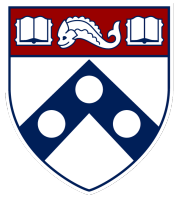
Fault-tolerant and Transactional Stateful Serverless Workflows

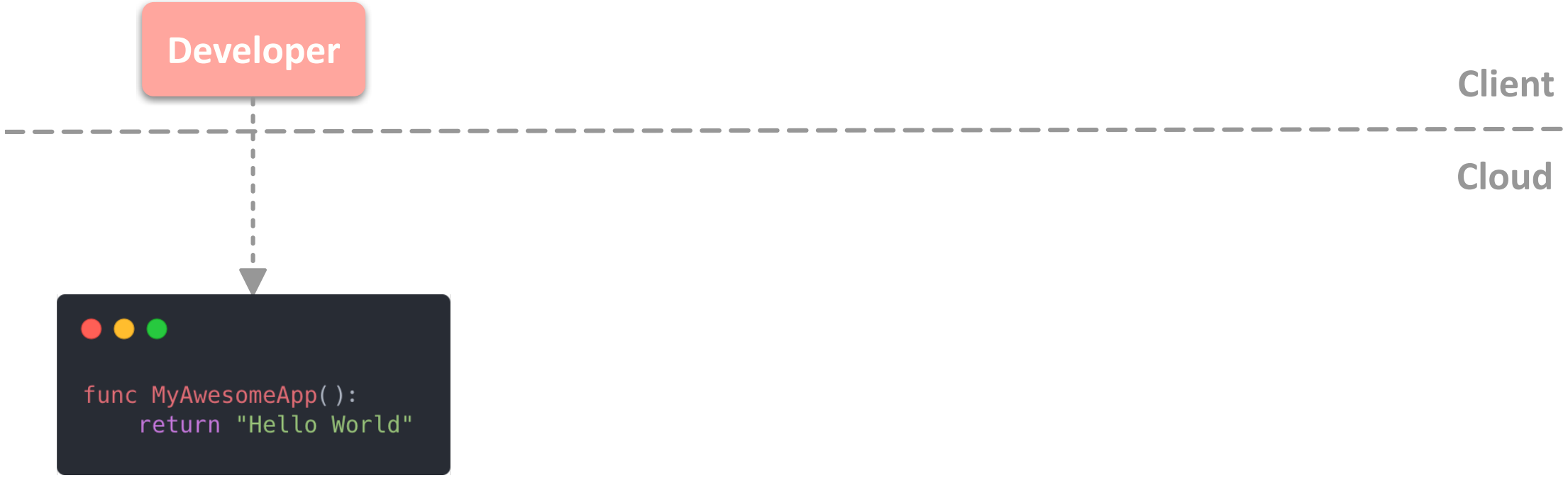
Haoran Zhang, Adney Cardoza, Peter Baile Chen
Sebastian Angel, Vincent Liu



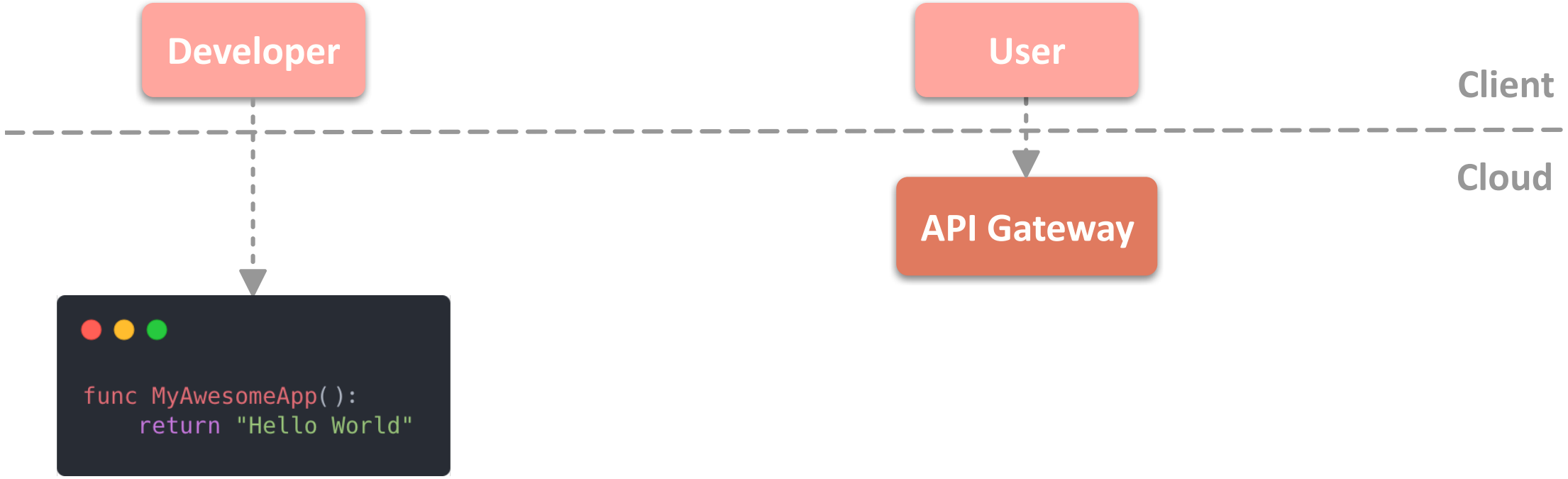
Penn
University of Pennsylvania

What is serverless?

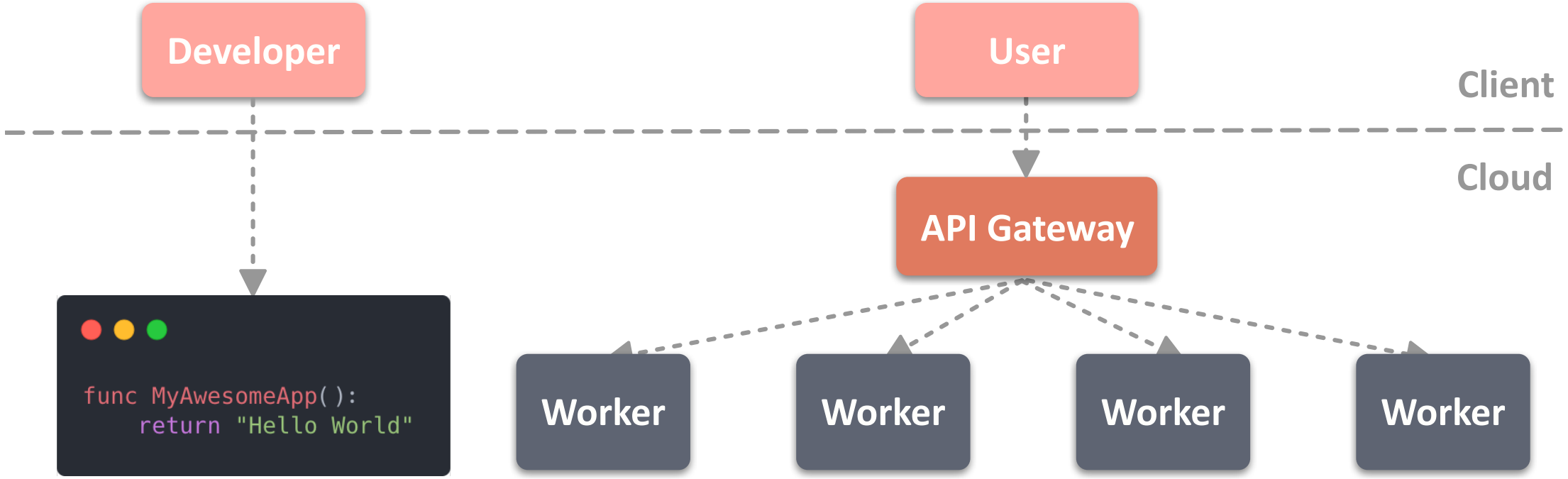
What is serverless?



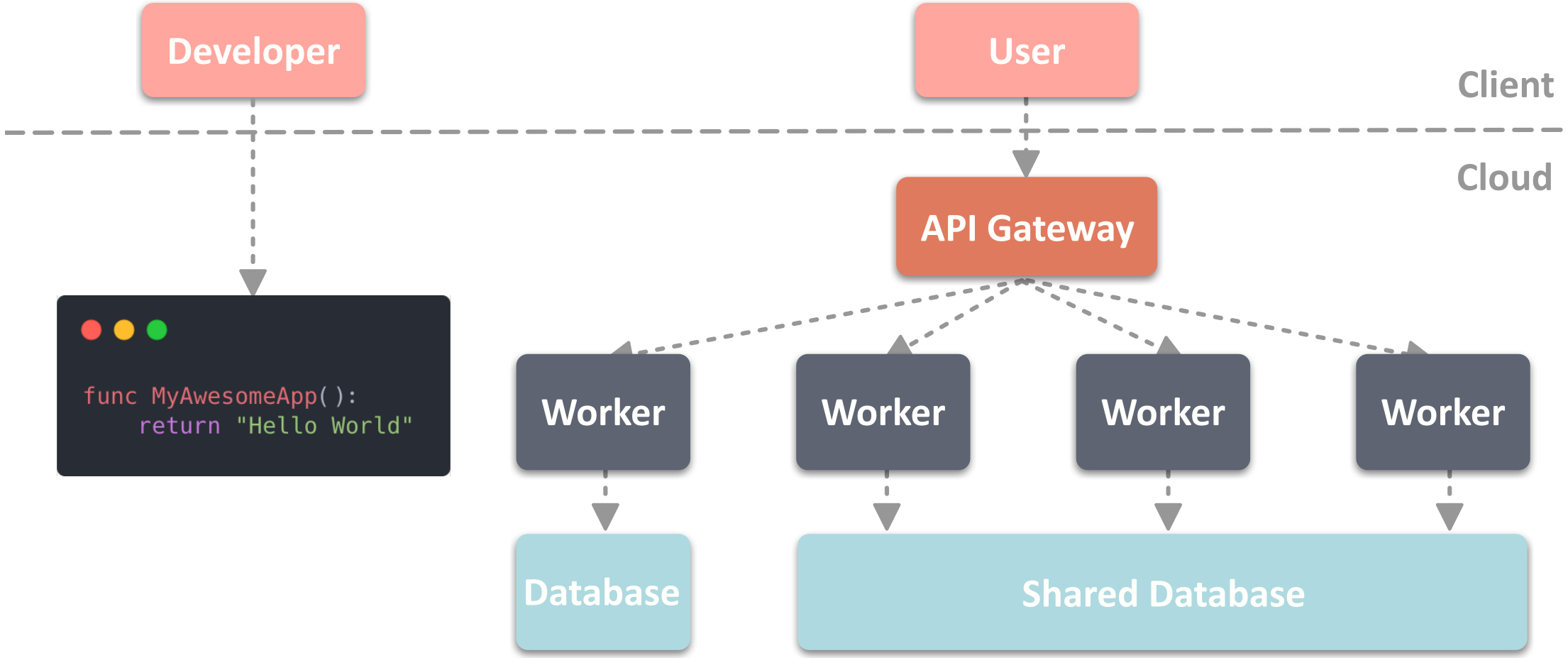
What is serverless?



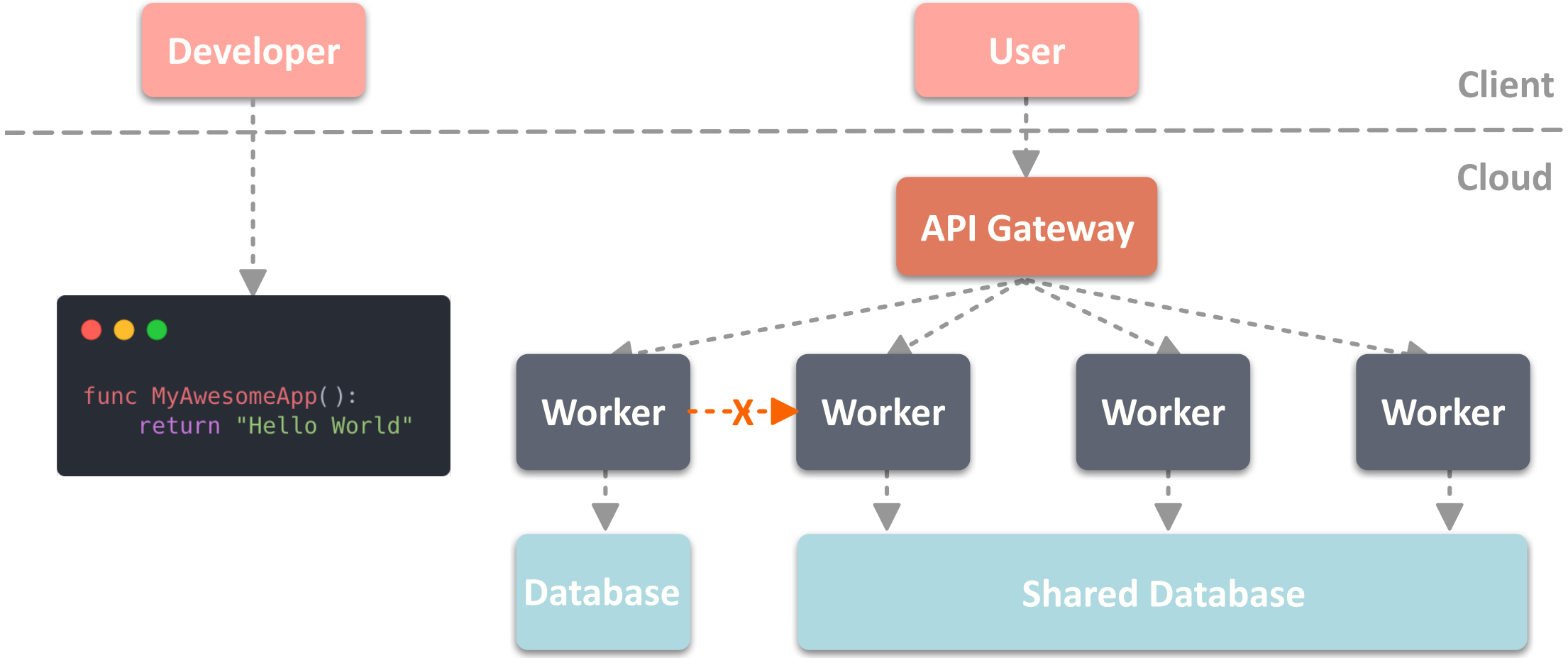
What is serverless?



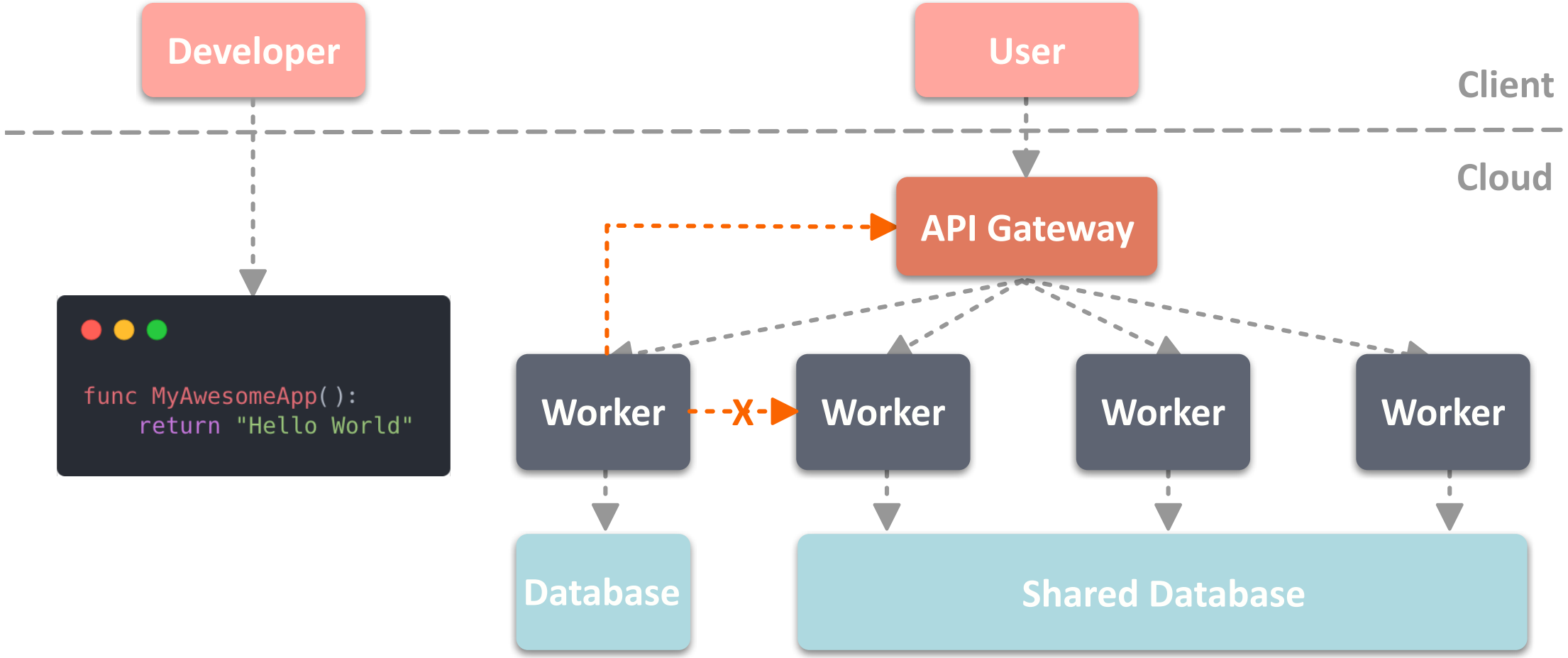
What is serverless?



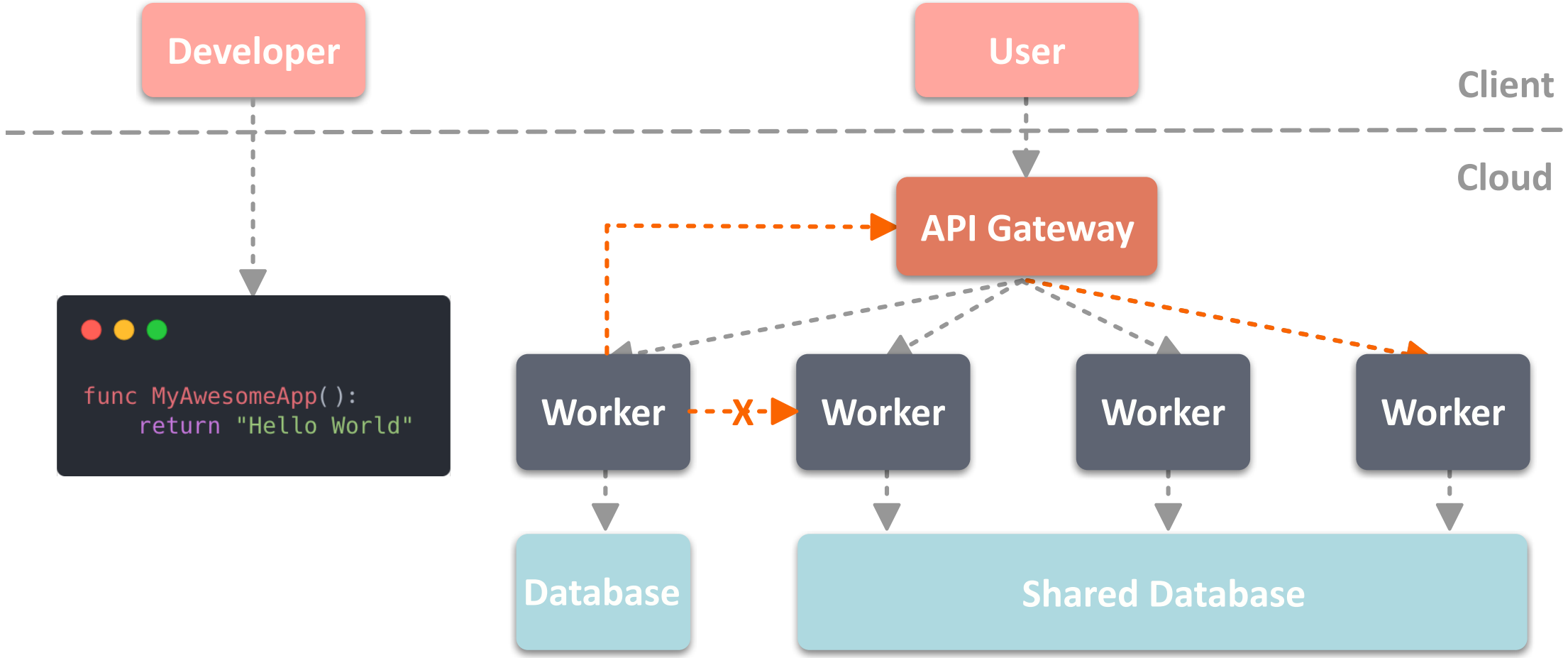
What is serverless?



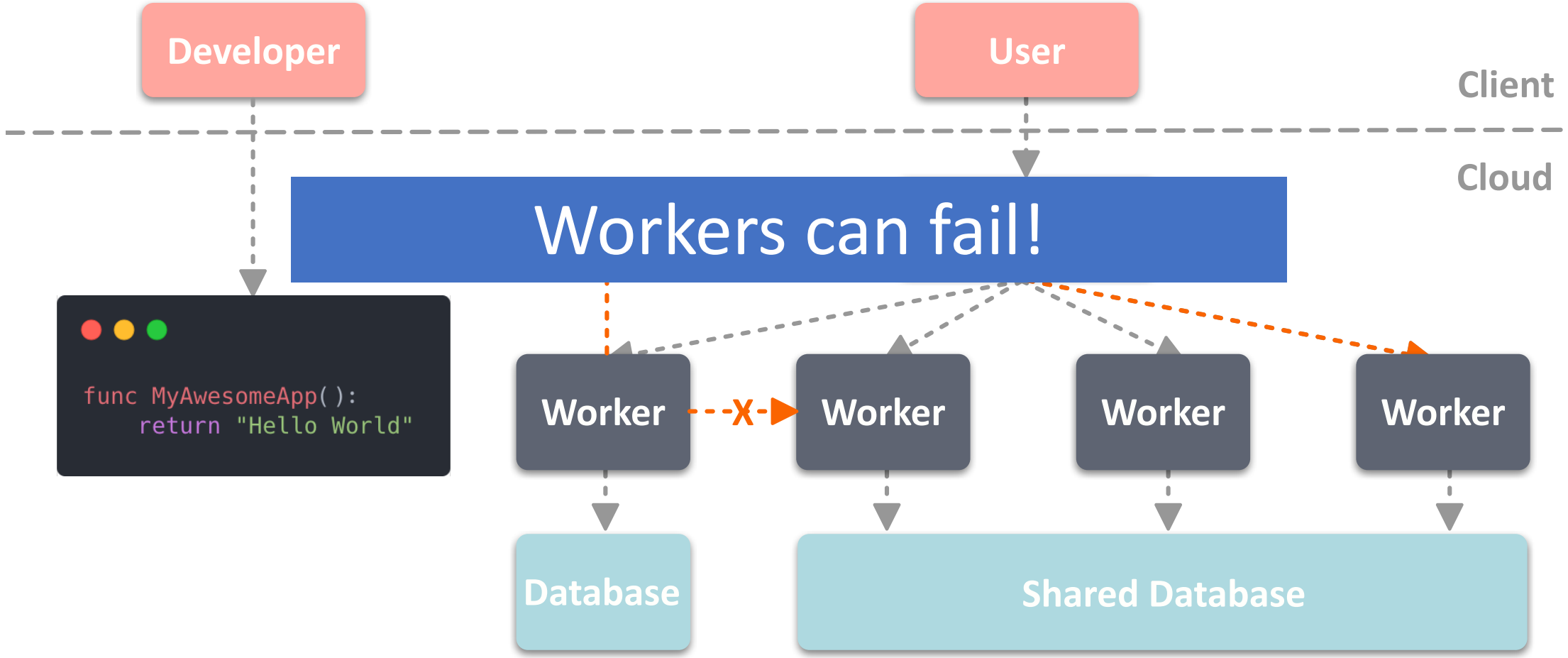
What is serverless?



