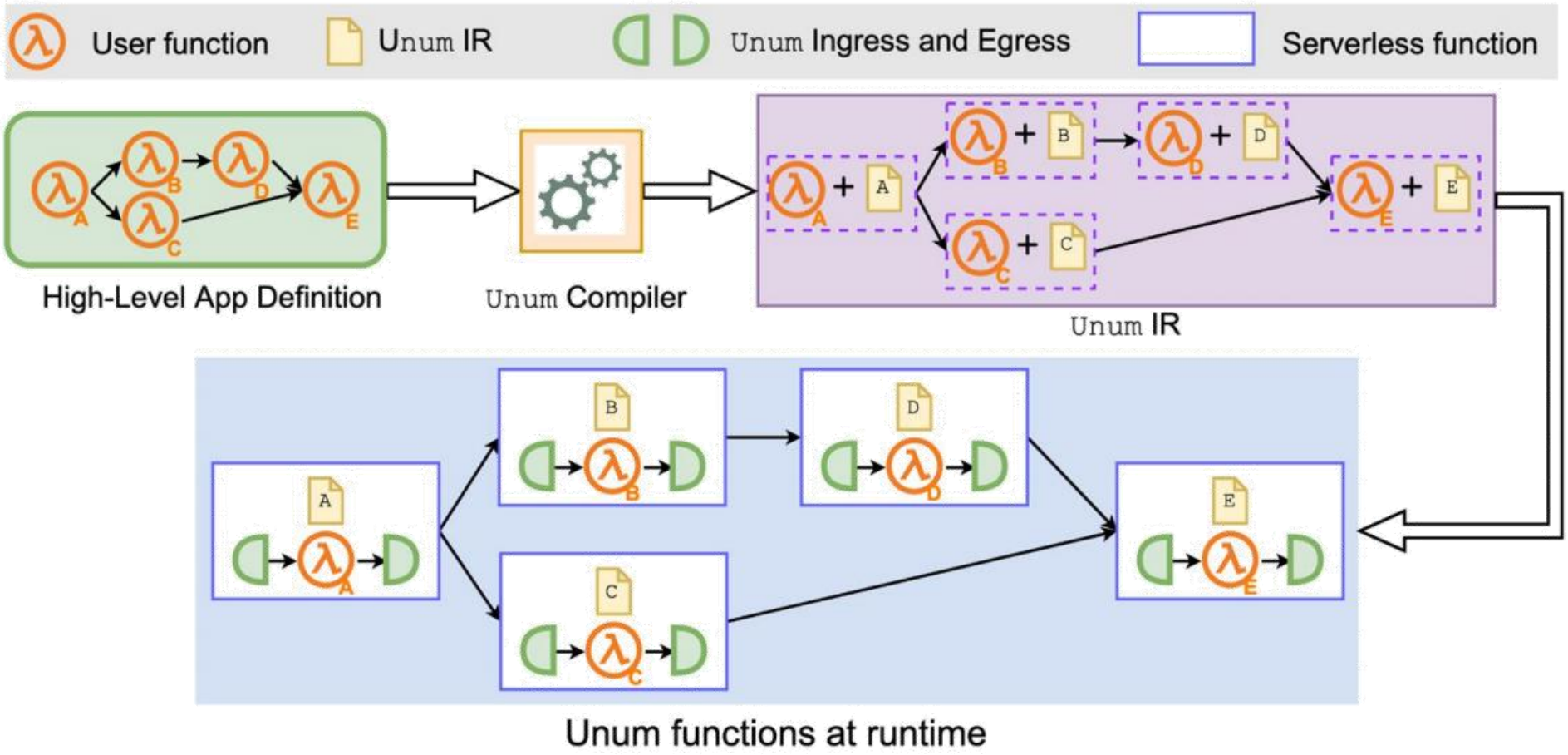


Unum: A Library Orchestrator

- No new services or changes to existing APIs
- Support programs from AWS Step Functions
- Exactly-once semantics with checkpoints
- Coordinate stateful patterns via data stores
- Better performance & costs
- Migration with no app code changes