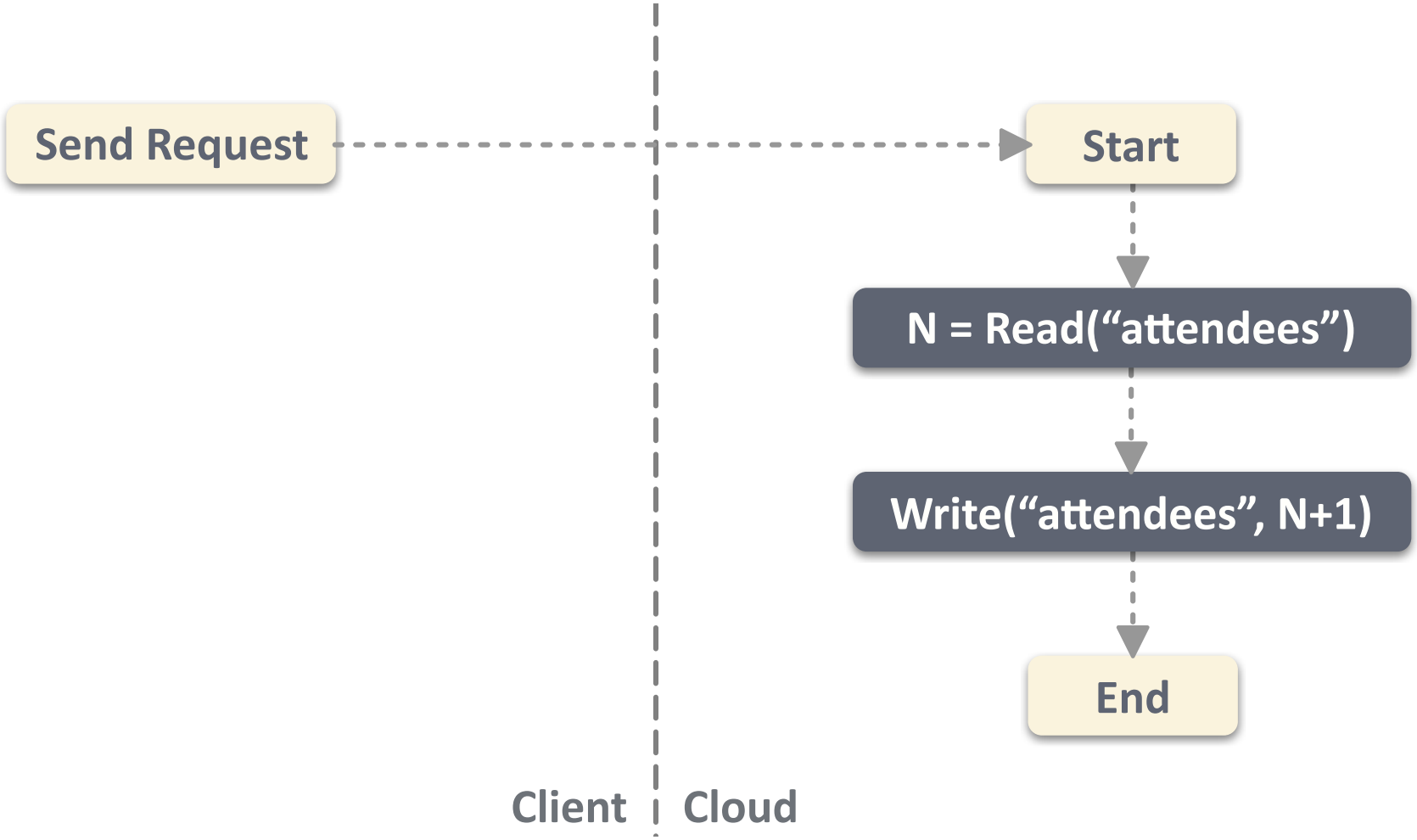
What is serverless?



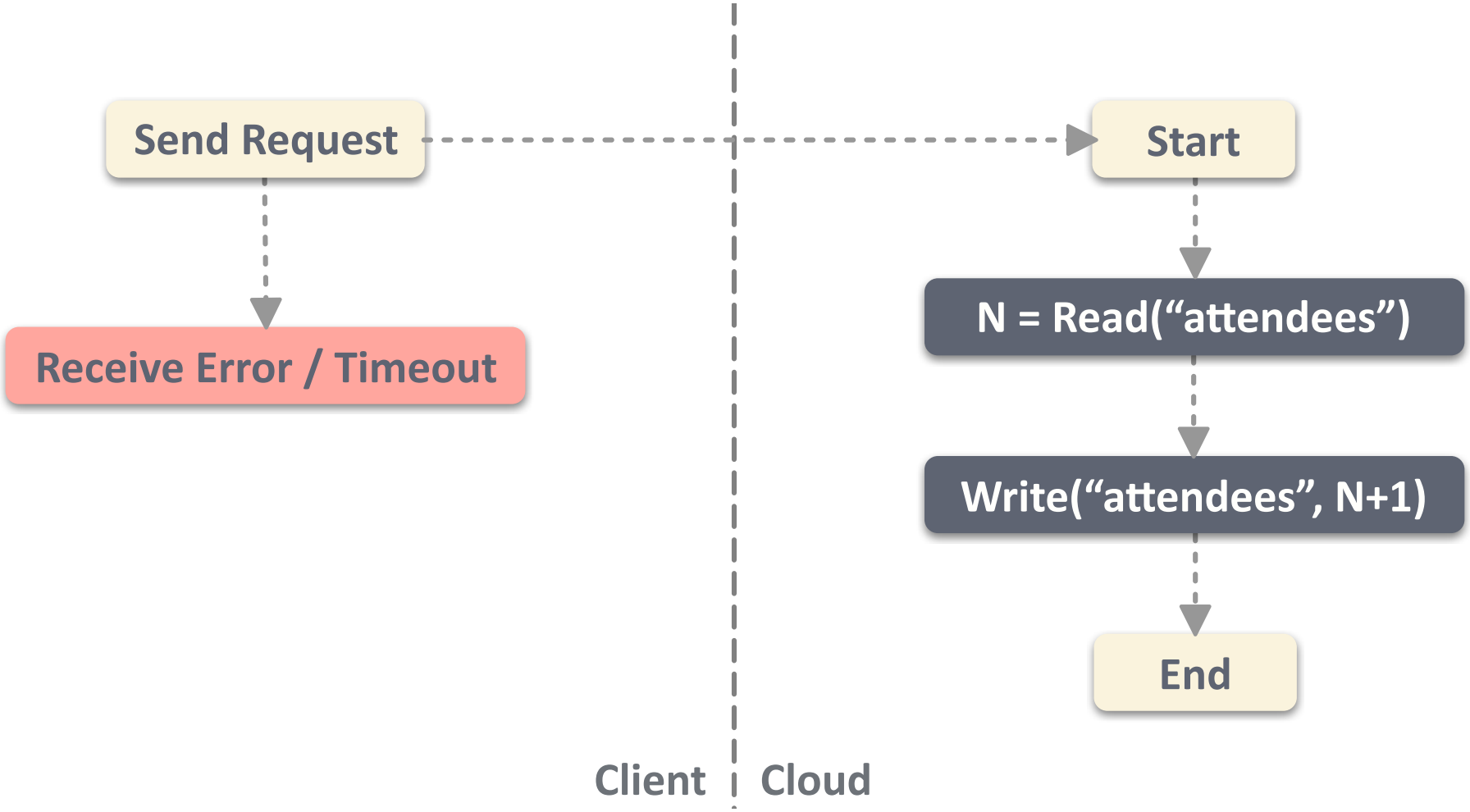
What is serverless?



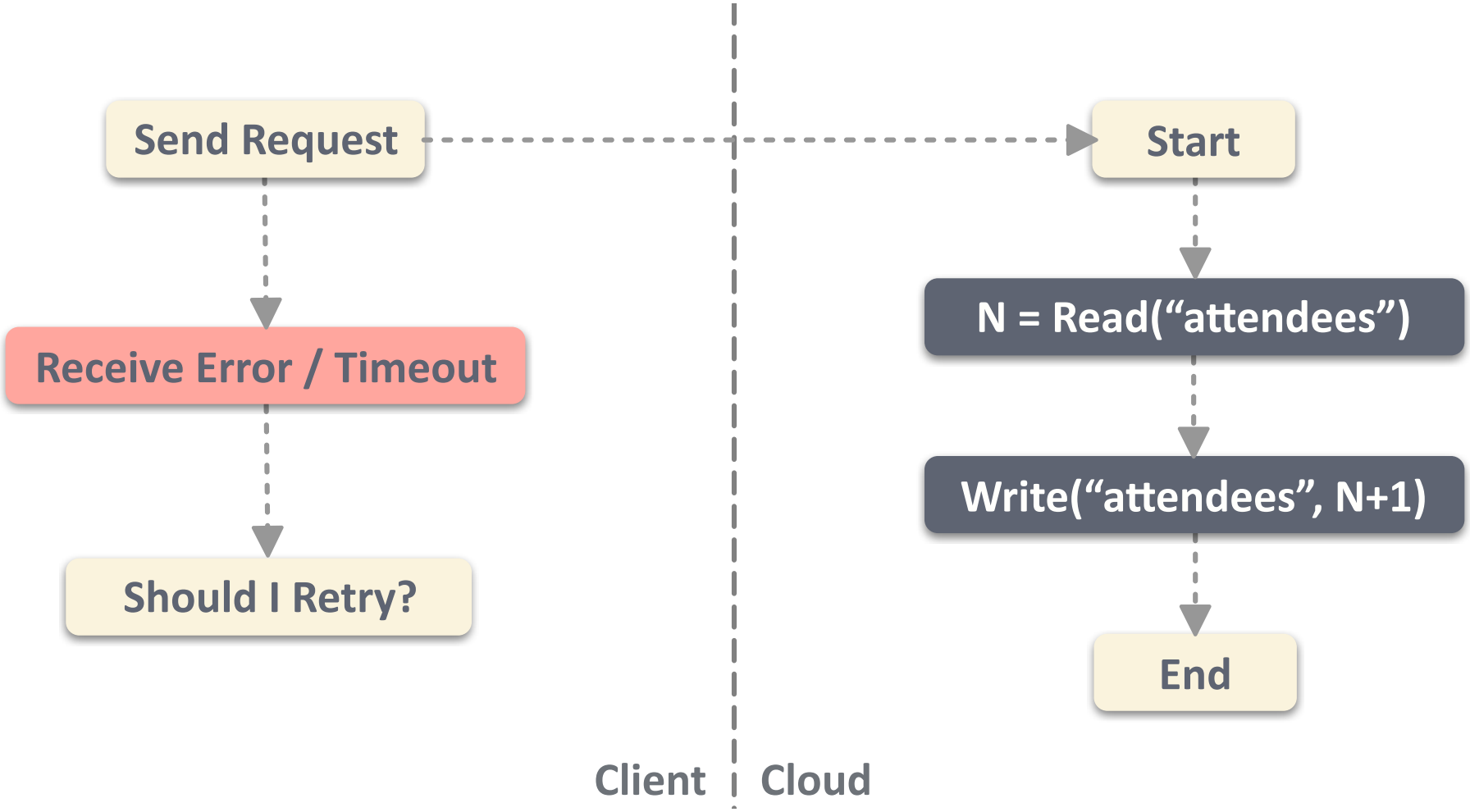
How could serverless go wrong?



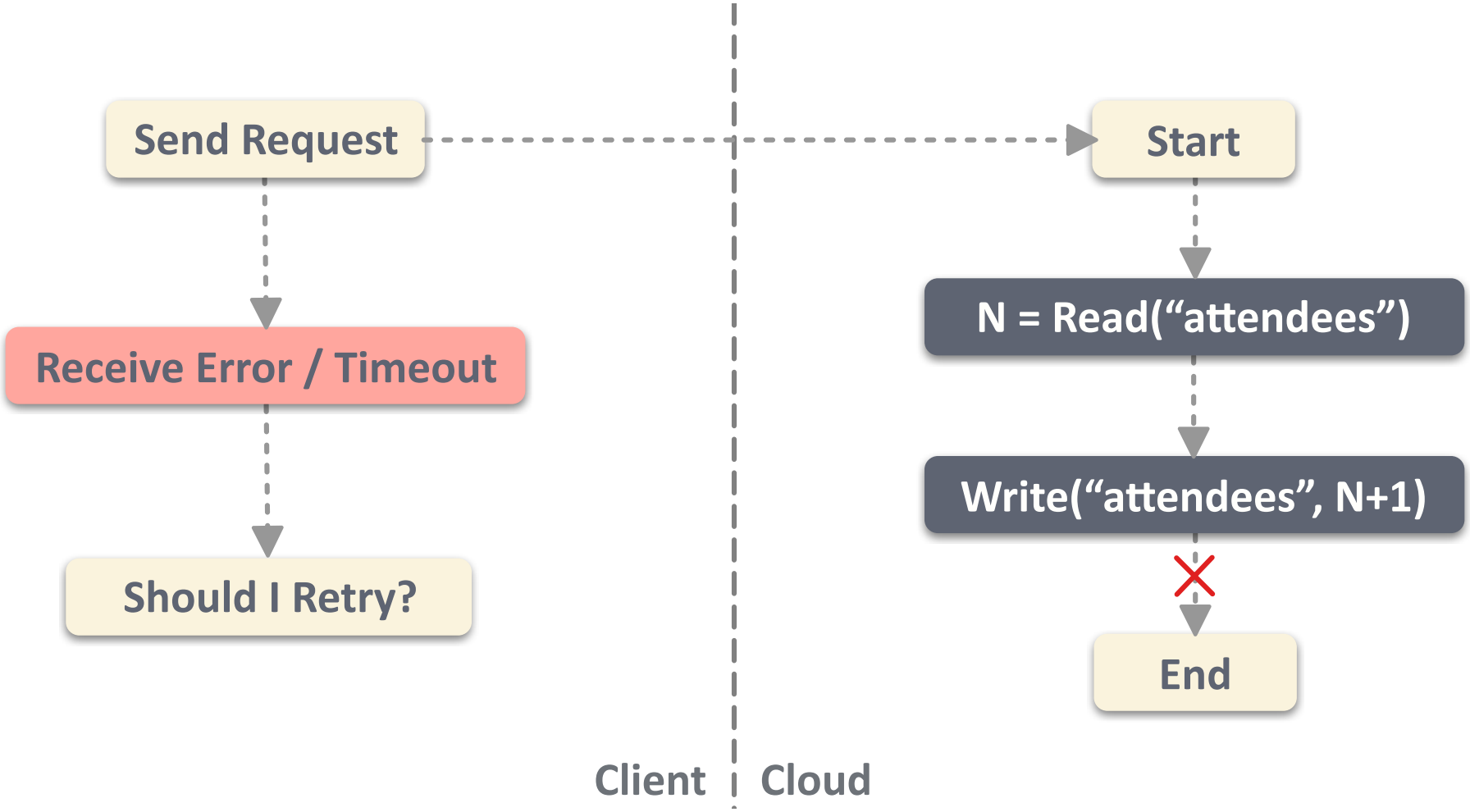
How could serverless go wrong?



How could serverless go wrong?



How could serverless go wrong?



How do I make my Lambda function idempotent to prevent inconsistencies and data loss in my application?

Last updated: 2019-05-28

I've heard that I should make my AWS Lambd

Short Description

In programming, *idempotency* is the capaci
duplicated, inconsistent, or lost data.

Making your AWS Lambda function *idempo*
more likely to reduce:

- Unnecessary API calls
- Code processing time
- Data inconsistency
- Throttles
- Latency

Resolution

To make your function idempotent, its code
application works dictates the best way to c

Example function logic

This example Lambda function logic applies

1. Extract the value of a unique attribute o
2. Check if the attribute value exists in a co
   Note: Adding AWS services to your archi
3. If a unique value exists (indicating a dup
   doesn't exist, continue the execution no
4. When the function work finishes succes
5. Finish the execution.

Best practices

Keep these best practices in mind as you de

- Plan idempotency features before devel
- Make sure that your Lambda code termi
  Lambda or by another service that invol
- Change the Lambda function timeout s
  control duplicated events might require
- Test and optimize your function as much as possible, simulating a real scenario and rate of requests. This is critical in helping to prevent the idempotency logic from causing side effects such as timeouts, excessive latency, or bottlenecks.
- Store session data using a service that can be easily escalated and that provides high throughput, such as DynamoDB.

Designing Azure Functions for identical input

09/12/2019 • 2 minutes to read •

The reality of event-
integrity and system

To illustrate, consider
moments later, some
smile back and chuck

Pressing an elevator
regardless of the nur
outcome no matter h

When it comes to bu

- What happens
- How
  perso
- Whe

There are

- Retry policies s
- Cached comma
- Application erro

To protect data integ

- Verifying of the
- Checking to see
- Reconciling log
- Concurrency co
- Duplication det
- Data freshness
- Guard logic to

Ultimately idempotency is achieved by ensuring

Cloud Functions › Documentation

Tips & Tricks

Send feedback

This document desc

Note: Several of
the execution en
time to complete
possible (for exa

Write idempote

Your functions shou
the previous invocat

Ensure HTTP fu

If your function is H
your function execu
cause unpredictable
latency.

Make retryable background functions idempotent

Background functions that can be retried must be idempotent. Here are some general guidelines for making a background function idempotent:

ter. If you are using such an

. So a general best practice for

- Make sure that your code is internally idempotent. For example:
  - Make sure that mutations can happen more than once without changing the outcome.
  - Query database state in a transaction before mutating the state.
  - Make sure that all side effects are themselves idempotent.
- Impose a transactional check outside the function, independent of the code. For example, persist state somewhere recording that a given event ID has already been processed.
- Deal with duplicate function calls out-of-band. For example, have a separate clean up process that cleans up after duplicate function calls.

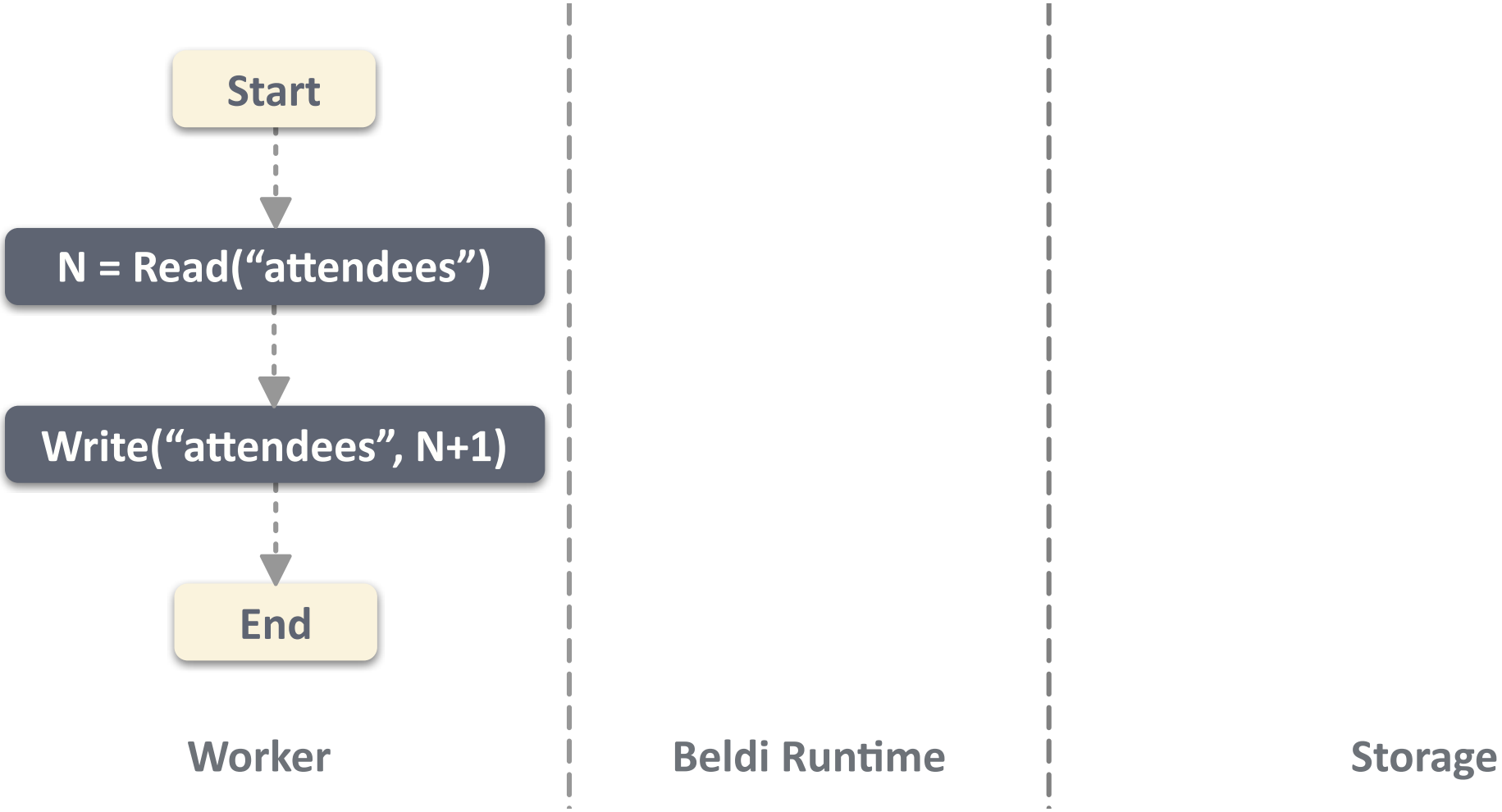
Write Idempotent Functions!

Beldi makes stateful serverless functions idempotent automatically!