Unum: A Library Orchestrator



Unum: A Library Orchestrator

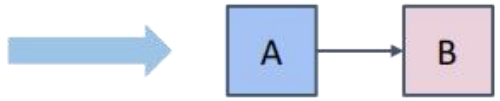
- Where to execute orchestration logic
  - Unum Intermediate Representation: Decentralized Orchestration
- How to support stateful patterns such as aggregation
  - Coordination via Data Store
- How to ensure strong execution guarantees
  - Checkpoints via Data Store

Unum IR: Decentralize Orchestration

- Single orchestrator function is inefficient and limits application sizes
  - Double billing
  - Time-out limits application sizes
- Decentralize orchestration logic
  - Local portion of orchestration
  - Decentralize at compile time
- Represent high-level application definitions as directed graphs
  - Vertices are functions
  - Edges are transitions

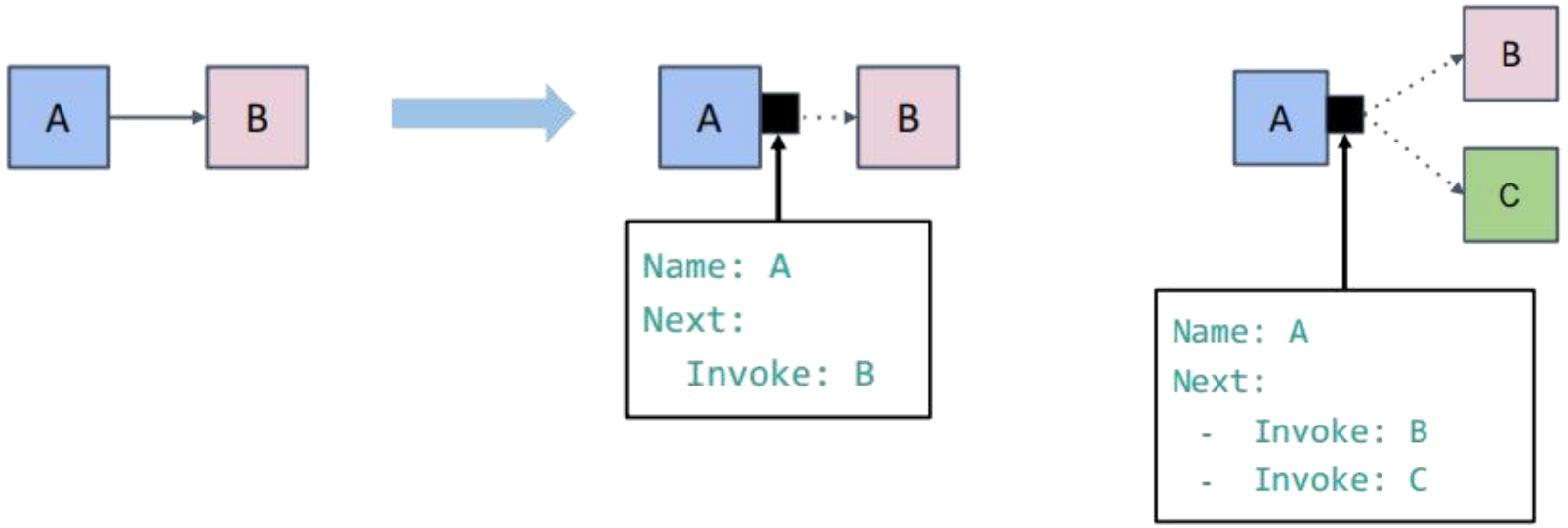
```
"States": {
    "A": {
        "Type": "Task",
        "Next": "B"
    },
    "B": {
        "Type": "Task",
        "End": true
    }
}
```