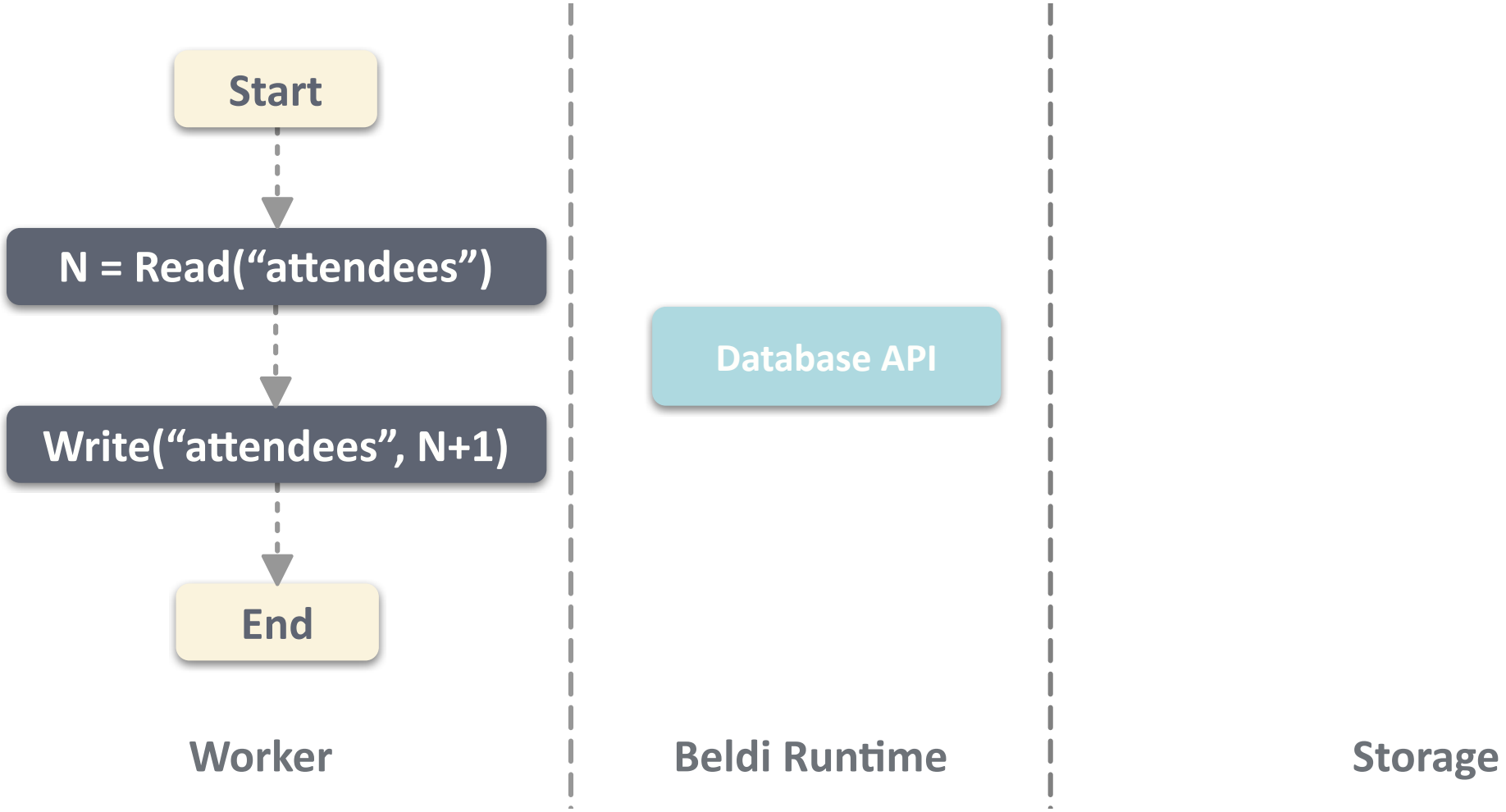
Outline

- **Beldi’s Infrastructure**
- Linked DAAL
- Invocation with exactly-once semantics
- Evaluation
- Conclusion

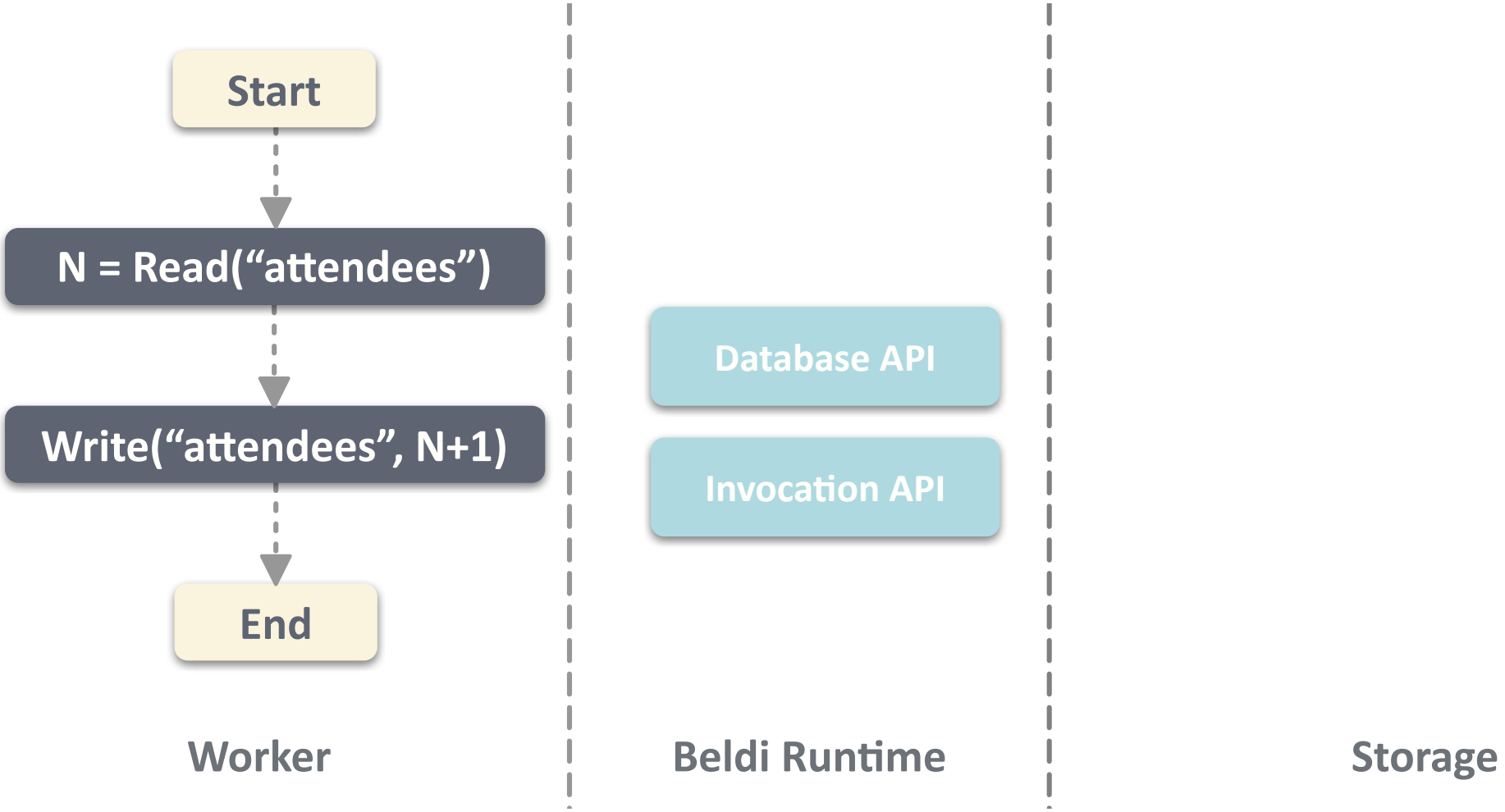
Beldi's architecture



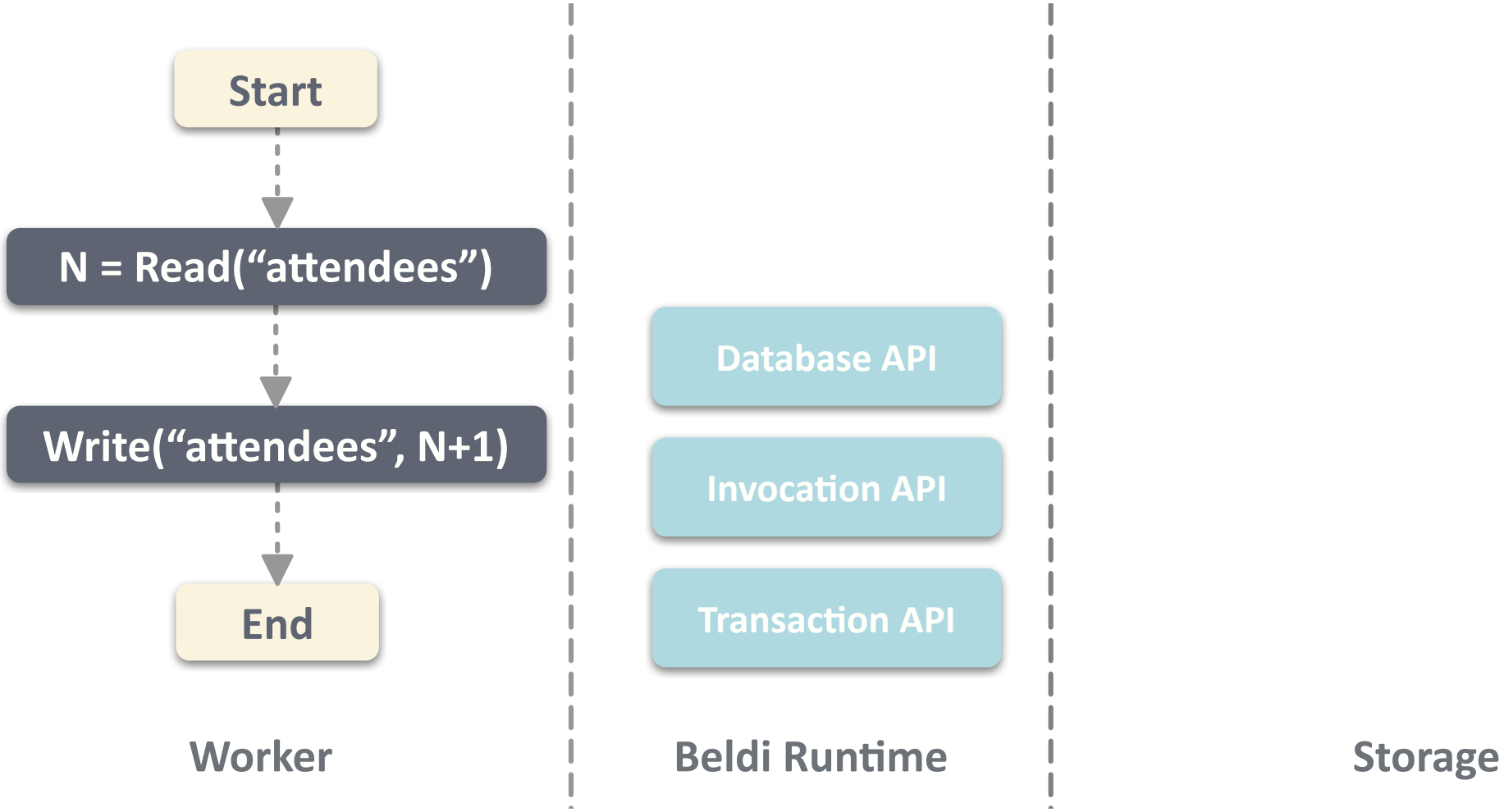
Beldi's architecture



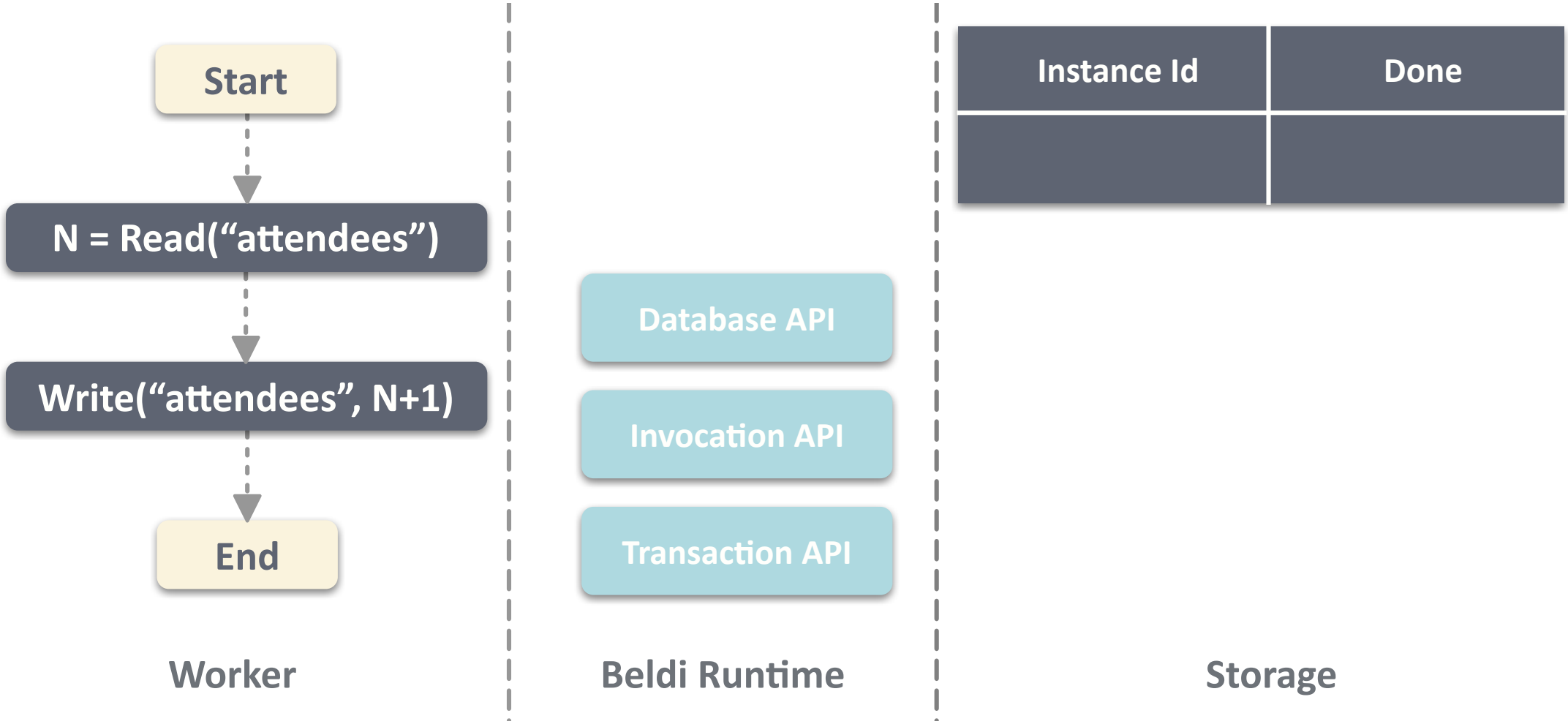
Beldi's architecture



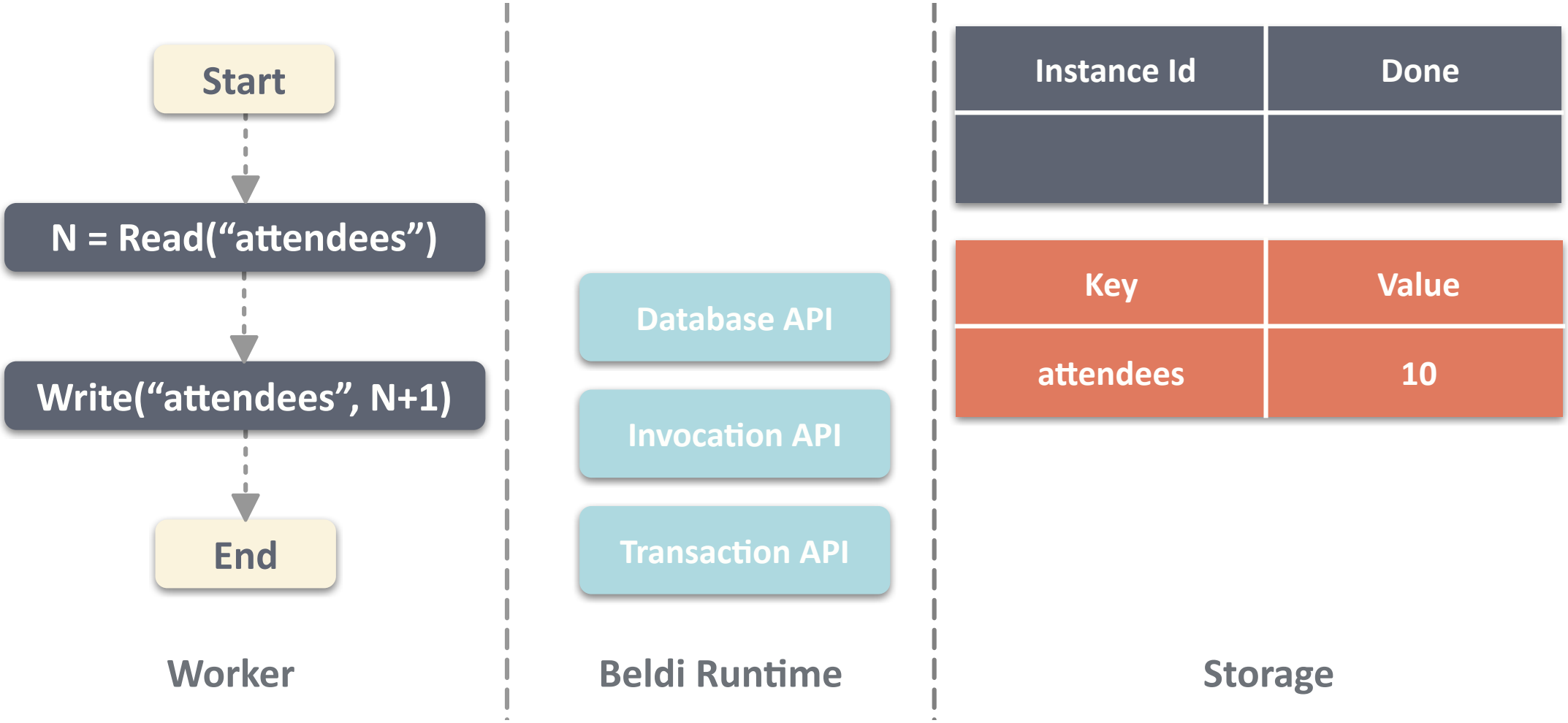
Beldi's architecture



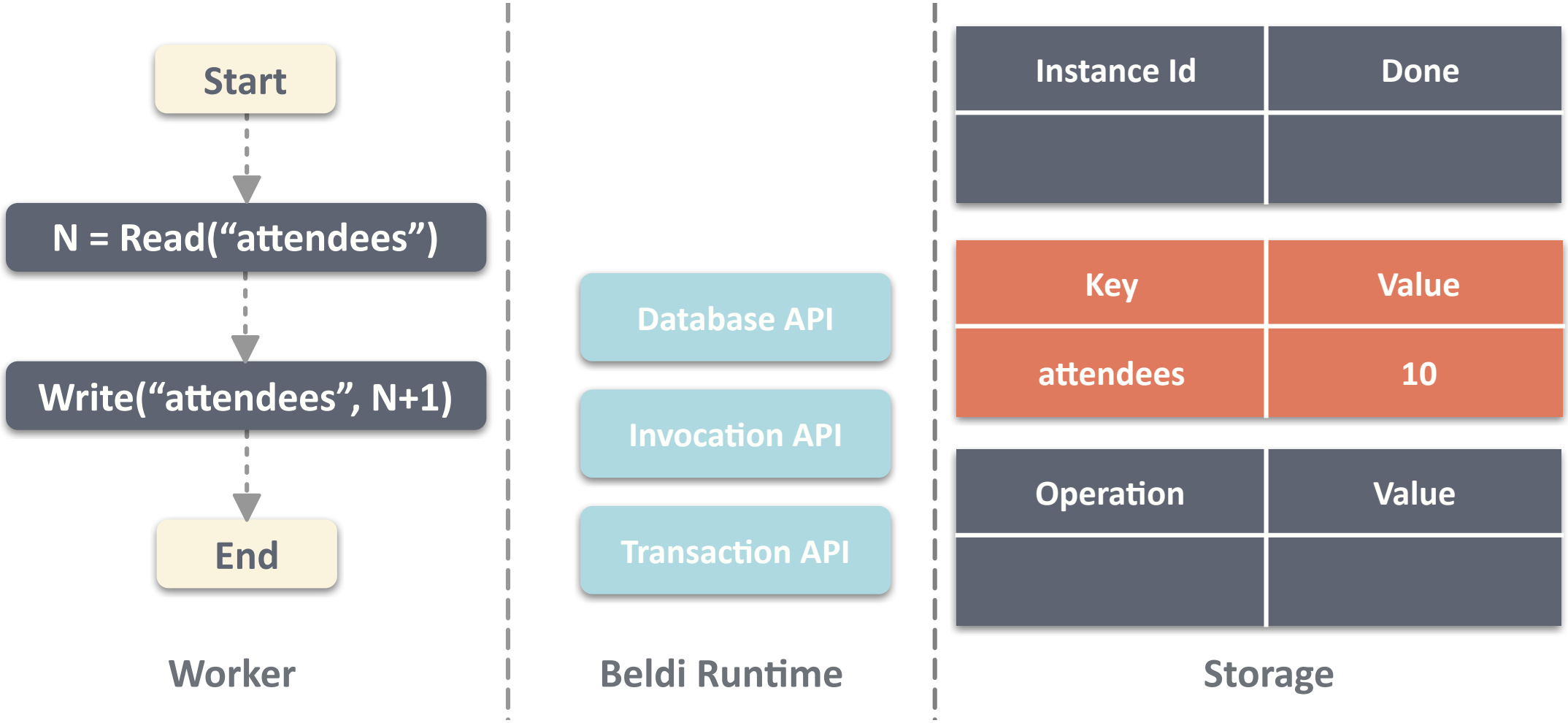
Beldi's architecture



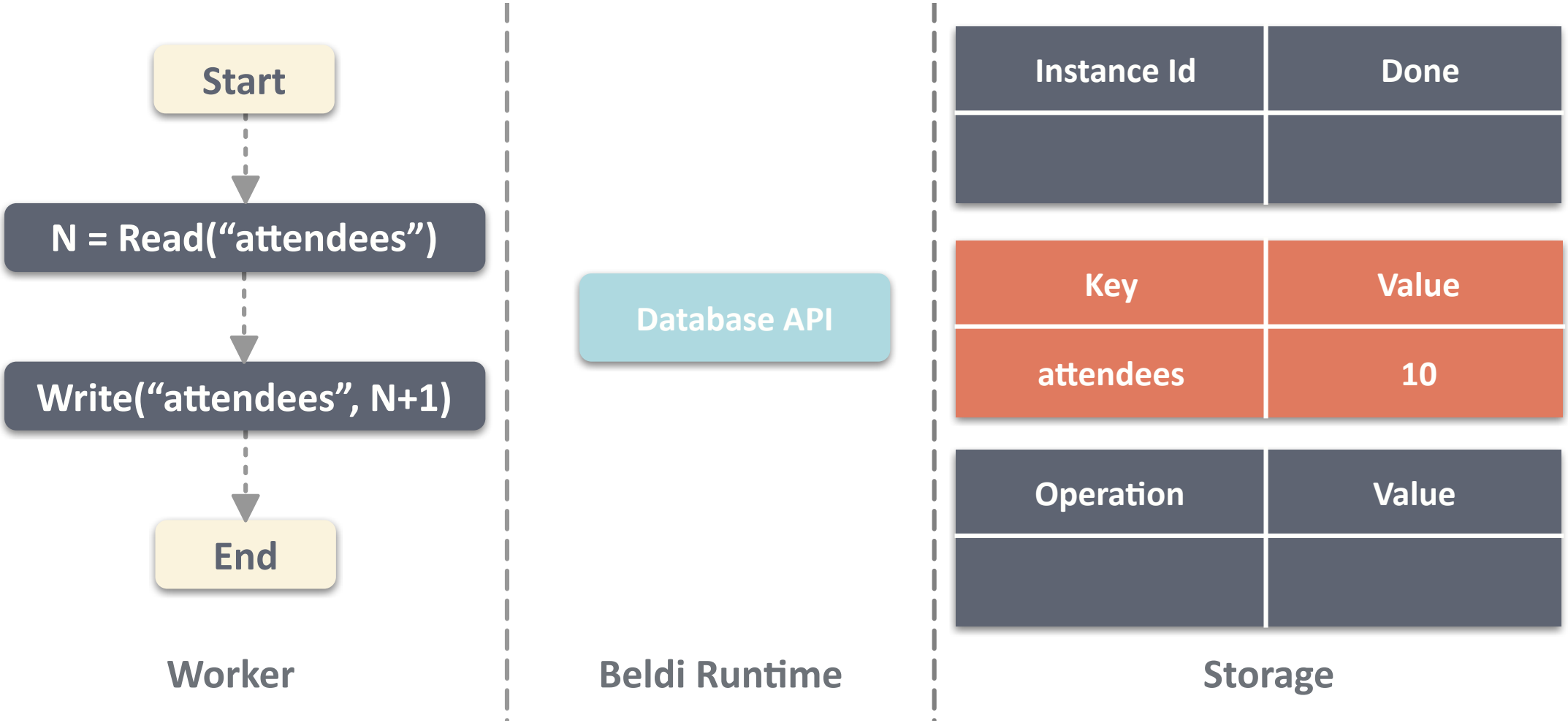
Beldi's architecture



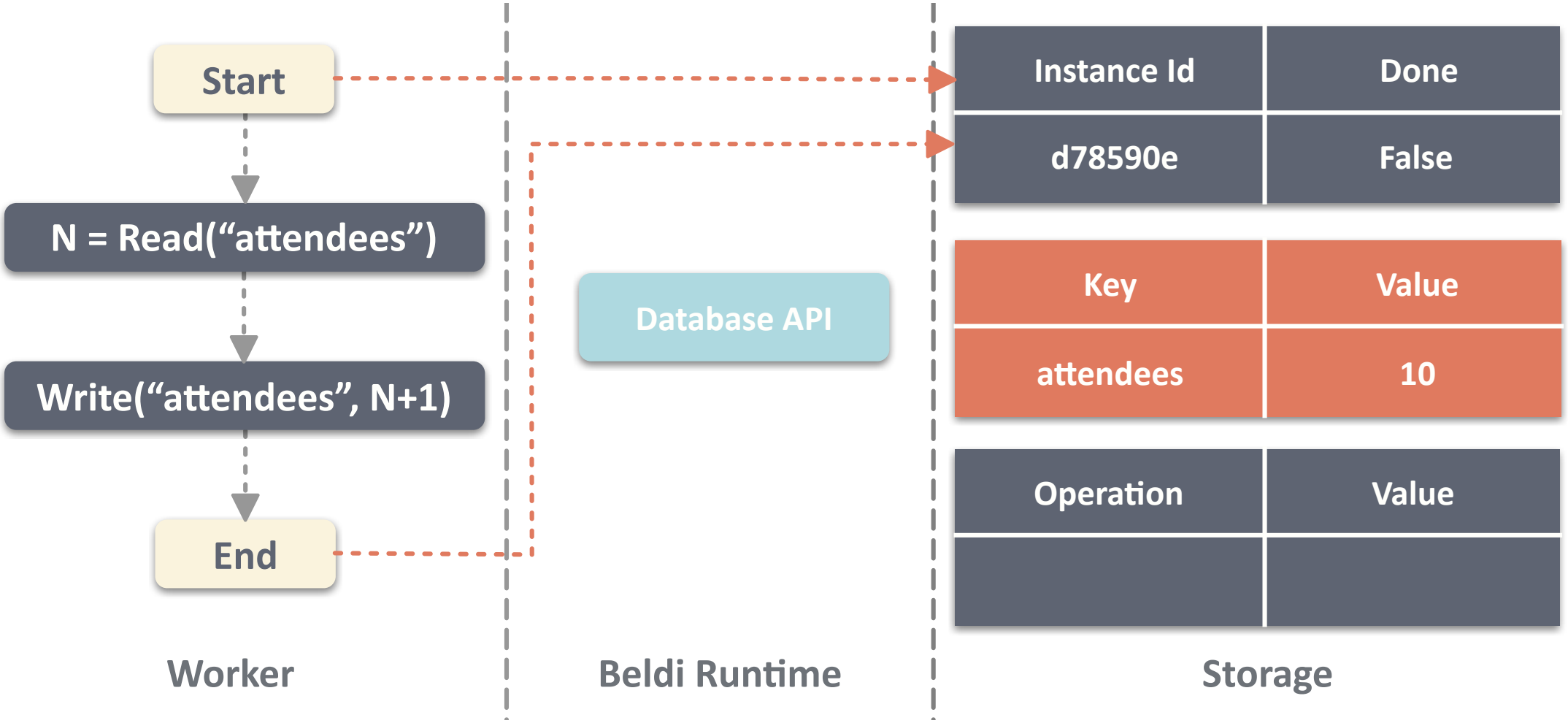
Beldi's architecture



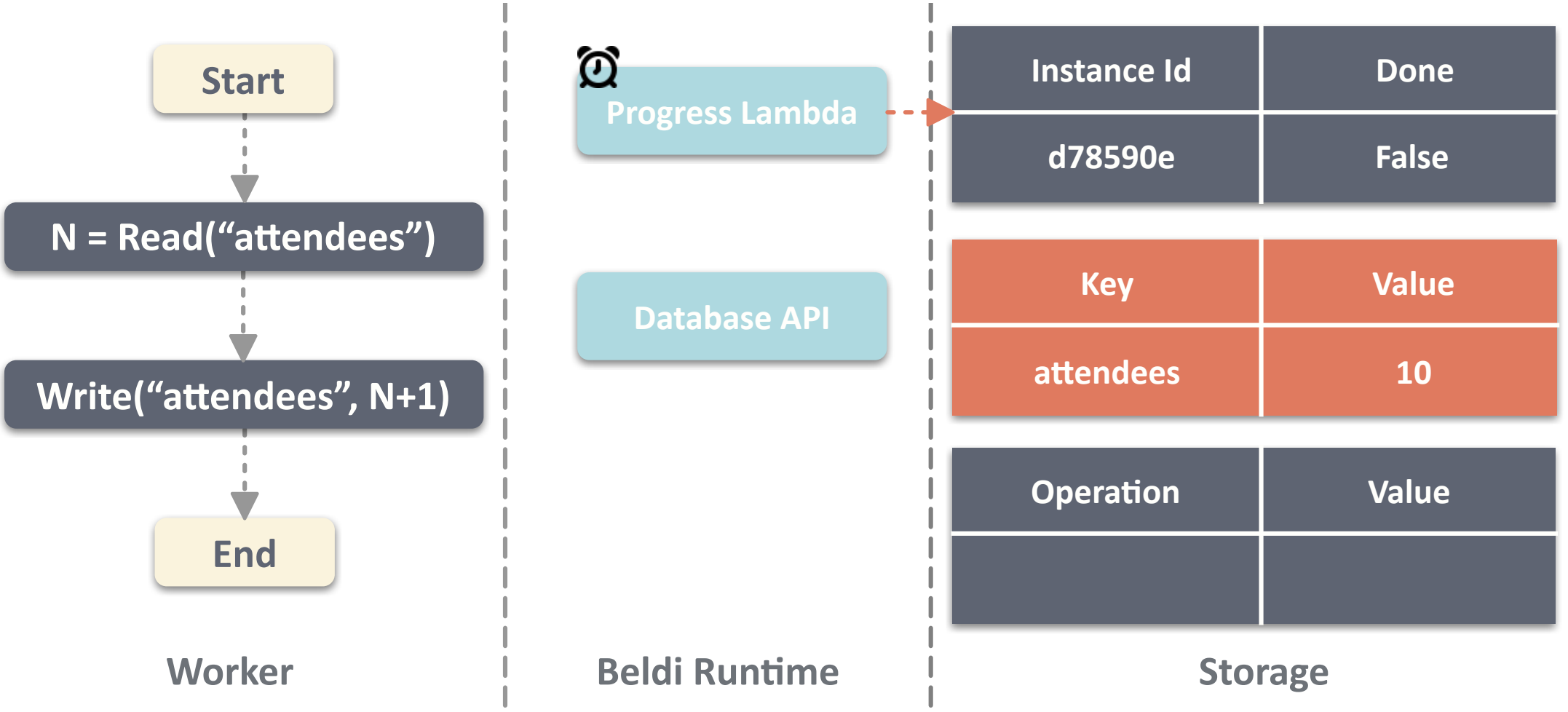
Beldi's architecture



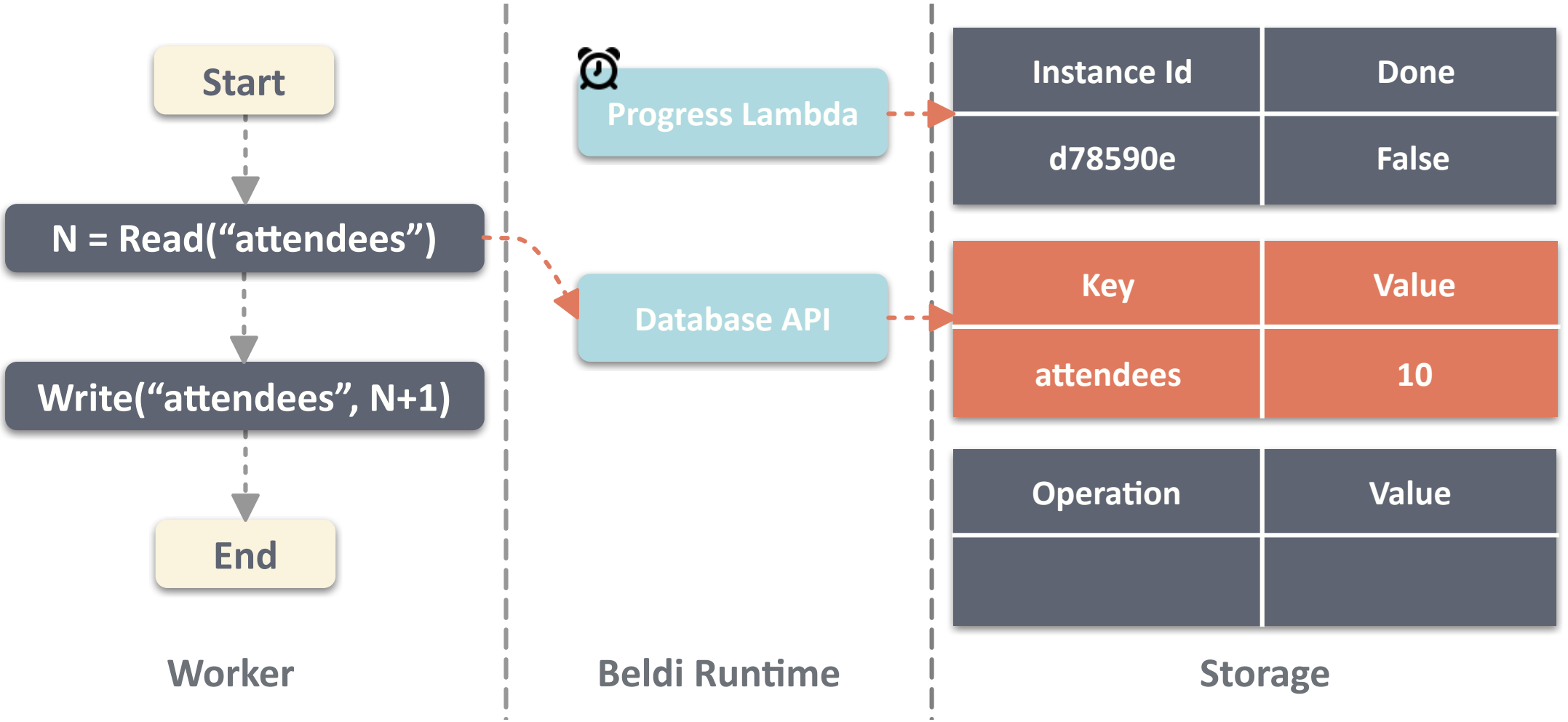
Beldi's architecture



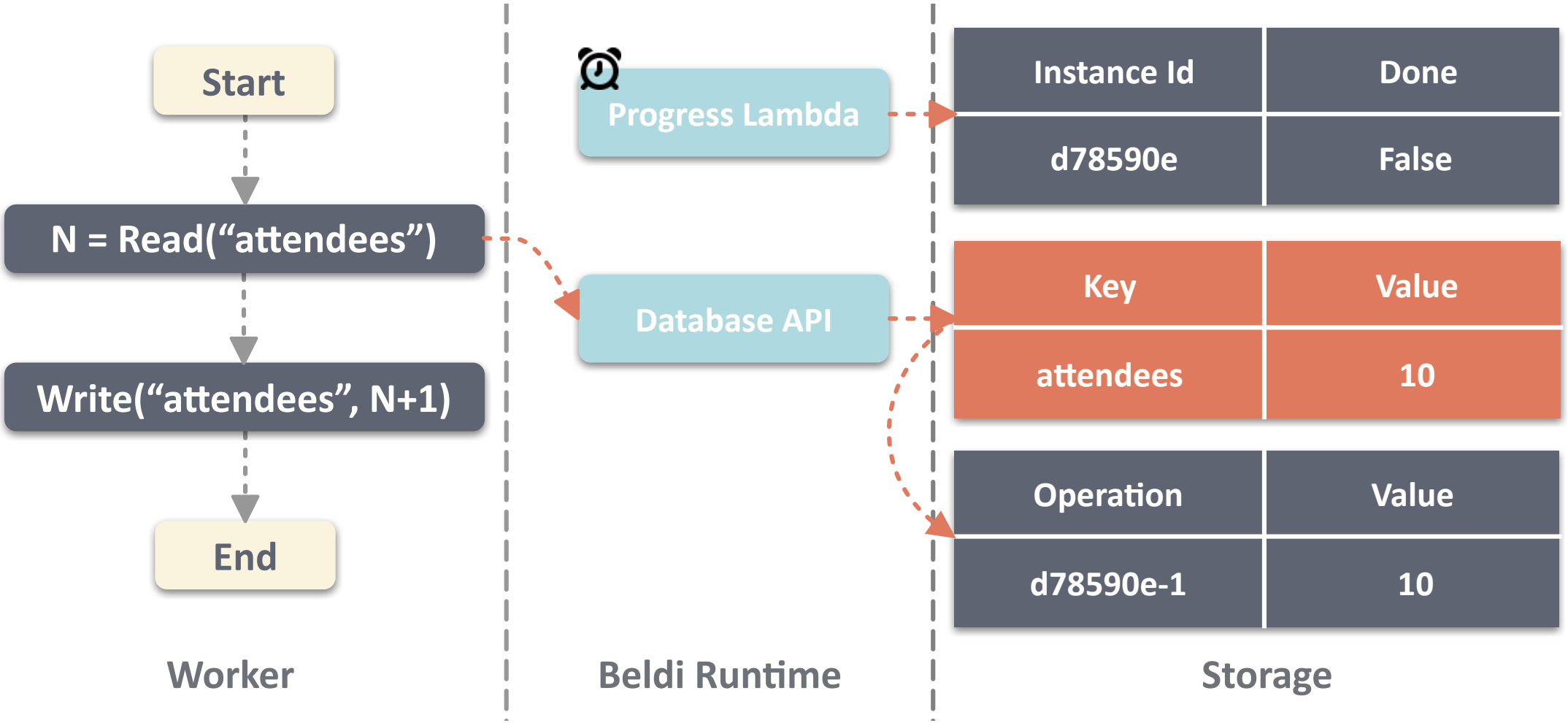
Beldi's architecture



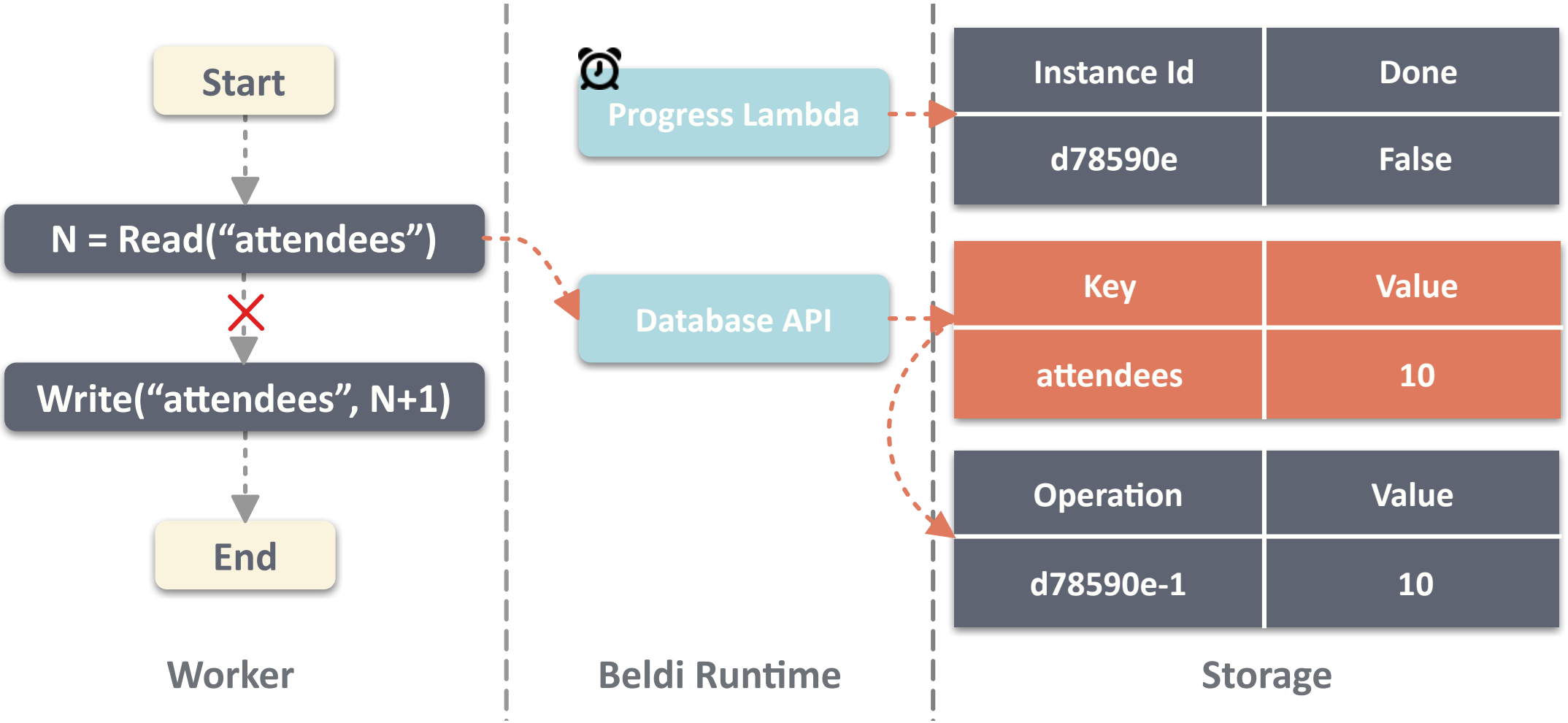
Beldi's architecture



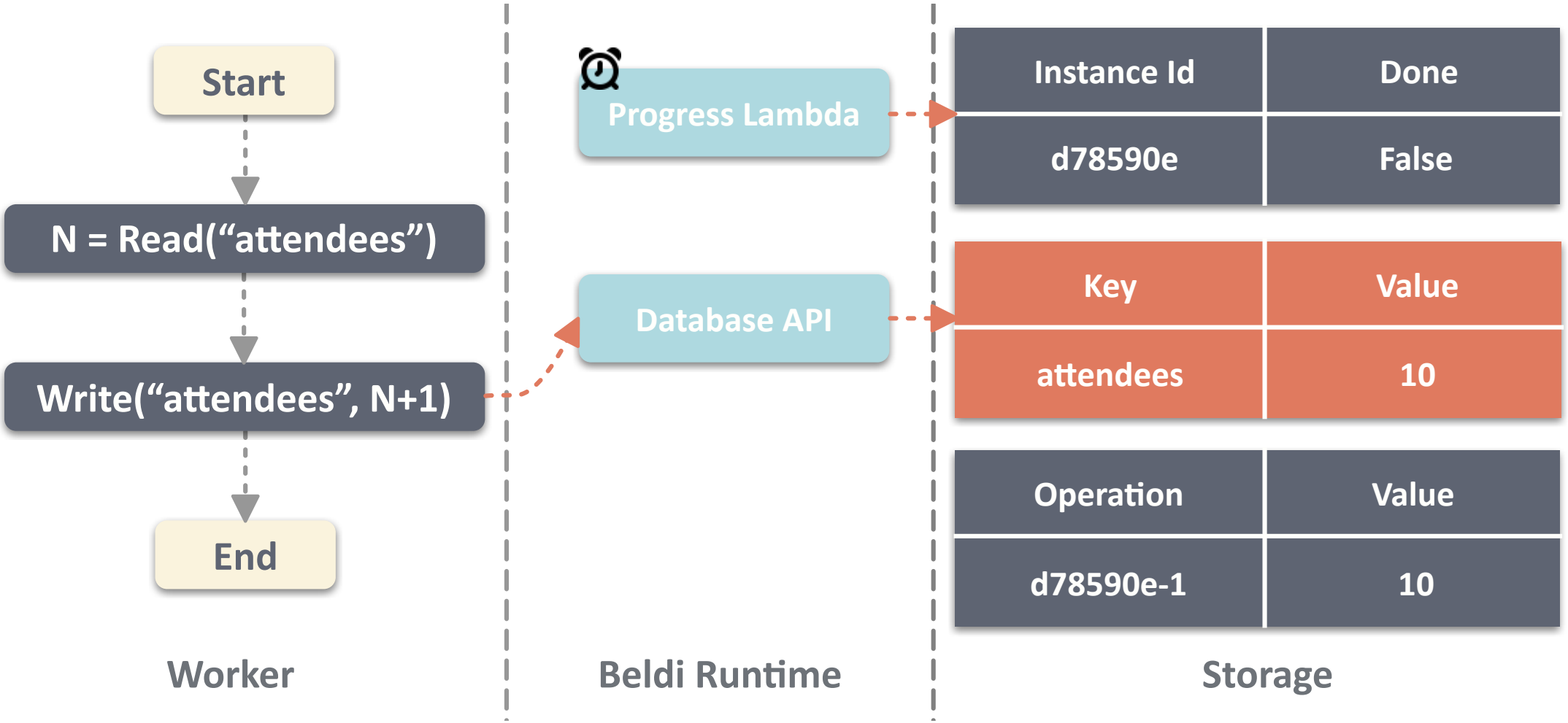