Step Functions



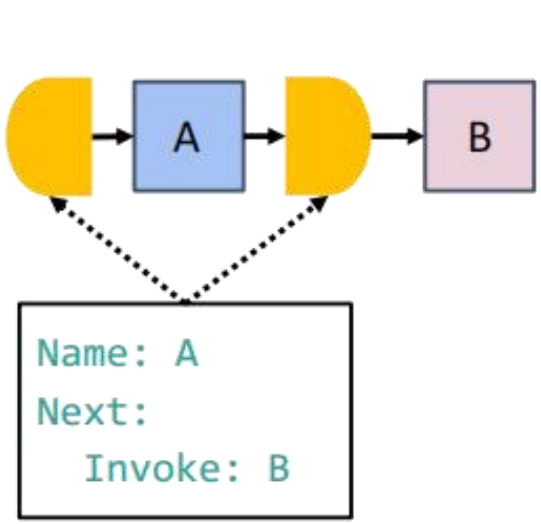
Unum IR: Decentralize Orchestration

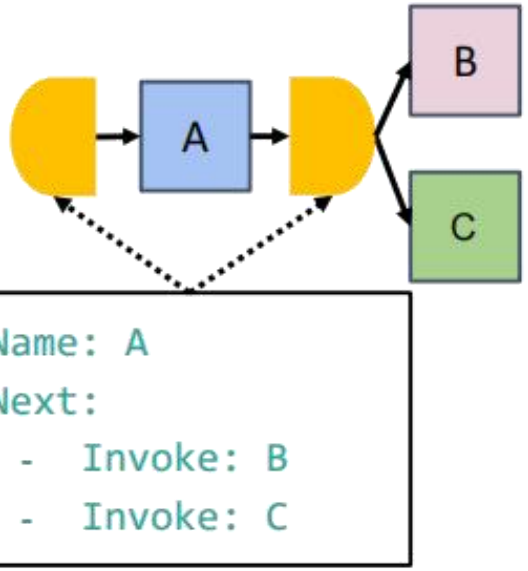
- Assign edges to tail nodes
  - Invoking the head node
  - Passing outputs to the head node as input
- Unum IR encodes edges as sequences of instructions



Unum IR: Decentralize Orchestration

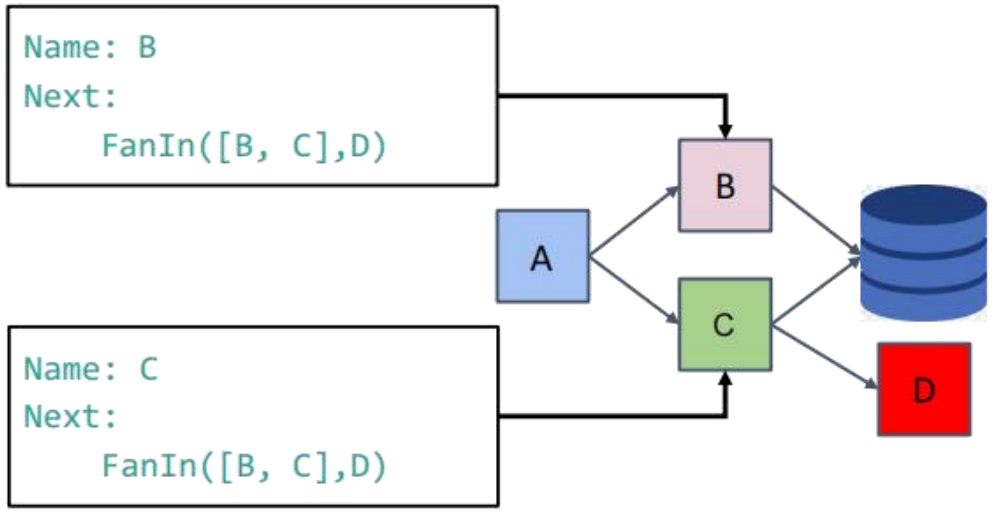
- Unum runtime executes IR instructions in-situ with user code
- The Unum standard library supports all patterns in Step Functions