Beldi's architecture



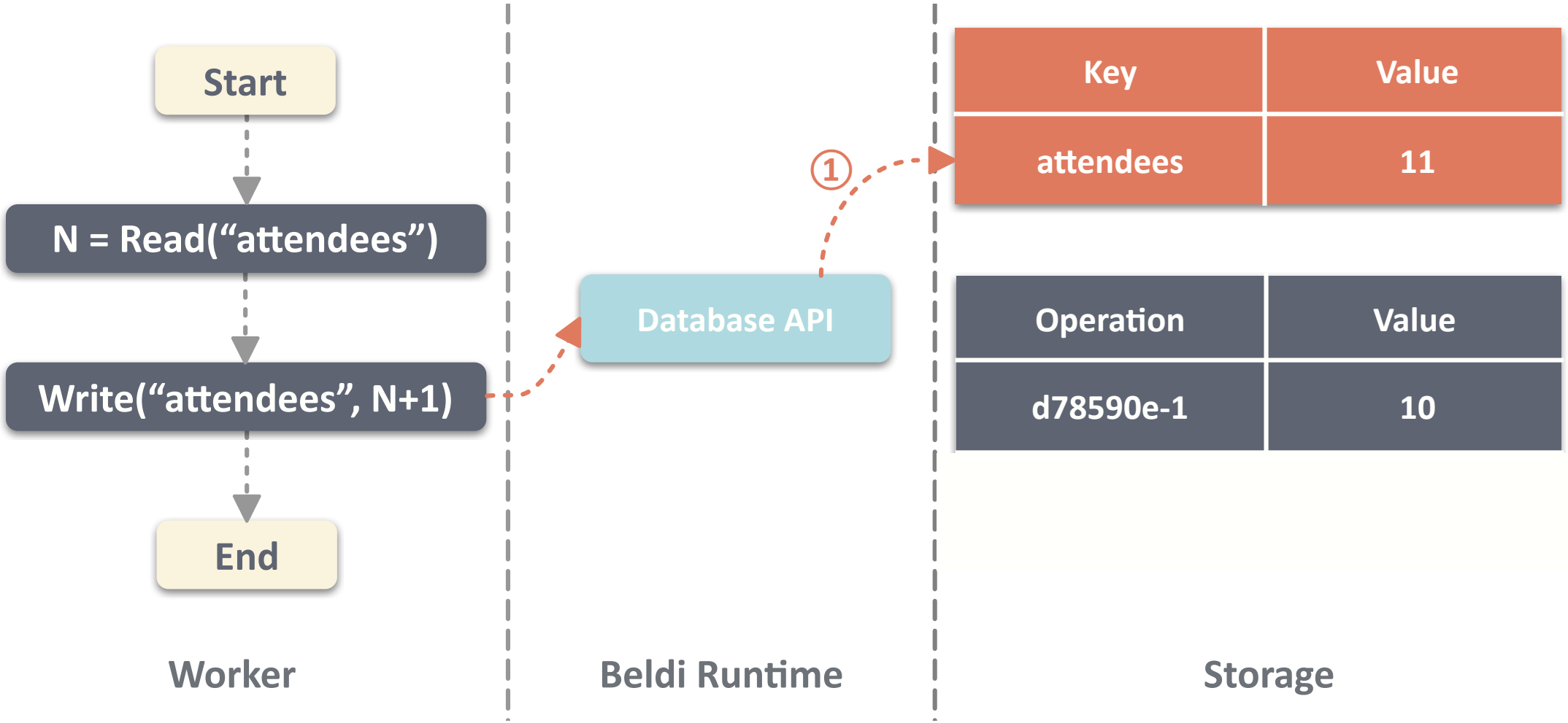
Beldi's architecture



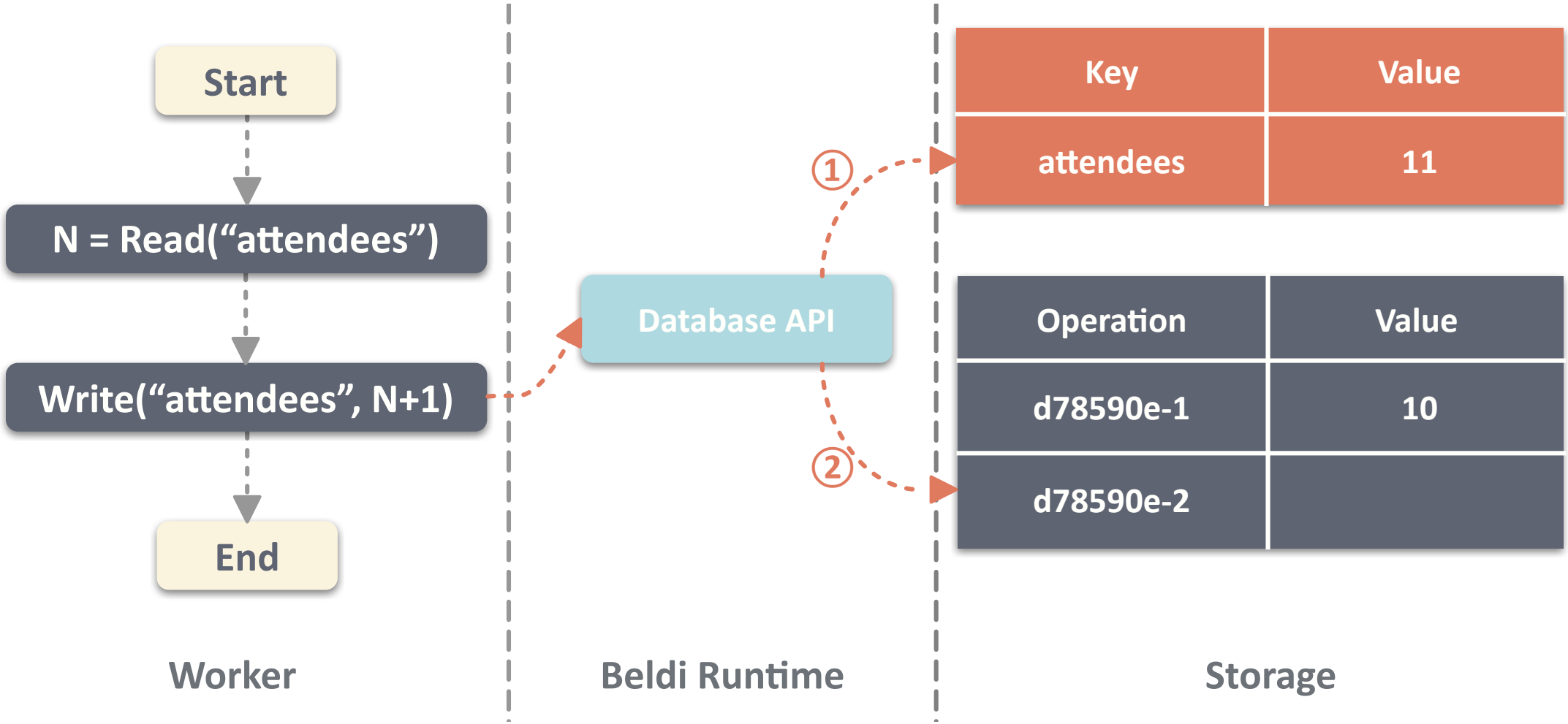
Beldi's architecture



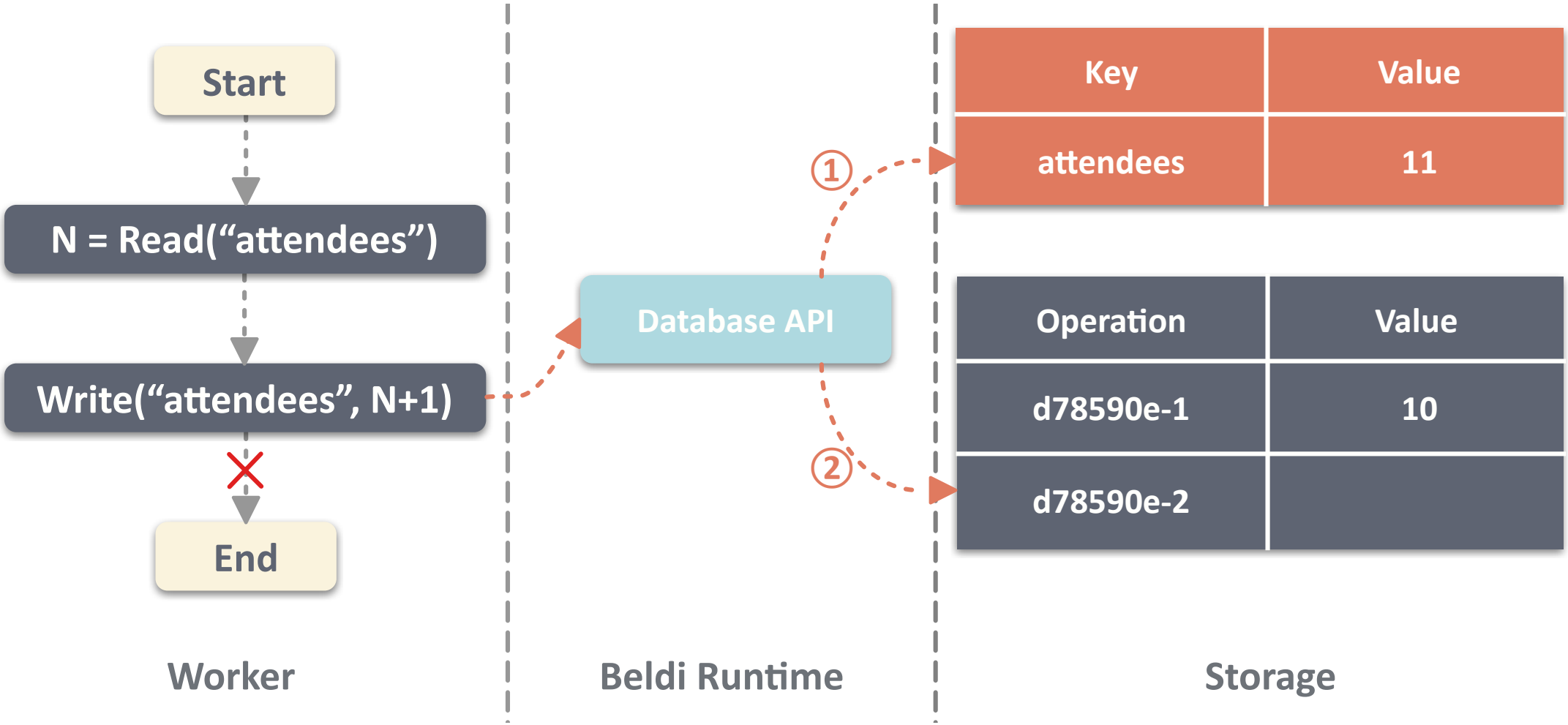
Beldi's architecture



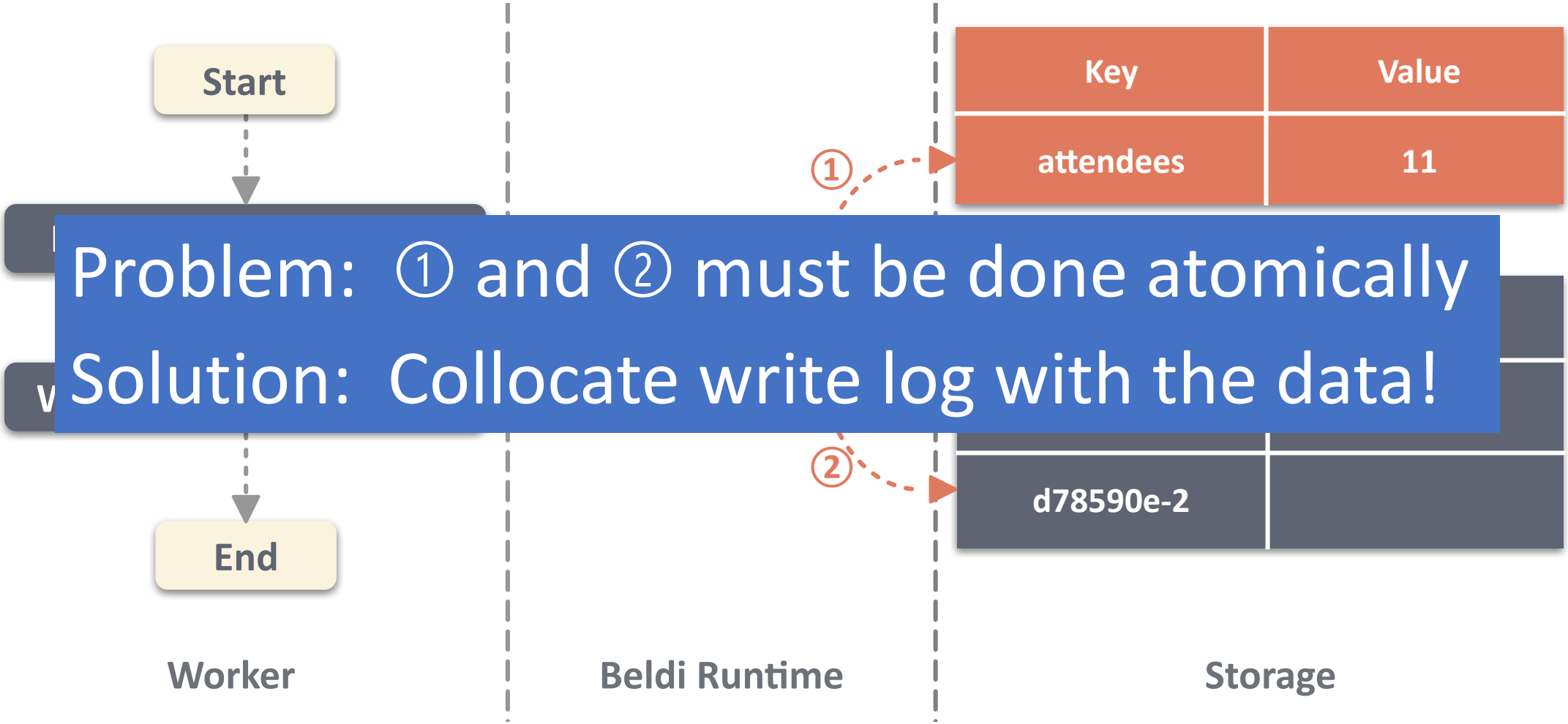
Beldi's architecture



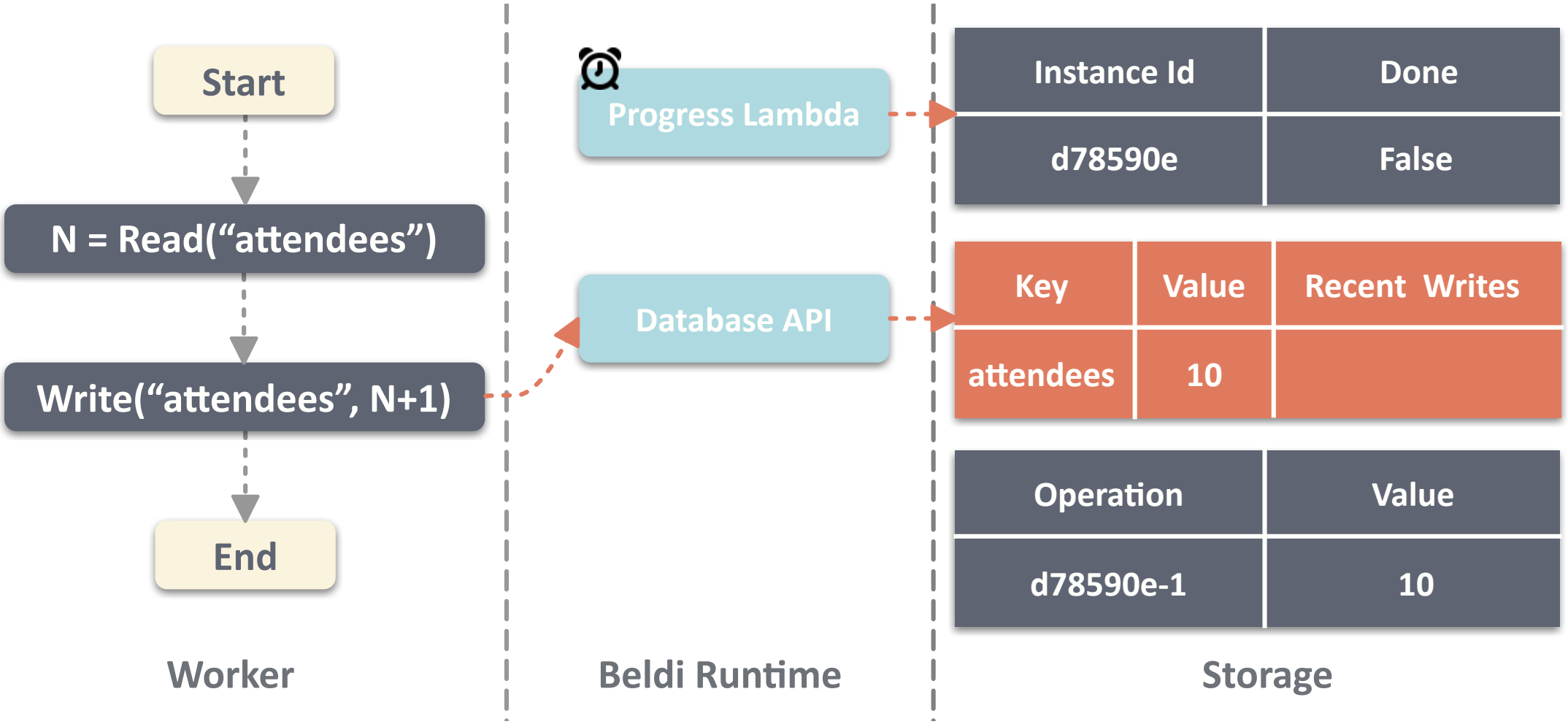
Beldi's architecture



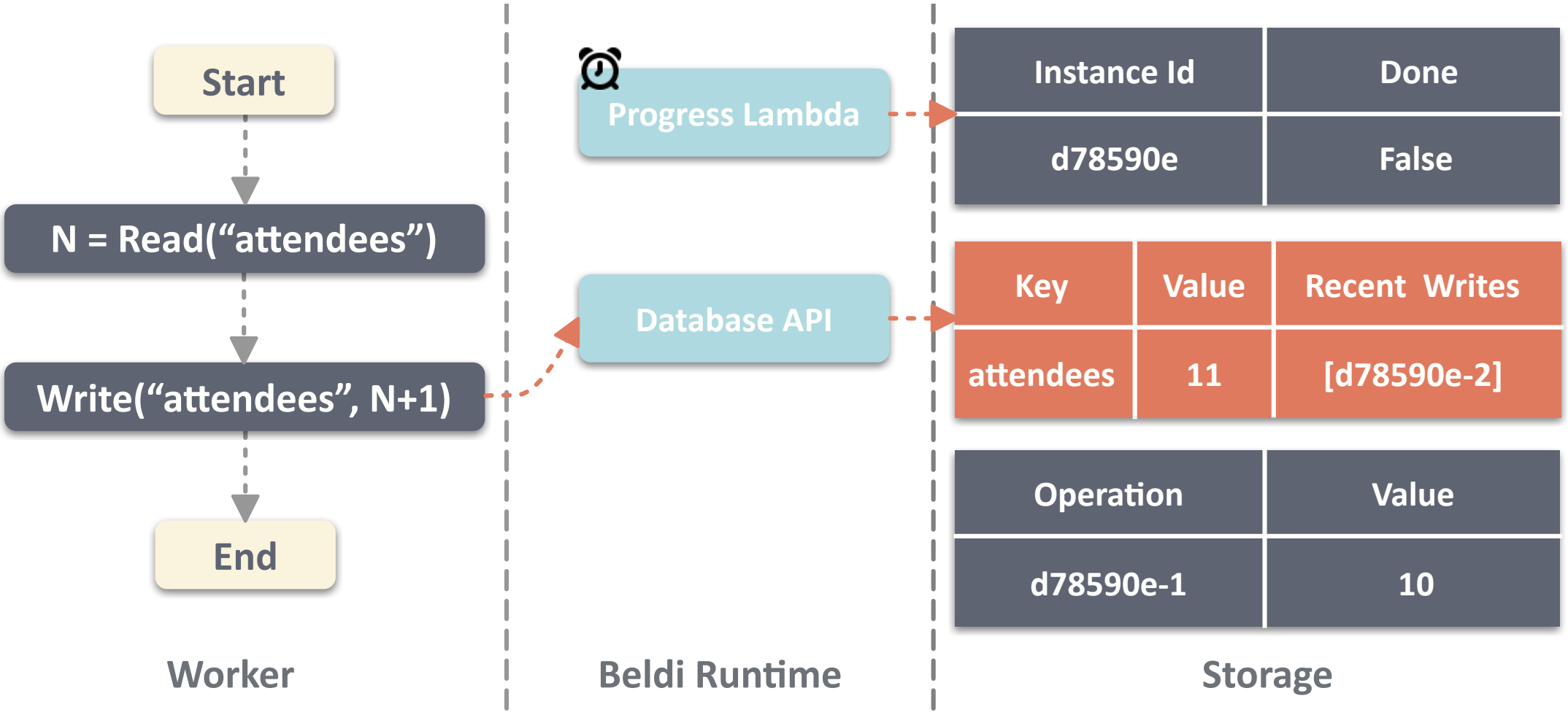
Beldi's architecture



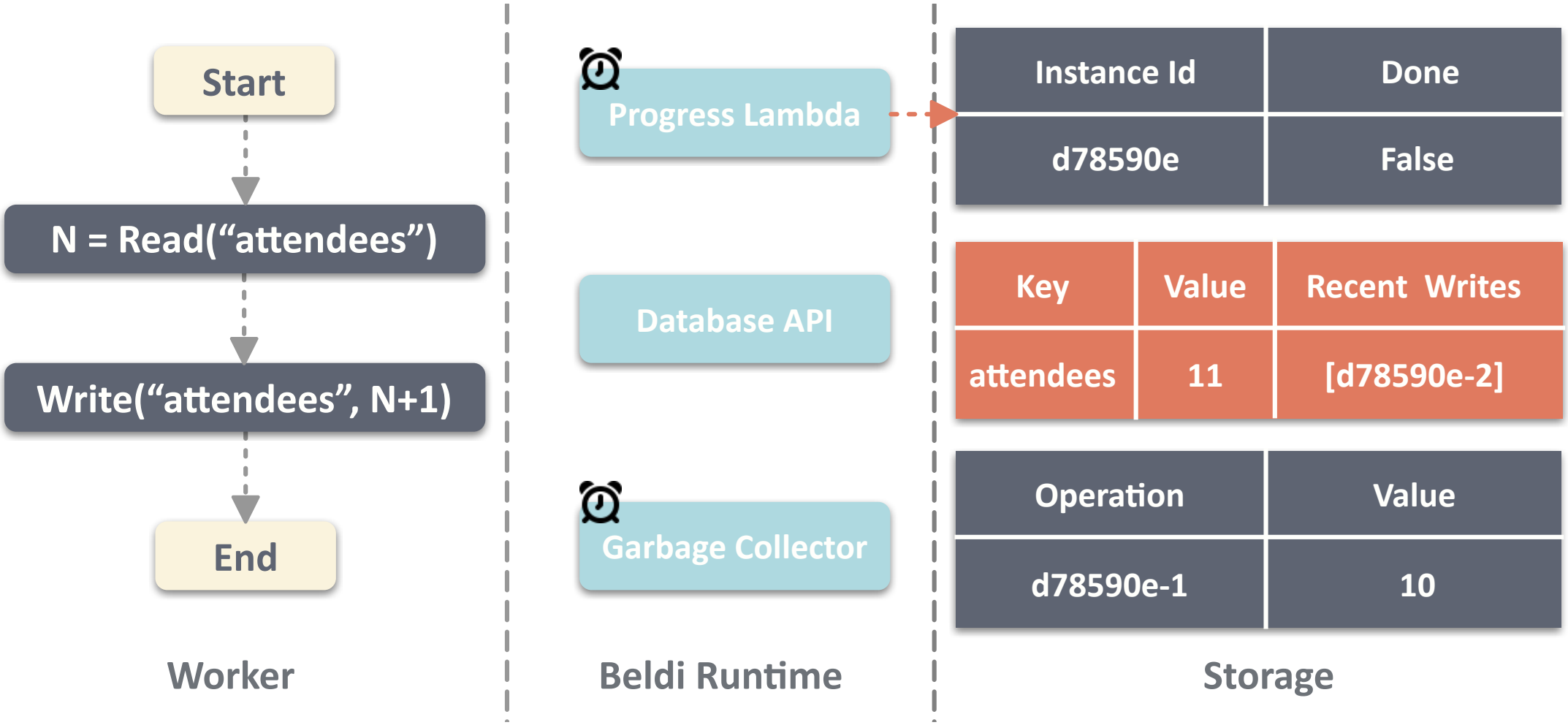
Beldi's architecture



Beldi's architecture



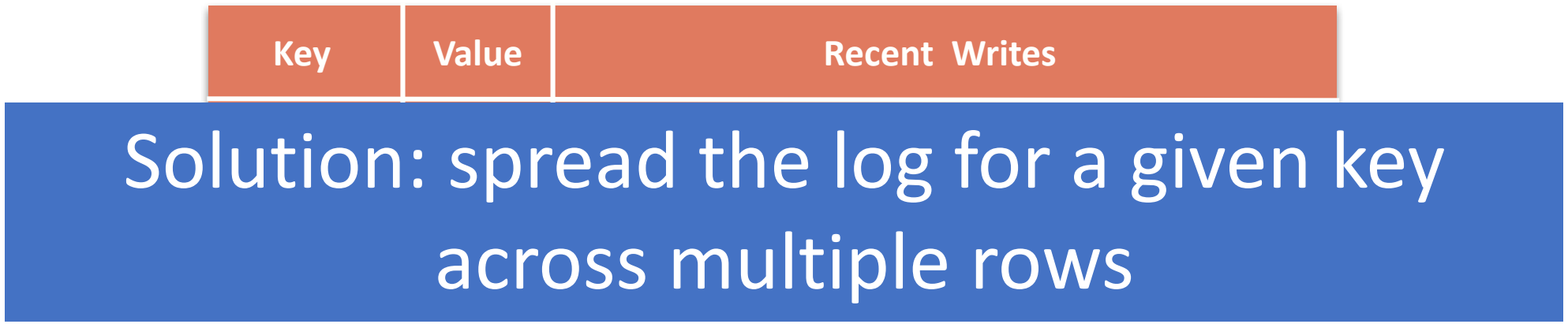
Beldi's architecture



Technical Challenges

1. Limitation of databases
2. Federated setup
3. Transactions across multiple lambdas

Limitation of databases

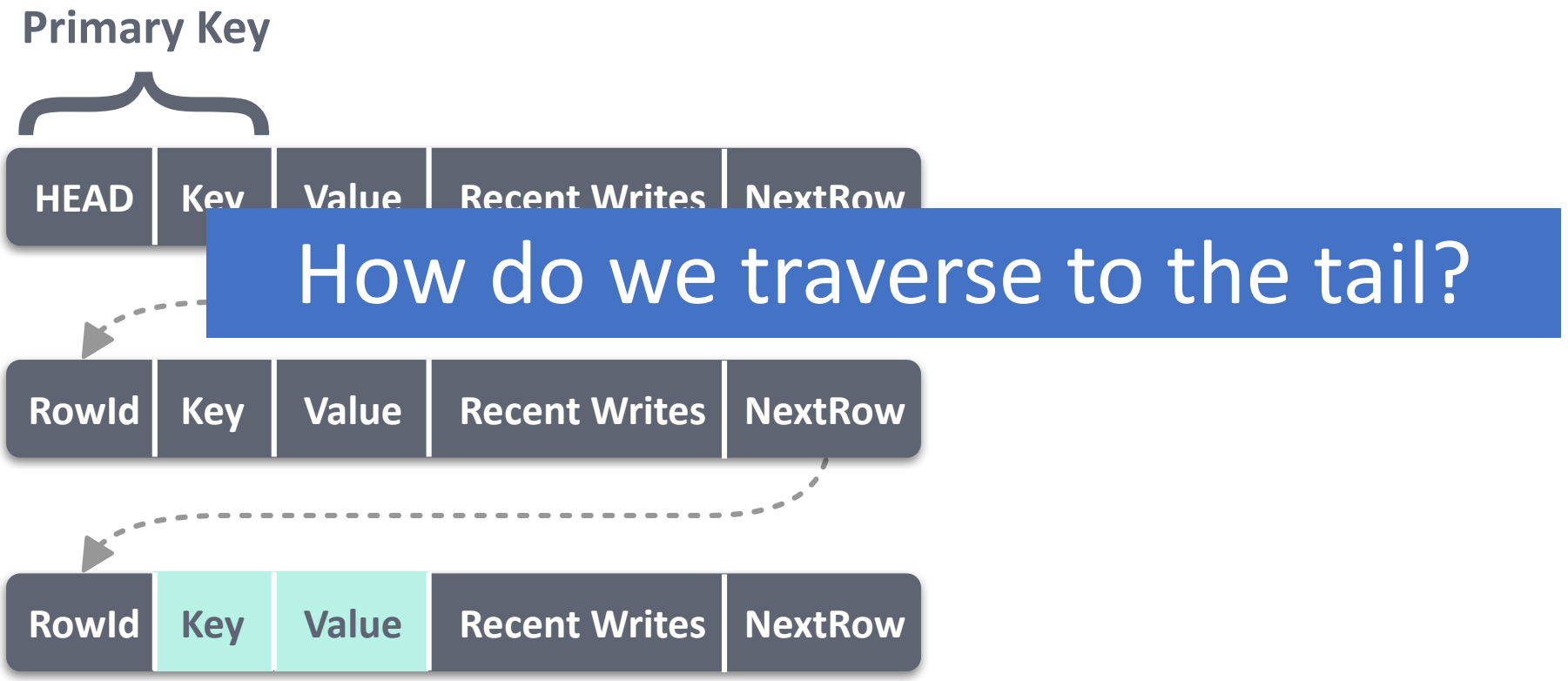


Limitation of databases

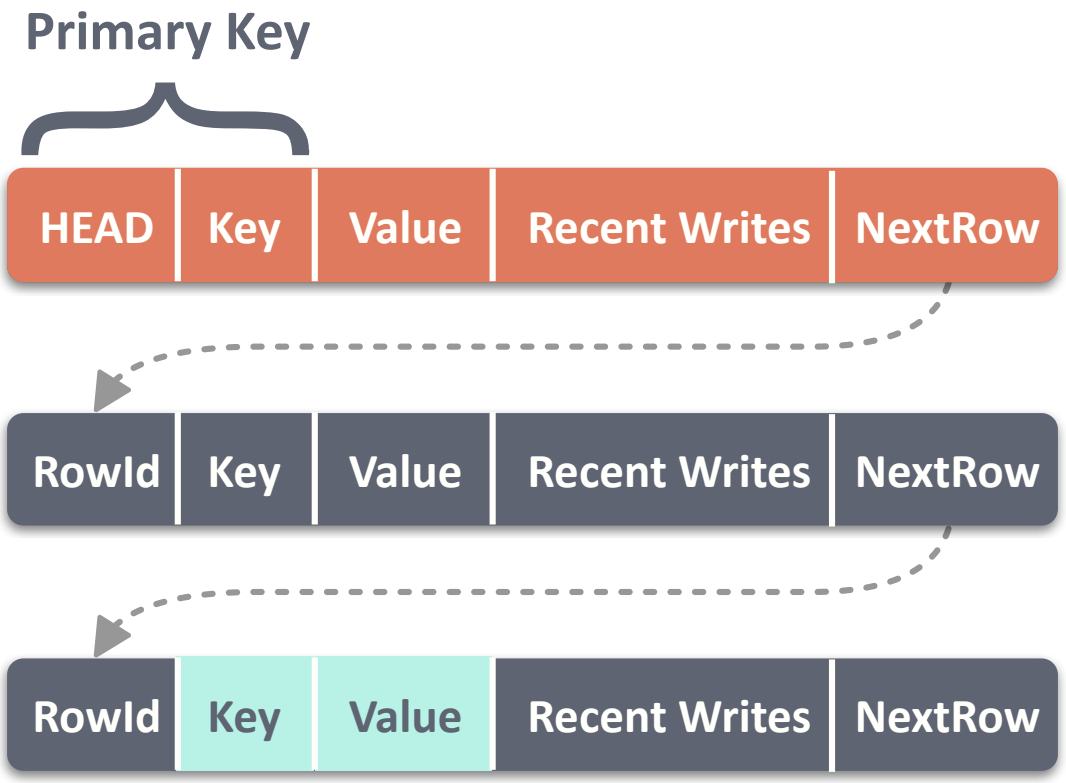
RowId	Key	Value	Recent Writes	NextRow
HEAD	attendees	10	[d78590e-1, d78590e-2, …, d78590e-1000]	f9cec2e

RowId	Key	Value	Recent Writes	NextRow
f9cec2e	attendees	11	[d78590e-1001]	

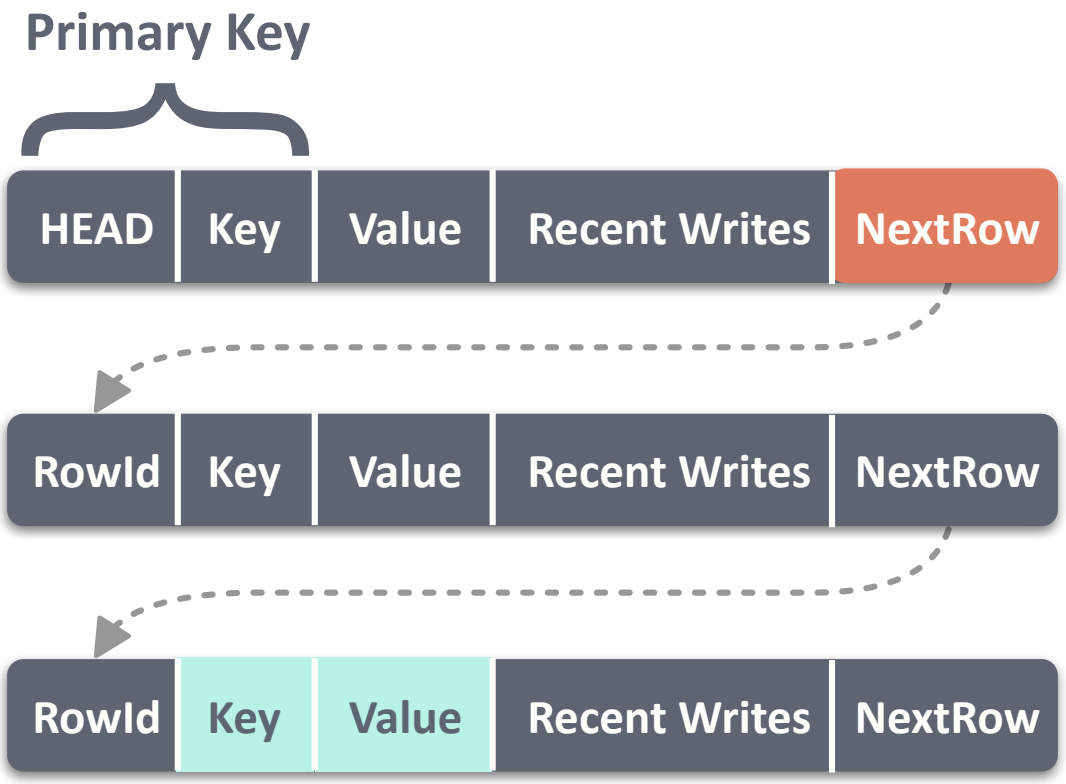
Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
How do we traverse to the tail?
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

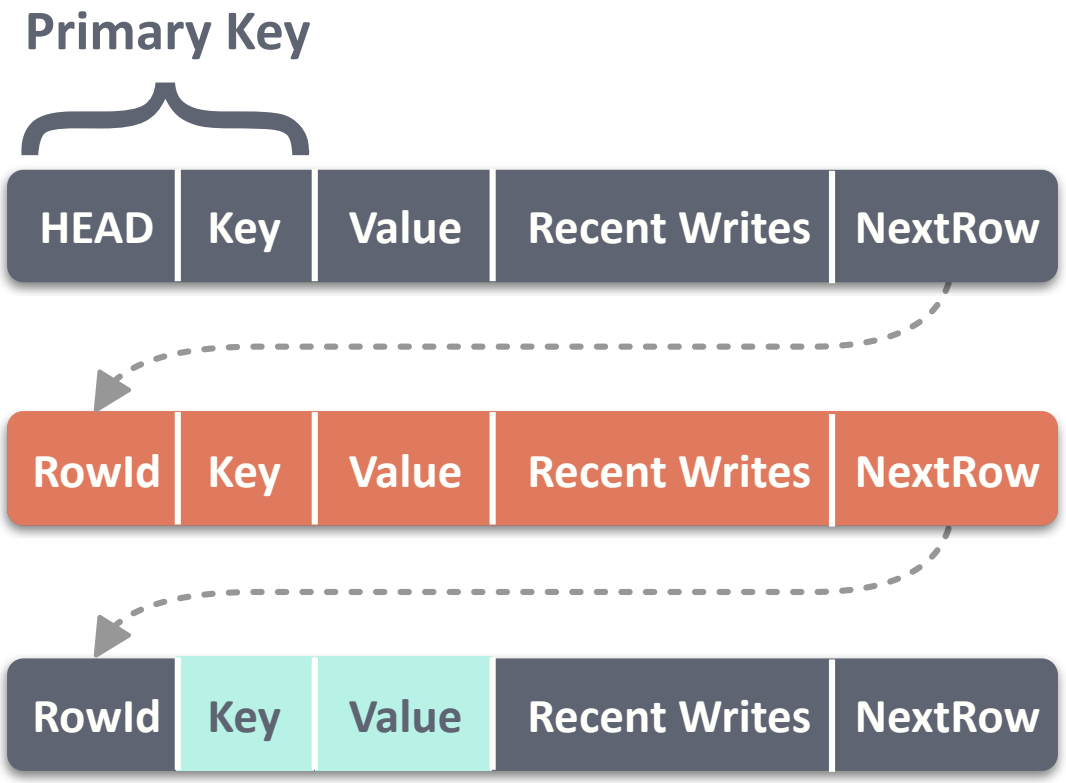
Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

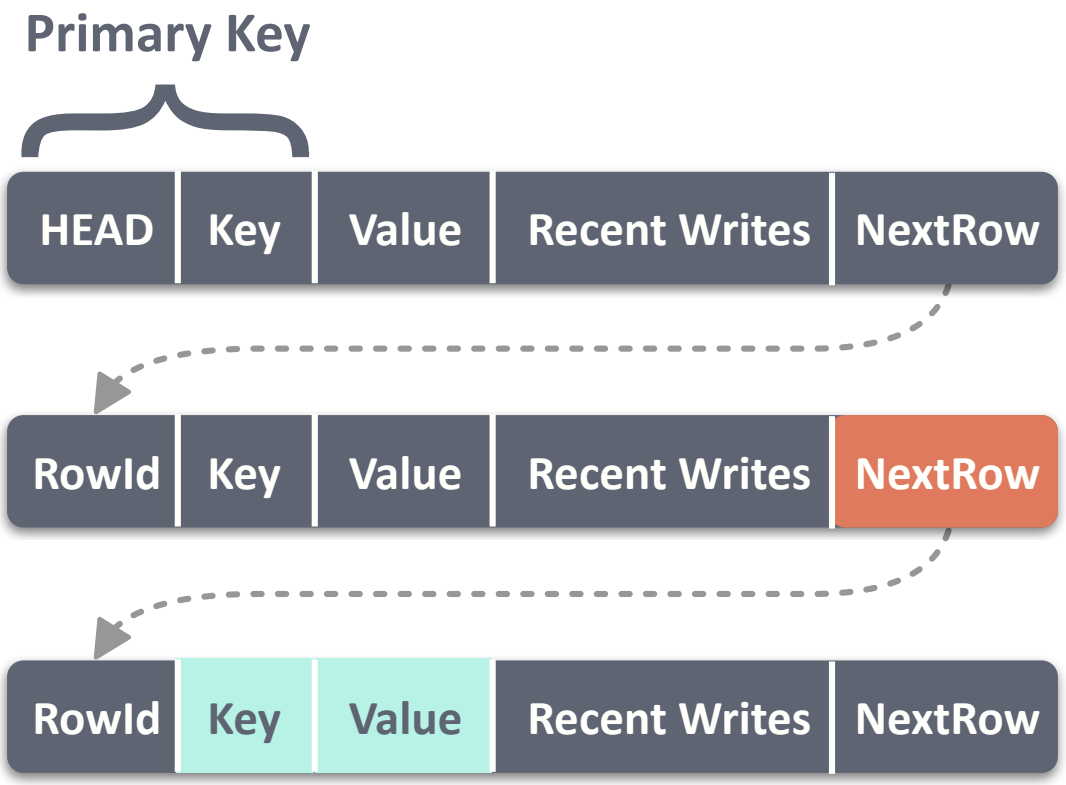
Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

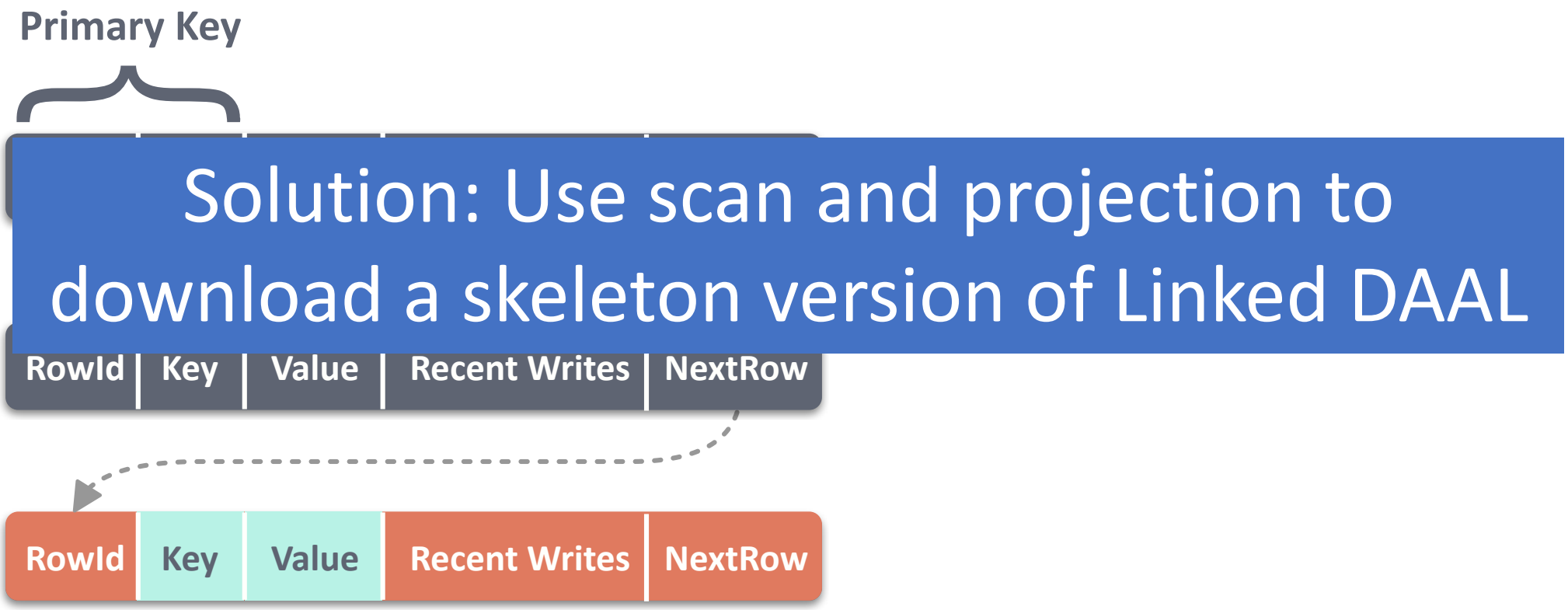
Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