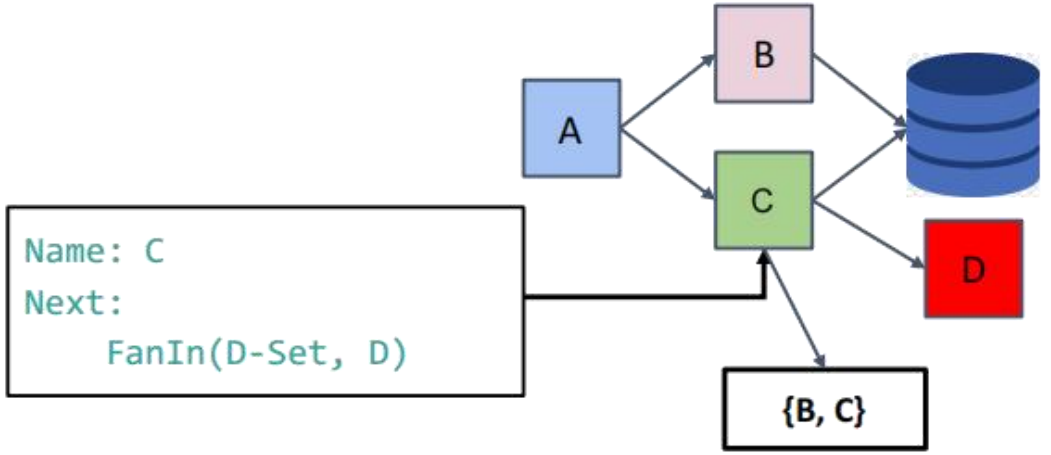
Coordination & Aggregation

- Tail nodes write results into the data store after completion
- Each tail node checks the completion of other tail nodes by reading from the data store
- Any one of the tail functions can invoke the head node
- Functions stay short-lived



Coordination & Aggregation

- Strongly consistent data stores to avoid missing invocations
- Each tail node performs
  - 1 Write
  - N - 1 Reads
- Coordination Set
  - Tail nodes additionally add their names to a set data structure
  - One coordination set per head node
  - Optimize data read volume
    - 1 read vs N-1 reads



Exactly-Once Execution Semantics

- Unum runtime checkpoints exactly one execution of a node
  - Checkpoints contain user code results
  - A checkpoint uniquely identifies a node
  - Concurrent-but-slower executions discard results
- Only checkpointed data is propagated downstream
- Bypass user code if a checkpoint already exists

Implementation

- A standard library that includes
  - Runtime for AWS using Lambda and DynamoDB
  - Runtime for Google Cloud using Cloud Functions and Firebase
- Toolchain
  - Compiler for Step Functions to Unum IR
  - Linker that packages the user functions with the target platform’s runtime
- library
  - CLI tool for deploying applications

```
$ unum-cli compile
$ unum-cli build
$ unum-cli deploy
```