Linked DAAL

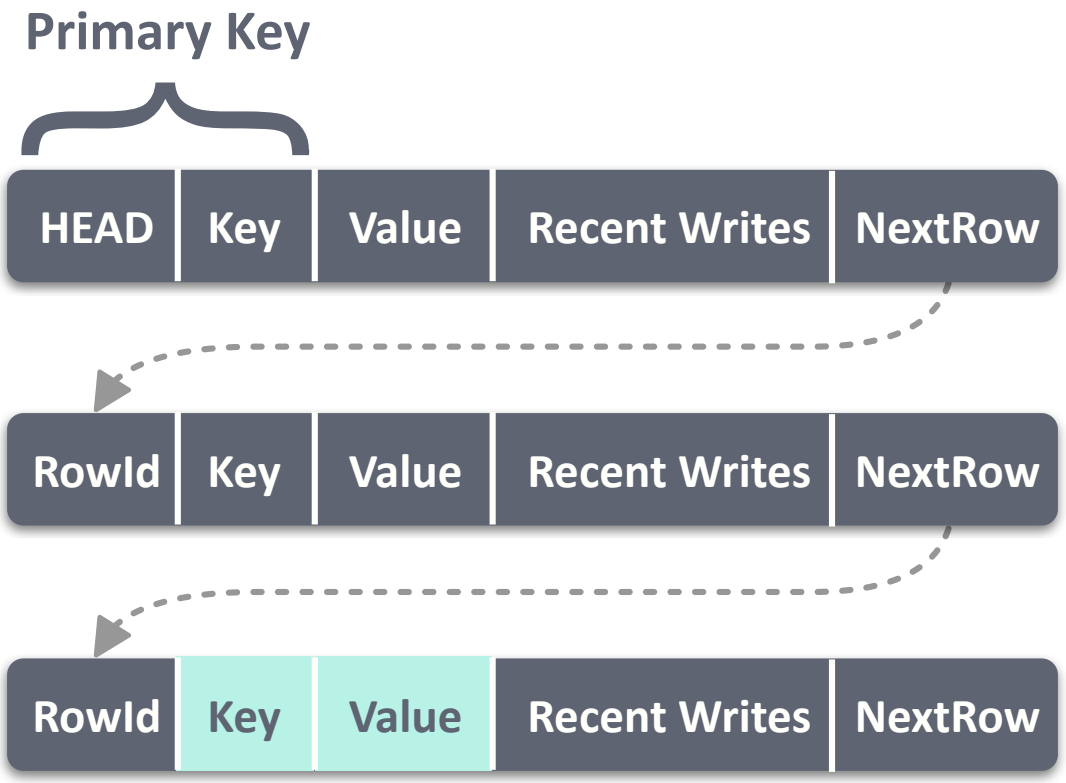

Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

Linked DAAL

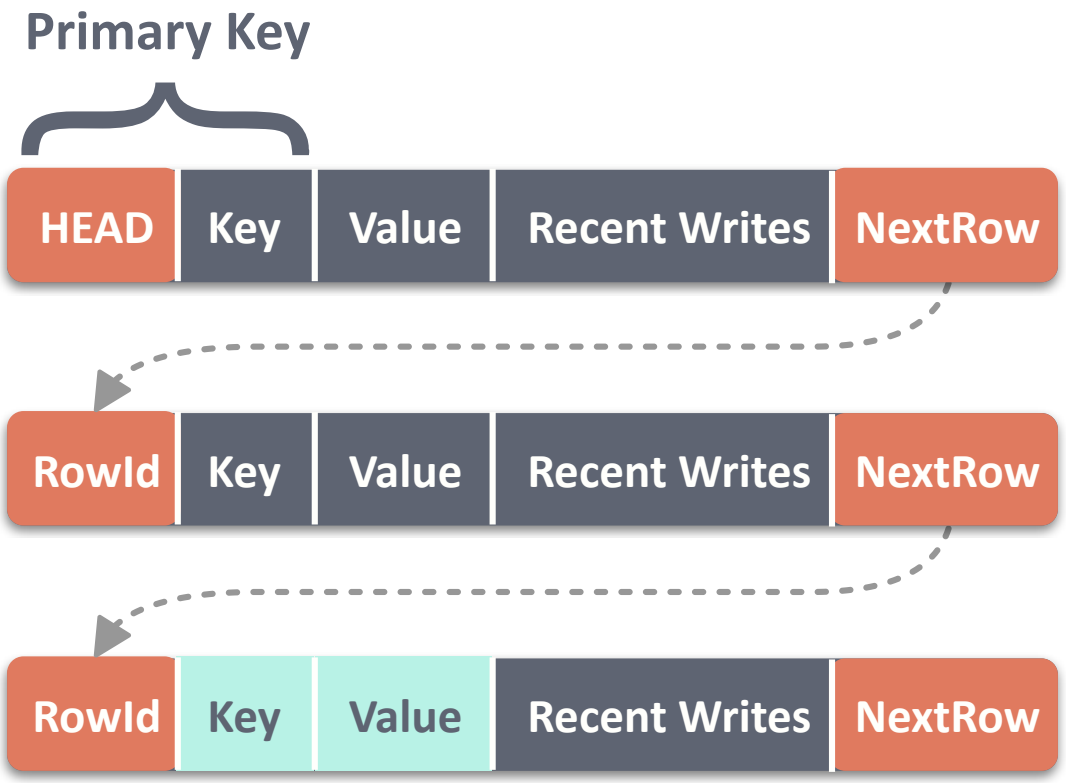


Solution: Use scan and projection to download a skeleton version of Linked DAAL

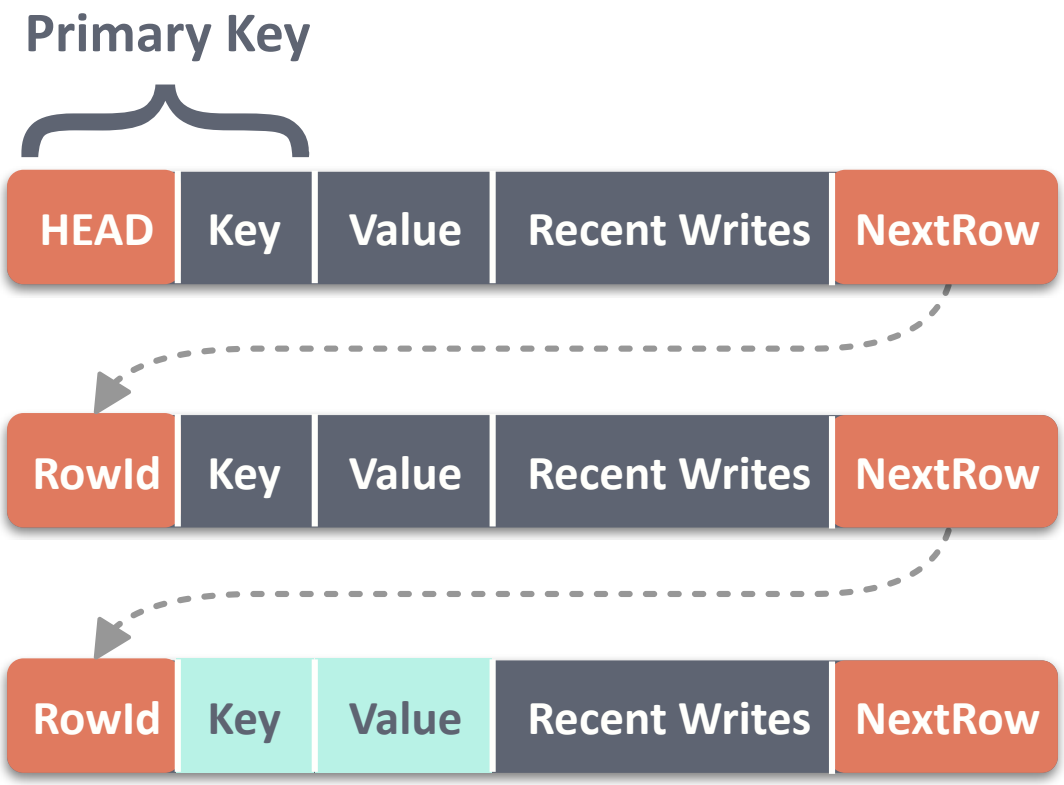
Linked DAAL

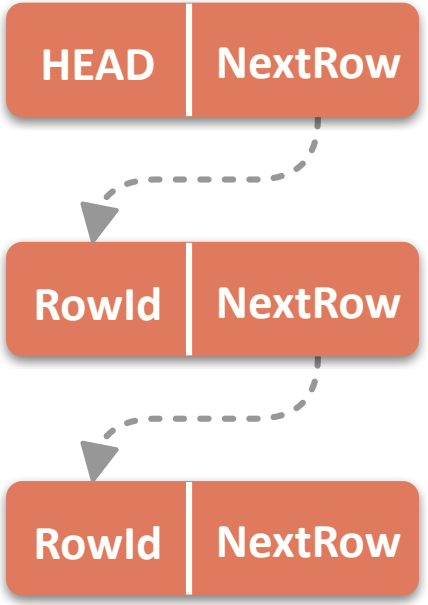

Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow

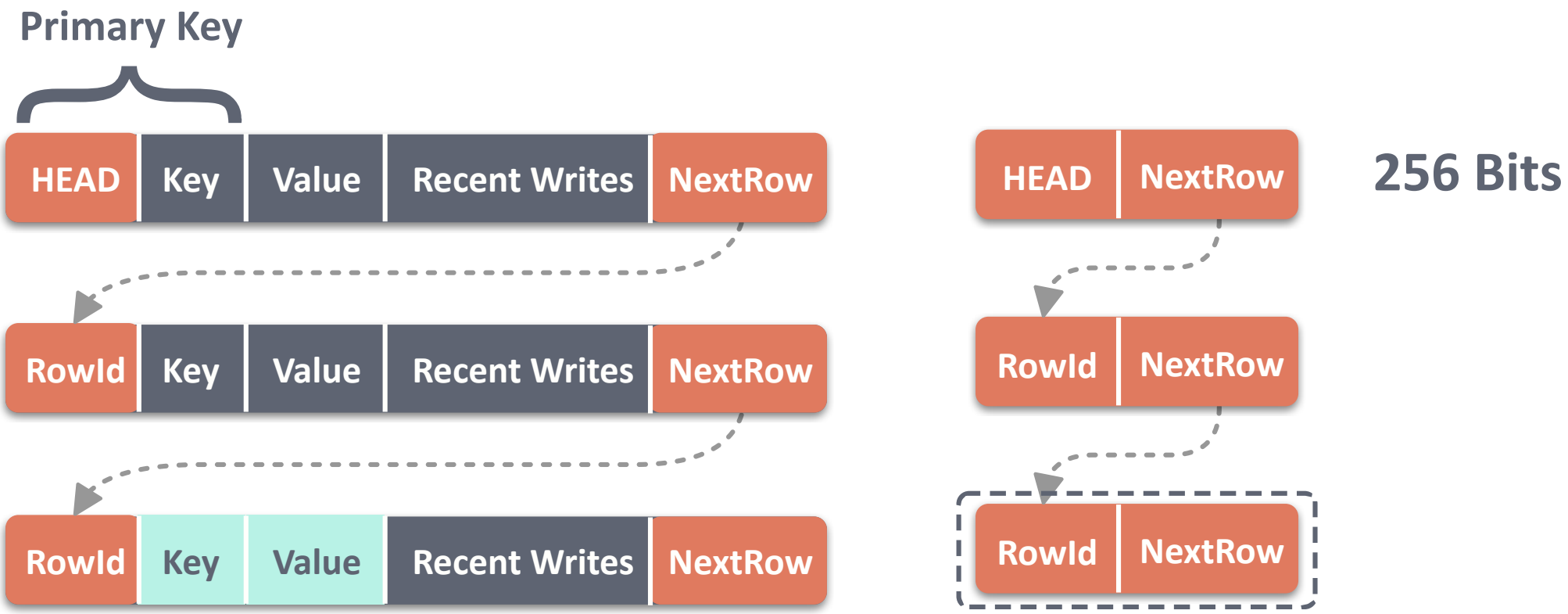
Linked DAAL





256 Bits

Linked DAAL


Primary Key
HEAD
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
RowId
Key
Value
Recent Writes
NextRow
HEAD
NextRow
RowId
NextRow
RowId
NextRow
256 Bits

Outline

- Beldi’s Infrastructure
- Linked DAAL
- **Invocation with exactly-once semantics**
- Evaluation
- Conclusion