Evaluation

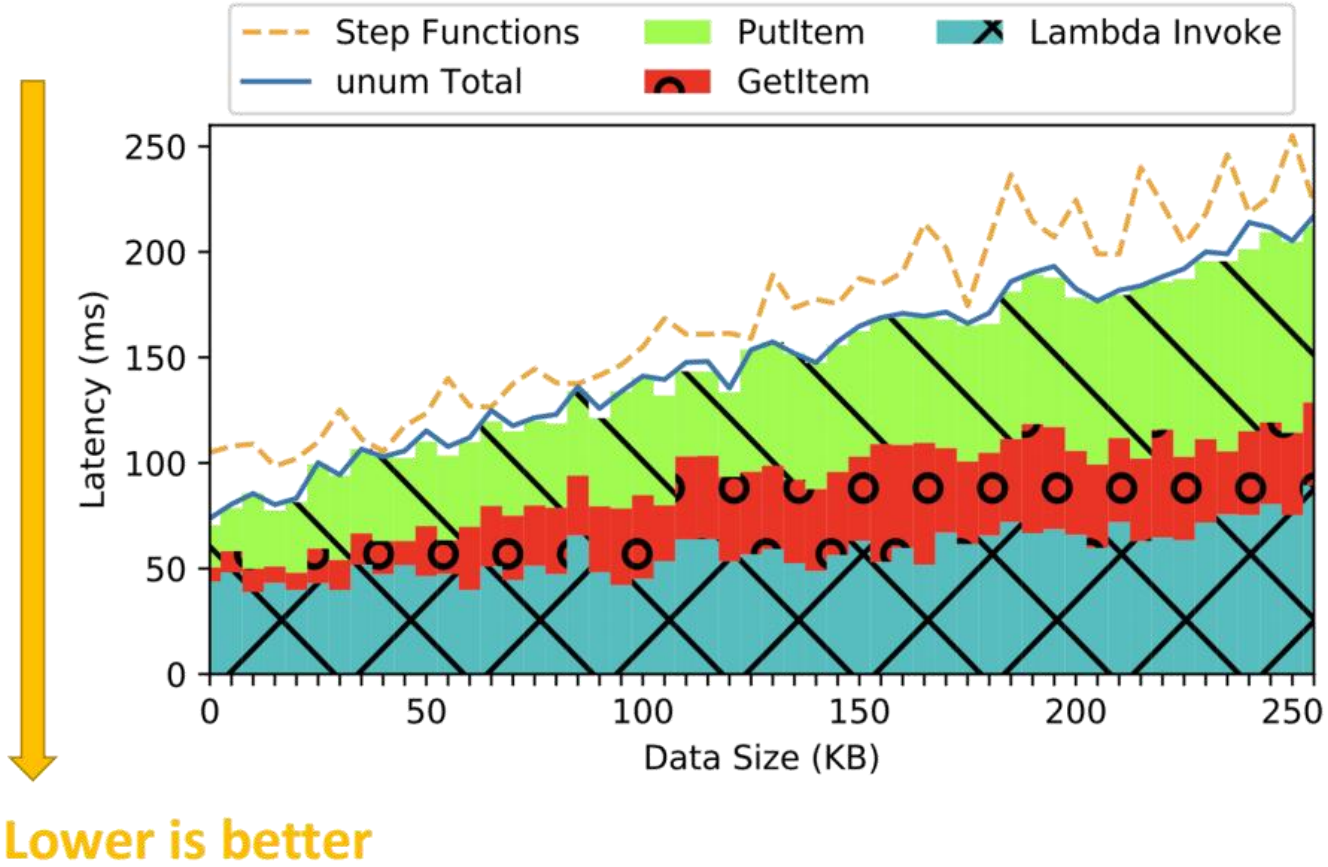
- Compare against Step Functions on AWS
- All applications are written as Step Functions
- All lambdas are similarly configured
- Unum applications are compiled from Step Functions definition
- Unum experiments use DynamoDB with on-demand provisioning

Evaluation: Single Transition in a Chain

- Latency of a single transition
  - From the end of the first function's user code to just before user code starts running in the second function
- Unum:
  - 1 storage write to create checkpoint
  - 1 lambda invoke
  - 1 storage read to check checkpoint existence
- Step Functions:
  - Timestamp between 1st function return and 2nd function start

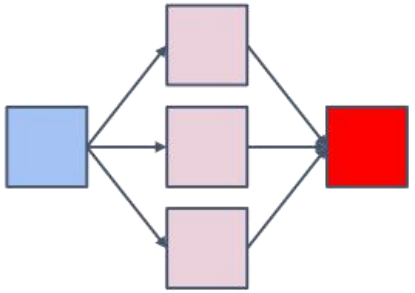
Evaluation: Single Transition in a Chain

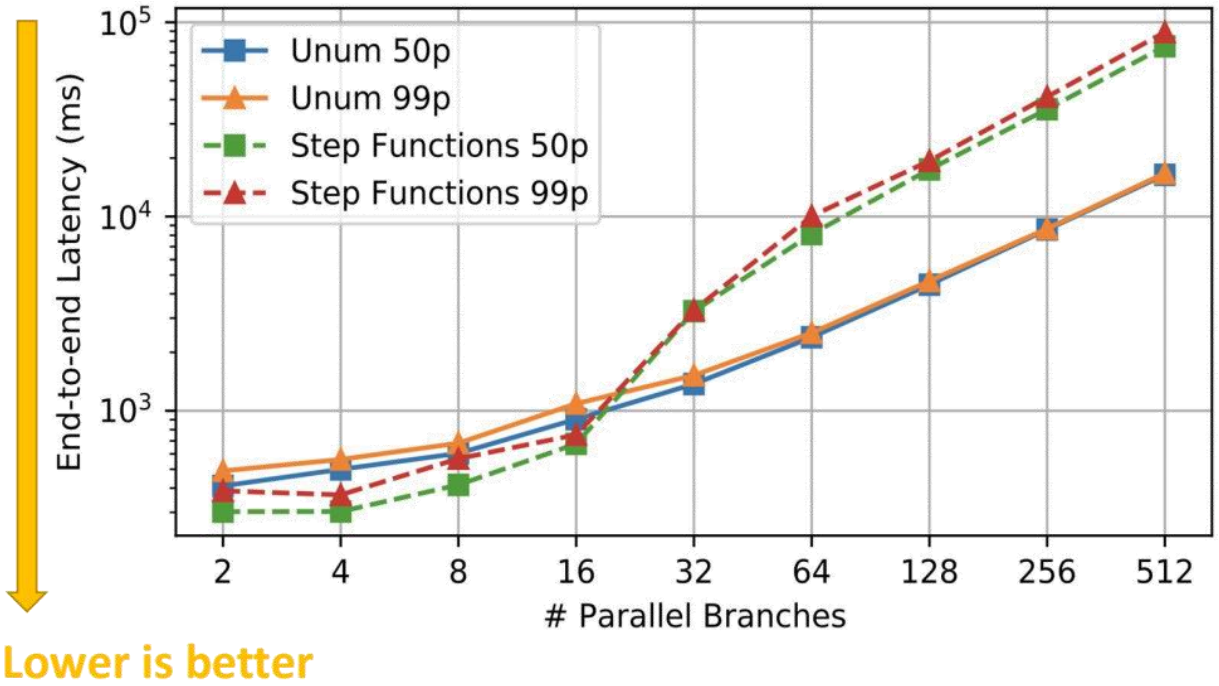
- Unum is consistently faster
- Storage writes, reads and Lambda invoke API calls make up the majority of the latency overhead



Evaluation: Parallel Performance

- End-to-end latency of fan-out and fan-in
- Varying the number of parallel branches
- Unum incurs a modest overhead (up to 200ms) at low branching degree
- Unum is up to 4x faster at higher branching degrees





Evaluation: Applications

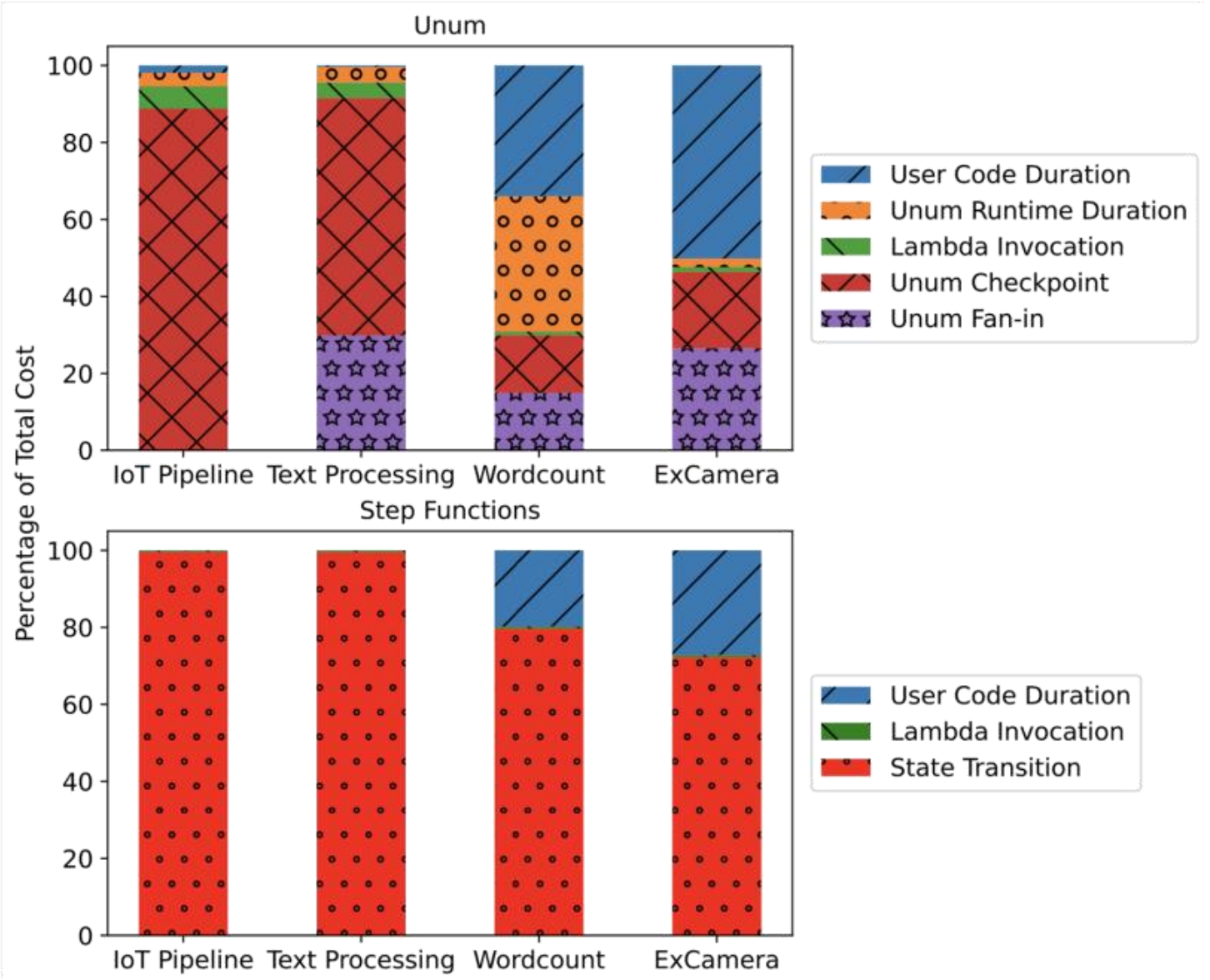
App	*Pattern(s)*	*Description*
IoT Pipeline	Chain	Process thermostat data and decide HVAC adjustment
Text Processing	Chain, Fan-out, Fan-in	Preprocess a social media post (from DeathStarBench)
Wordcount	Fan-out, Fan-in	MapReduce word count
ExCamera	Chain, Fan-out, Fan-in, Fold	Parallel video encoding

	Latency (seconds)			**Costs ($ per 1 mil. executions)**		
App	*Unum-aws*	*Unum-gcloud*	*Step Functions*	*Unum-aws*	*Unum-gcloud*	*Step Functions*
IoT Pipeline	0.12	0.81	0.23	$12.38	$6.3	$112.02
Text Processing	0.52	3.56	0.55	$60.42	$31.7	$225.29
Wordcount	408.88	484.12	898.56	$13,433.67	$11,727.3	$18,141.19
ExCamera	84.52	122.63	98.42	$62,684.29	$51,617.2	$114,633.13

- Unum is consistently faster and cheaper
- Up to 2x faster and 9x cheaper.

Evaluation: Applications

- Step Functions makes up the majority of the costs
- Unum: DynamoDB reads and writes make up the majority of the costs



Summary

- Decentralized Orchestration
- Preserved Serverless Benefits
- Exactly-Once Execution Guarantees
- Application-Level Coordination
- Scalability and Flexibility
- Seamless Integration and Migration
- Enhanced Developer Experience