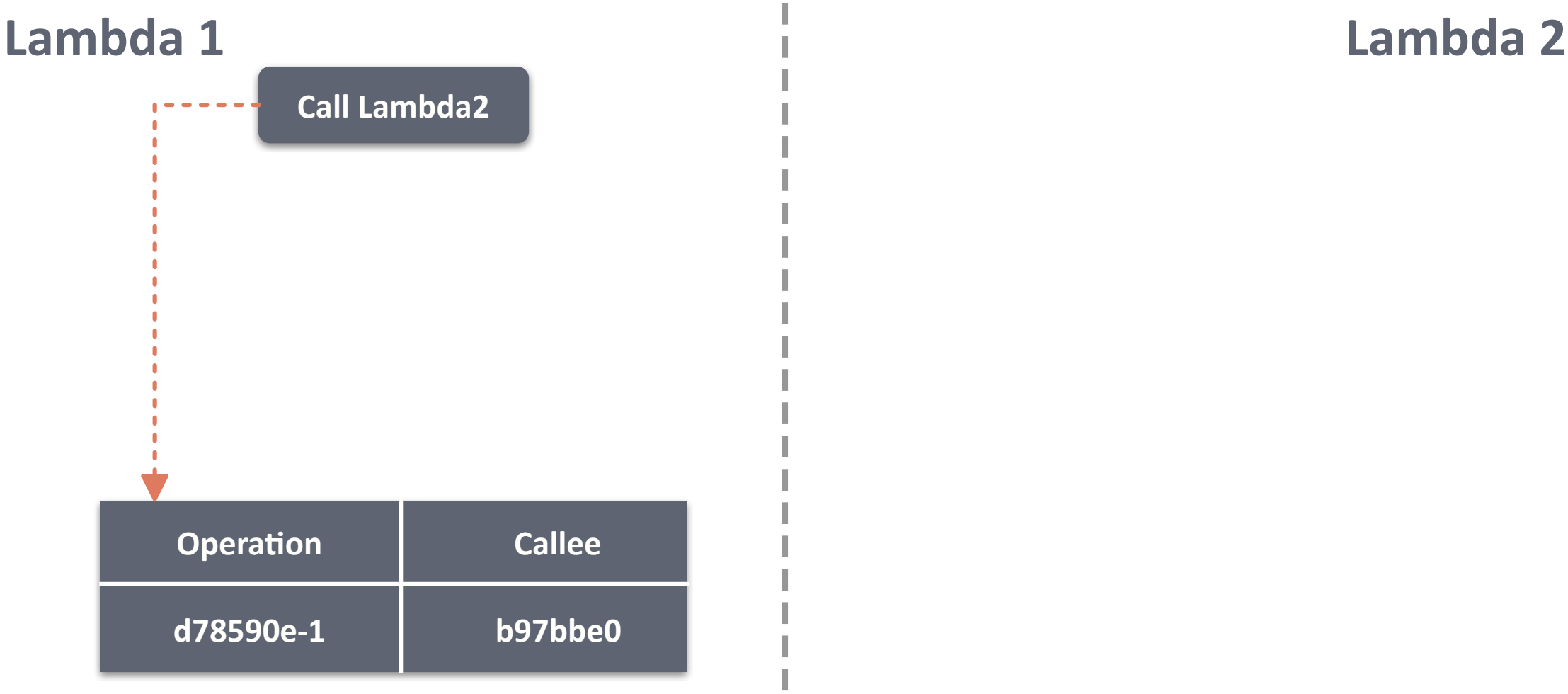
Invocation with exactly-once semantics

Lambda 1

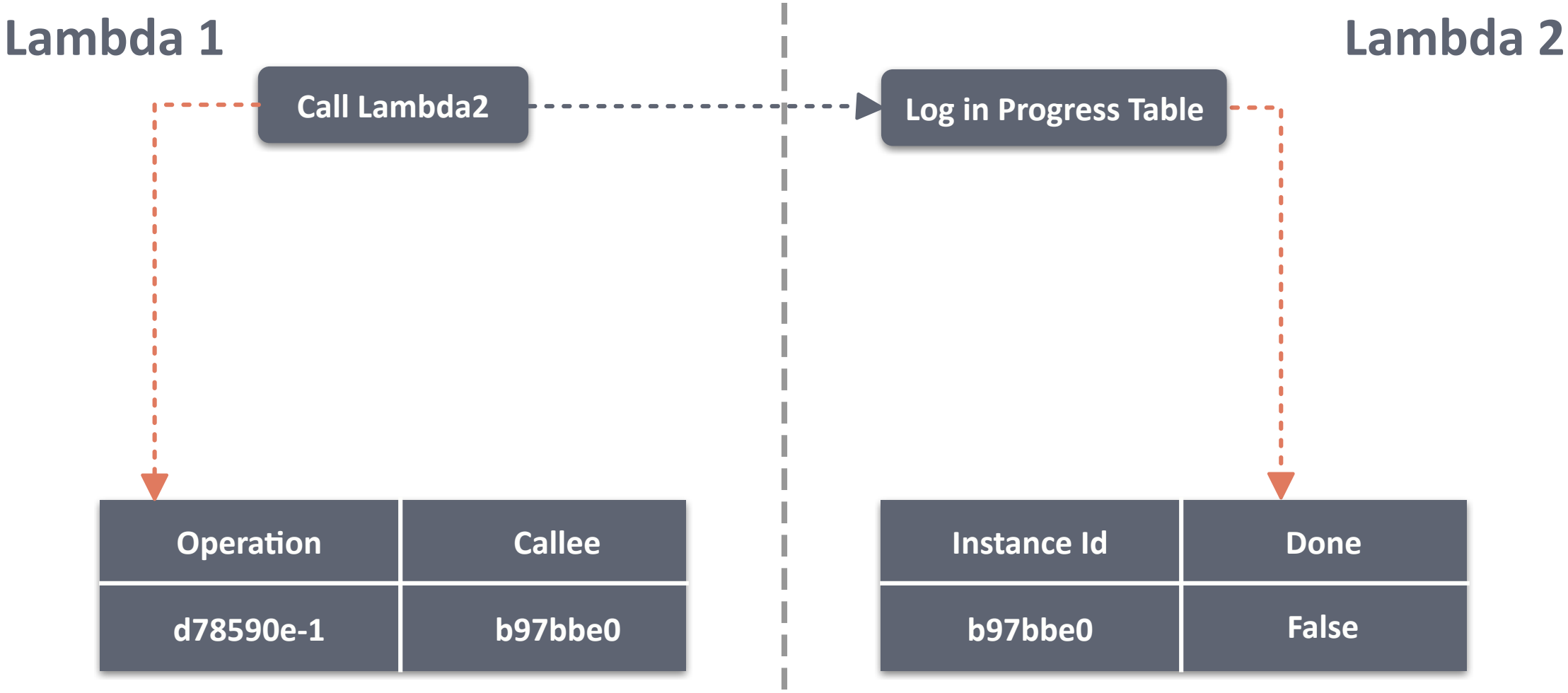
Call Lambda2

Lambda 2

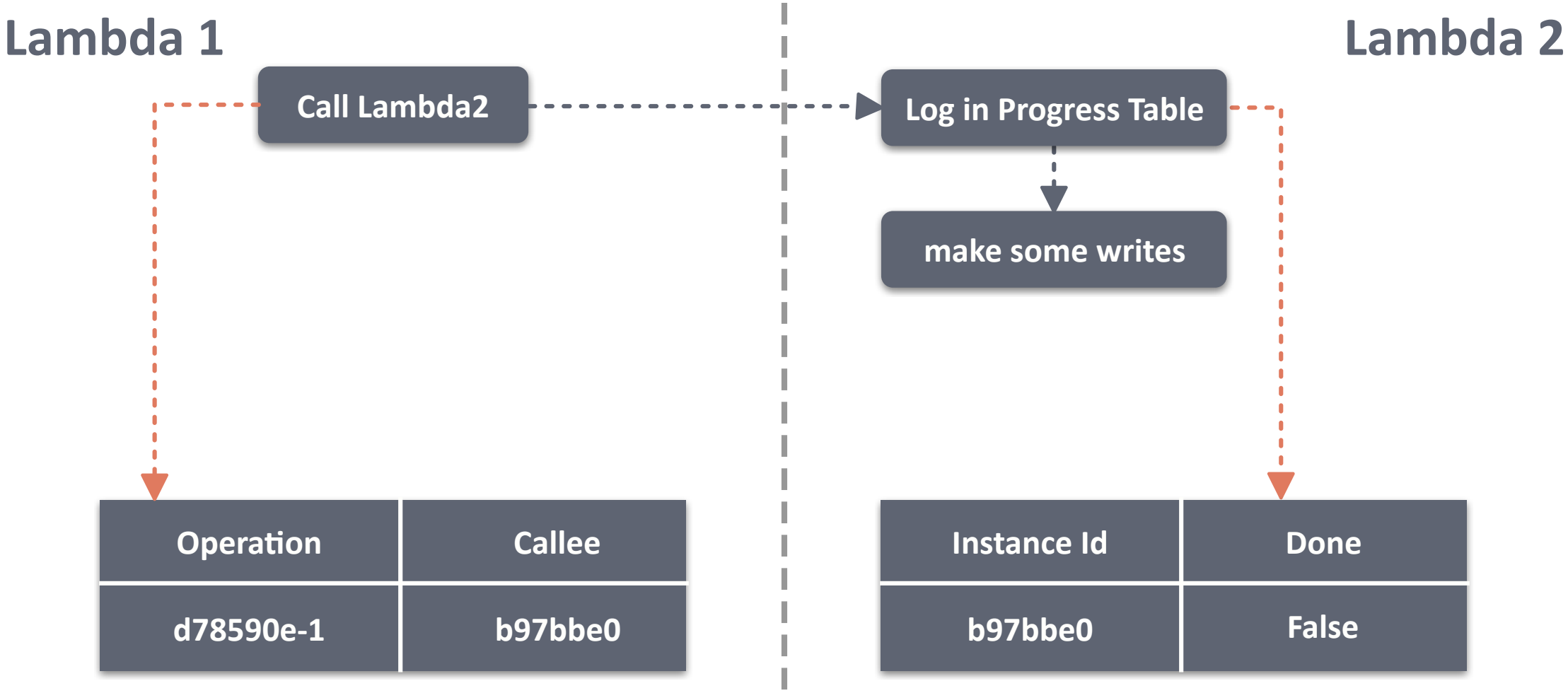
Invocation with exactly-once semantics



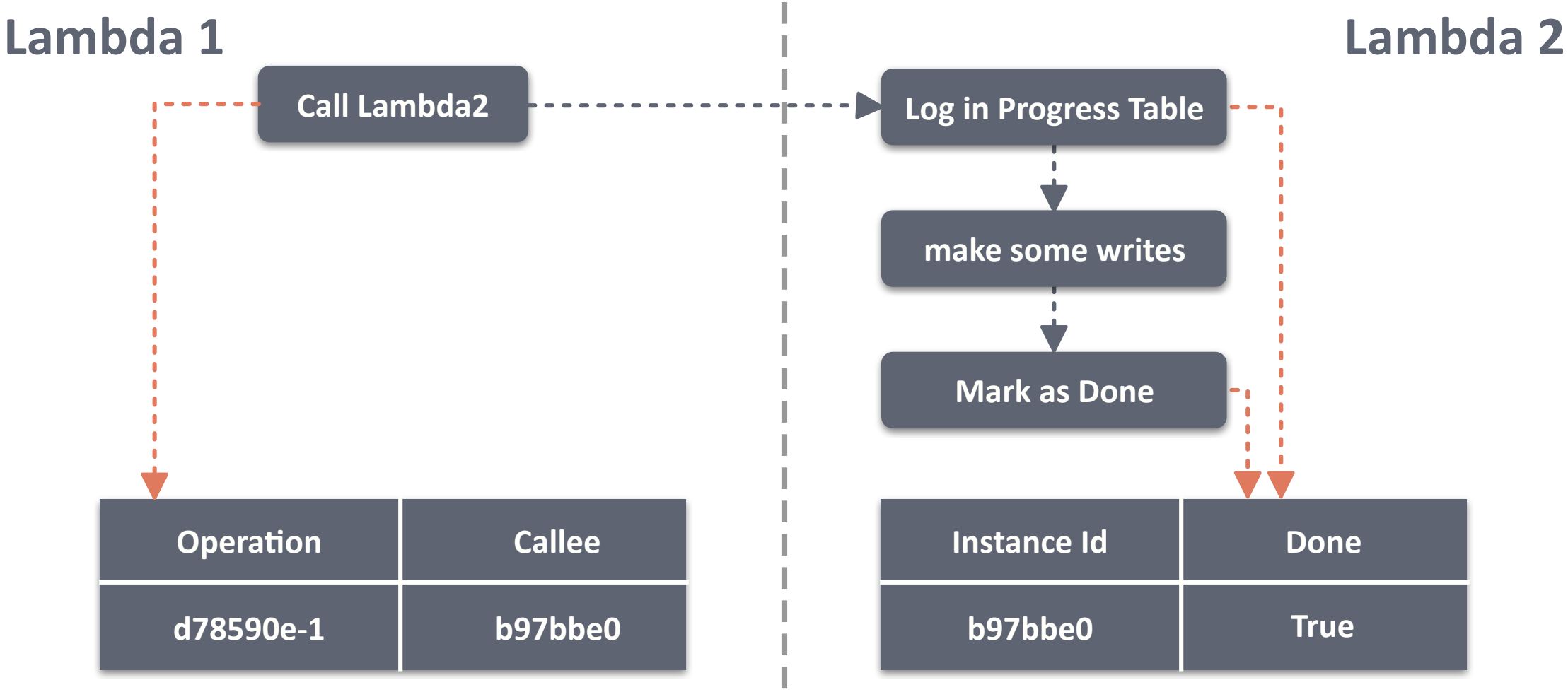
Invocation with exactly-once semantics



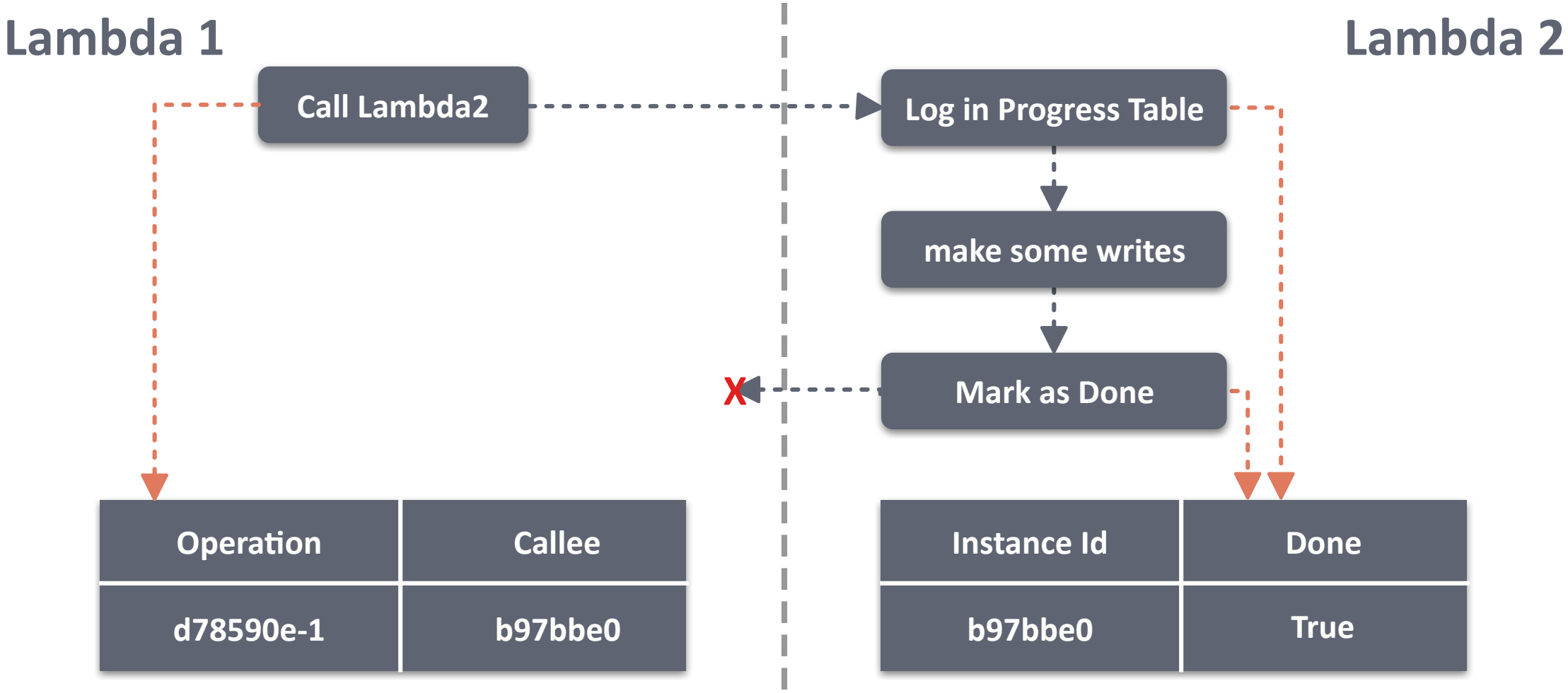
Invocation with exactly-once semantics



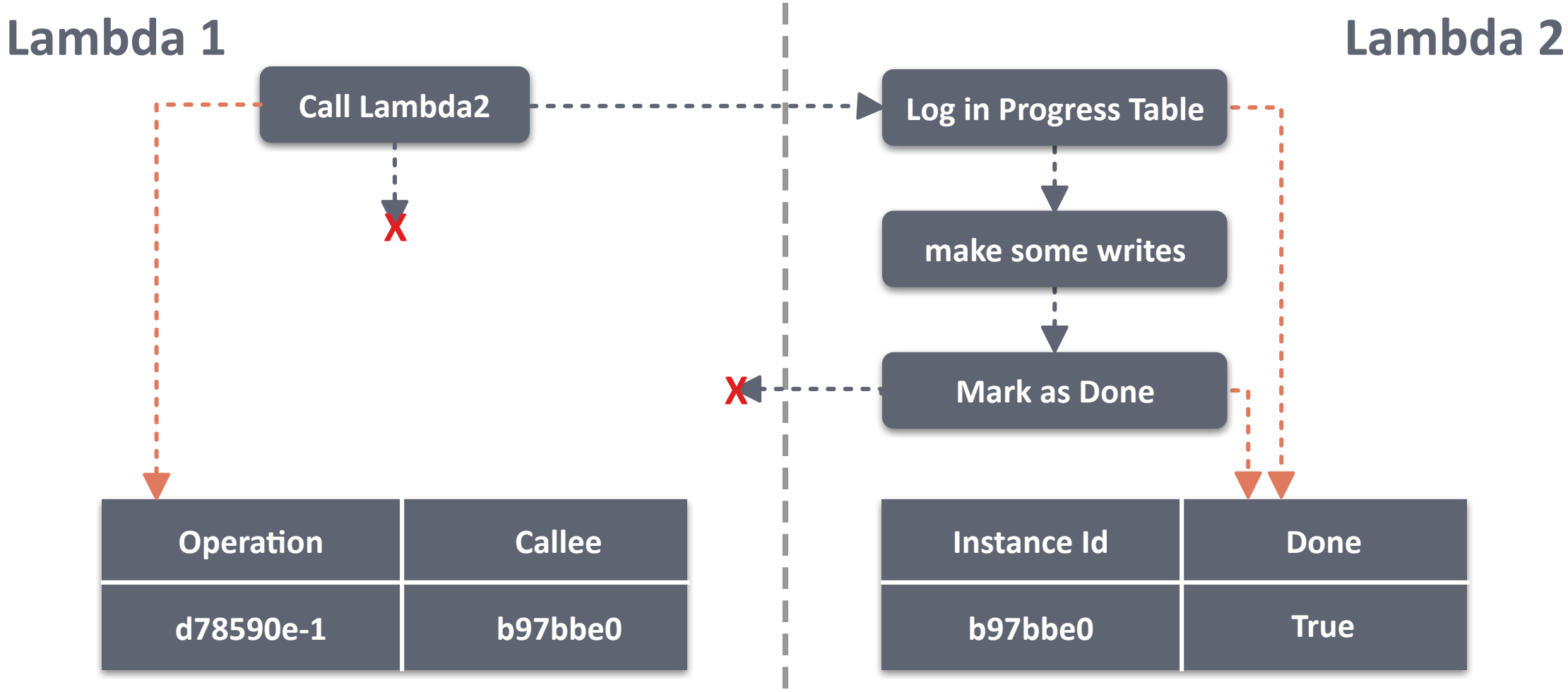
Invocation with exactly-once semantics



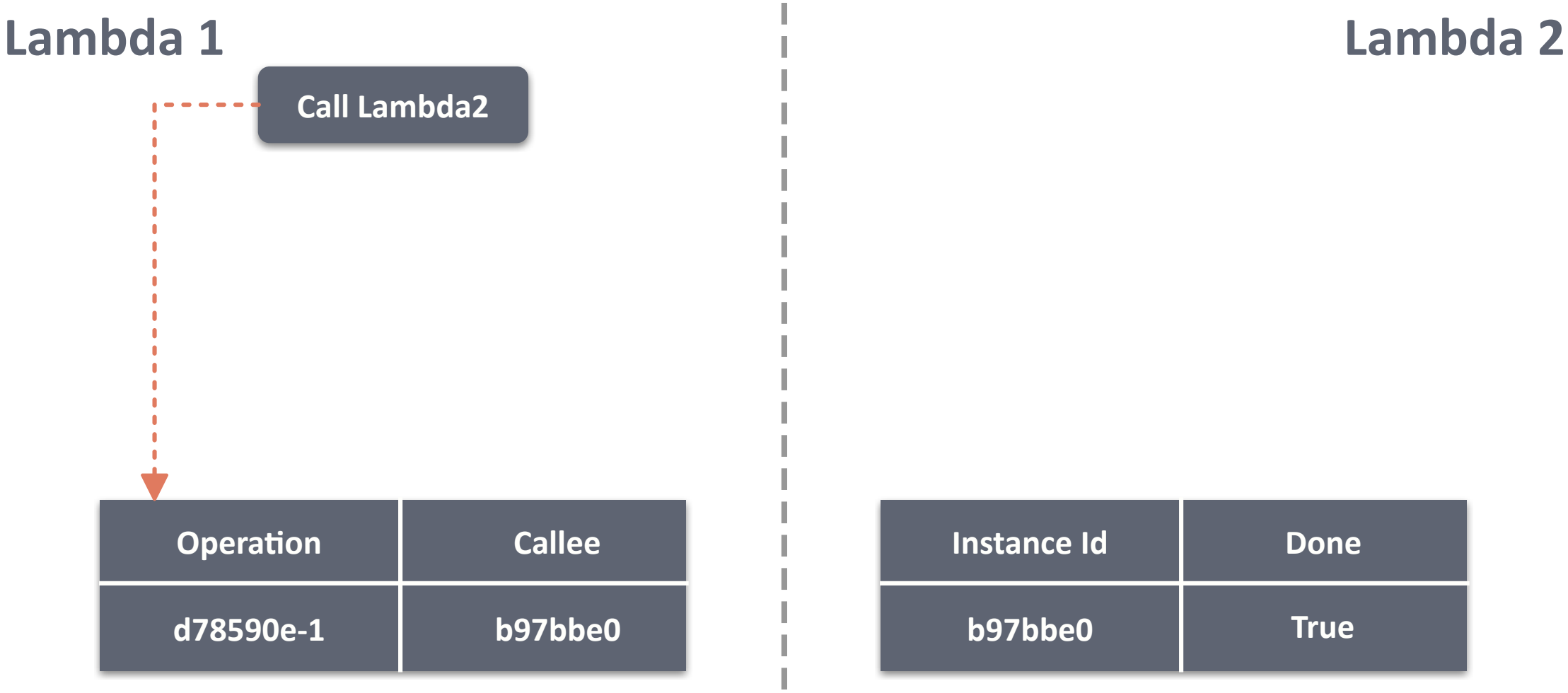
Invocation with exactly-once semantics



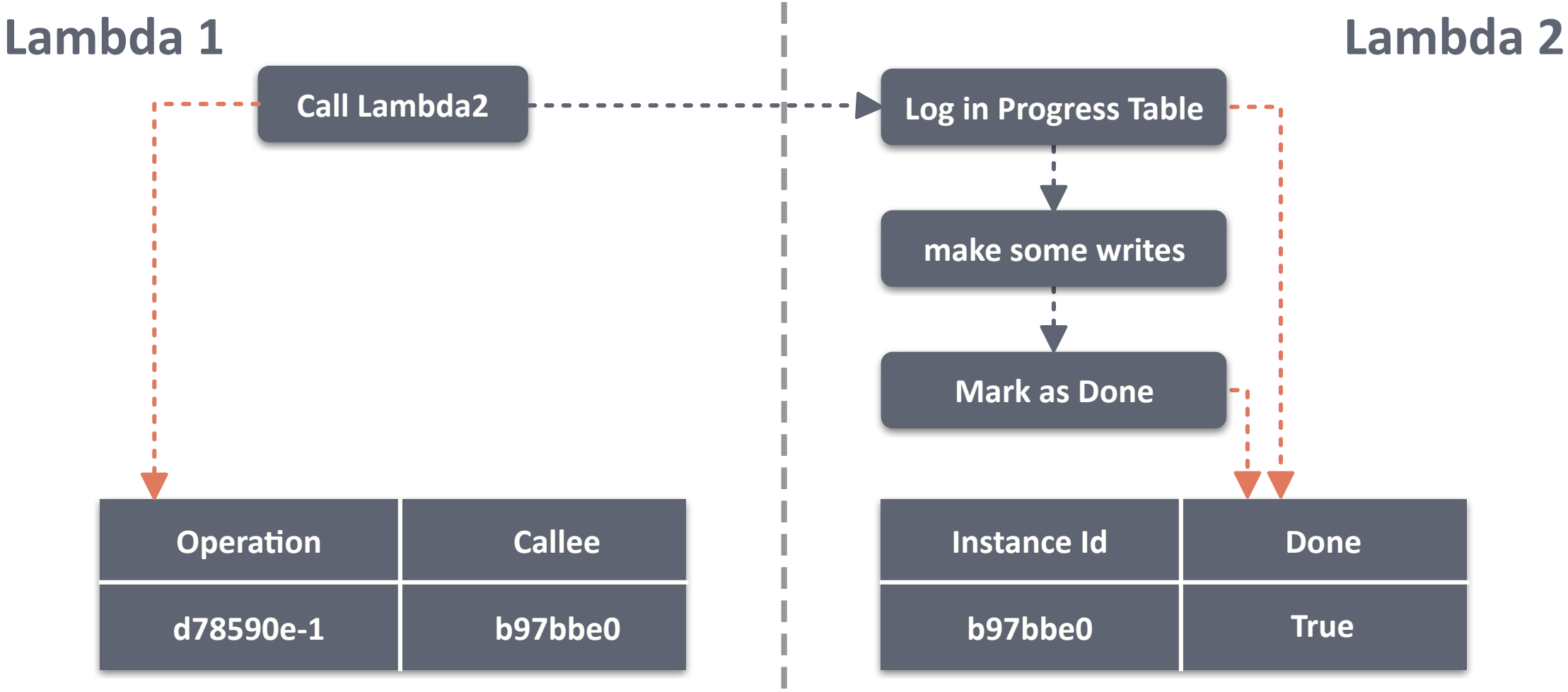
Invocation with exactly-once semantics



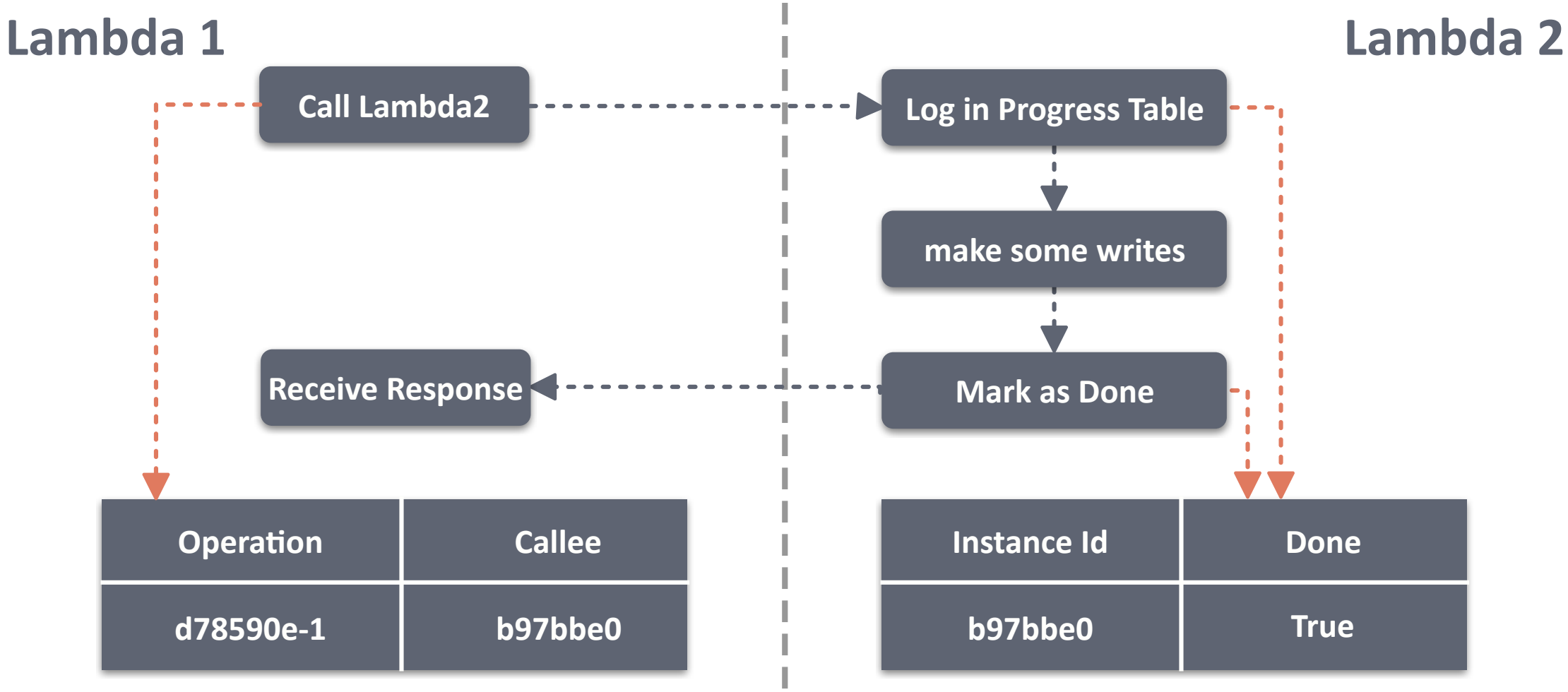
Invocation with exactly-once semantics



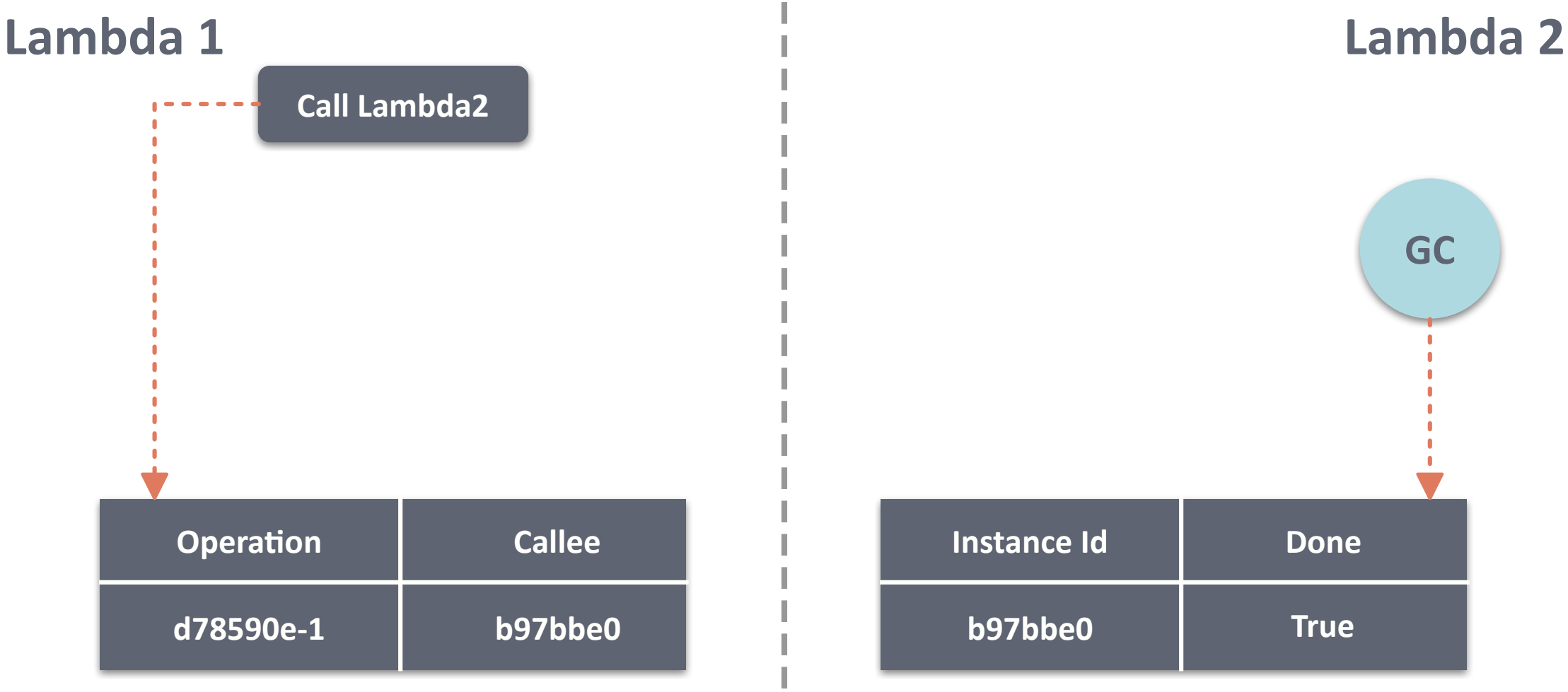
Invocation with exactly-once semantics



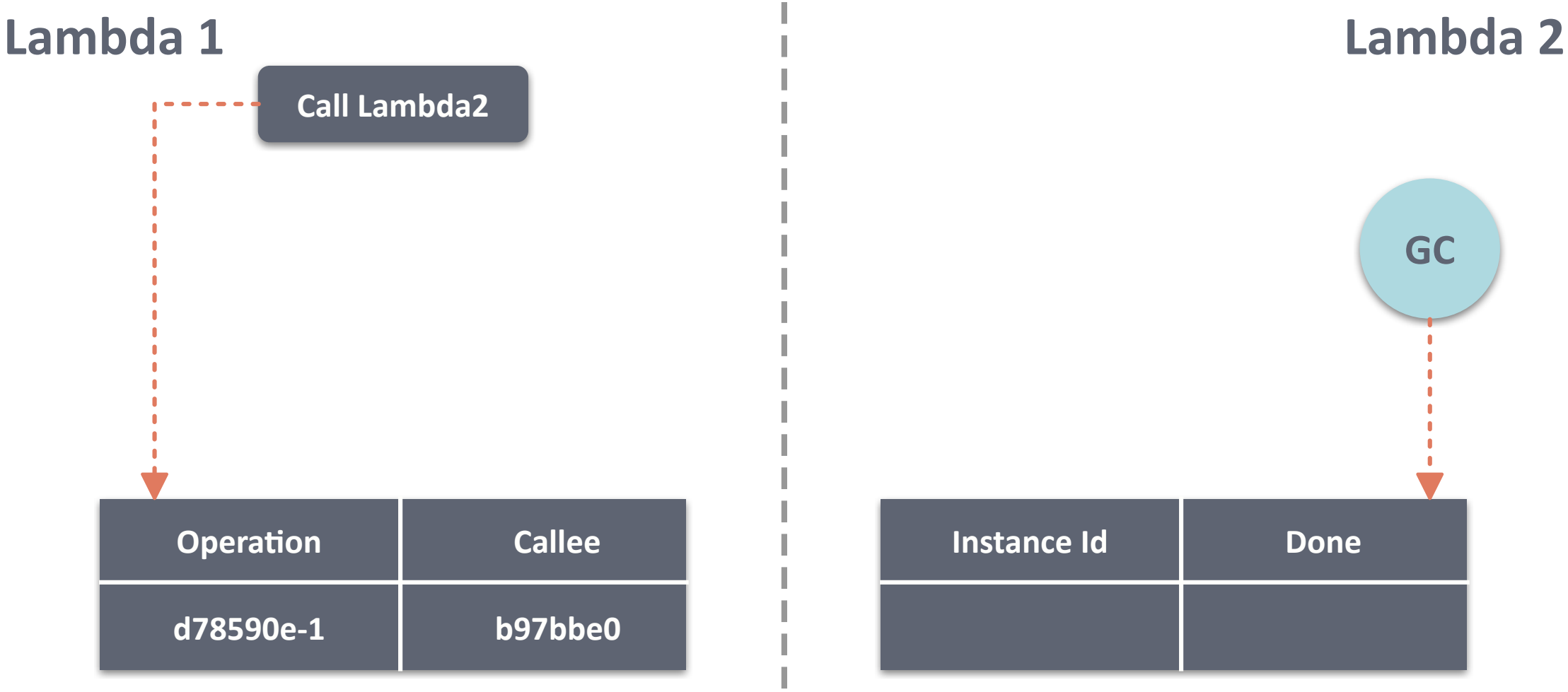
Invocation with exactly-once semantics



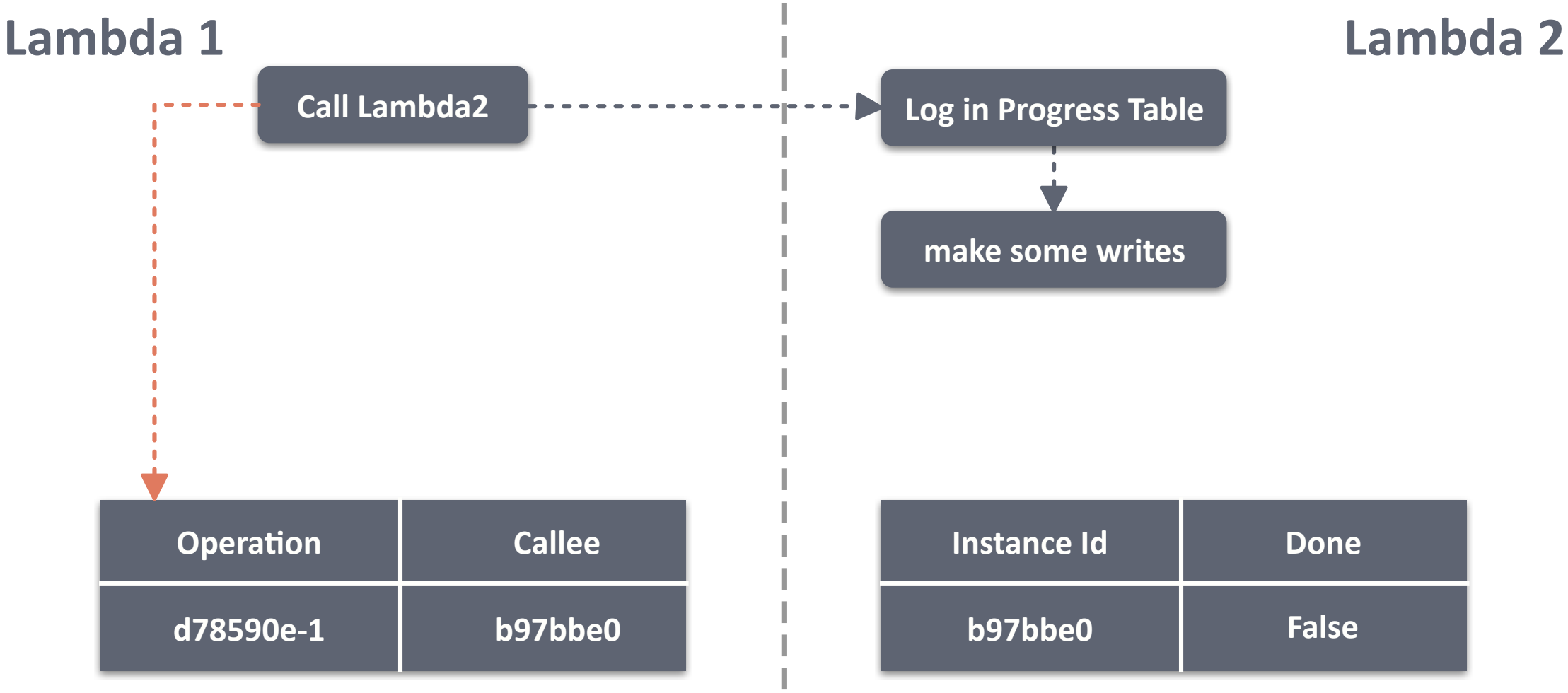
Invocation with exactly-once semantics



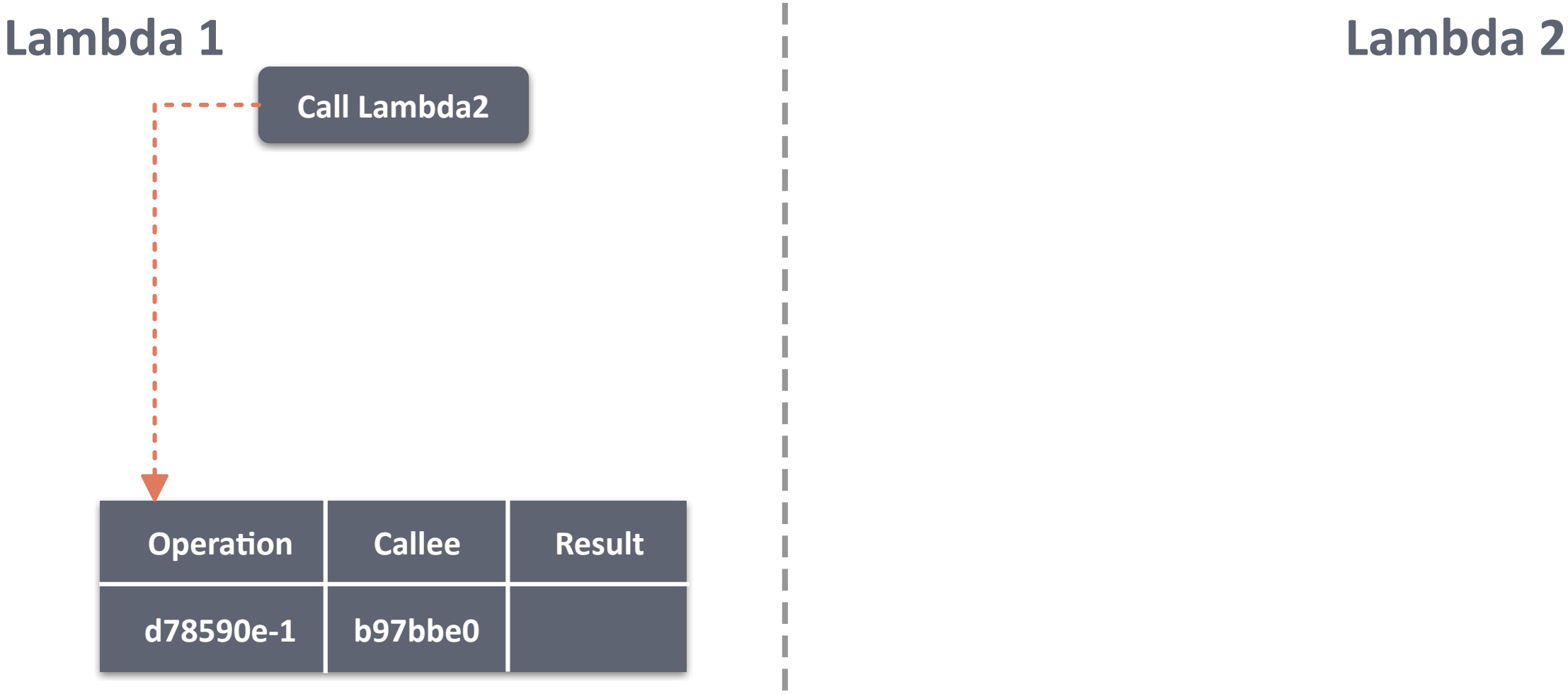
Invocation with exactly-once semantics



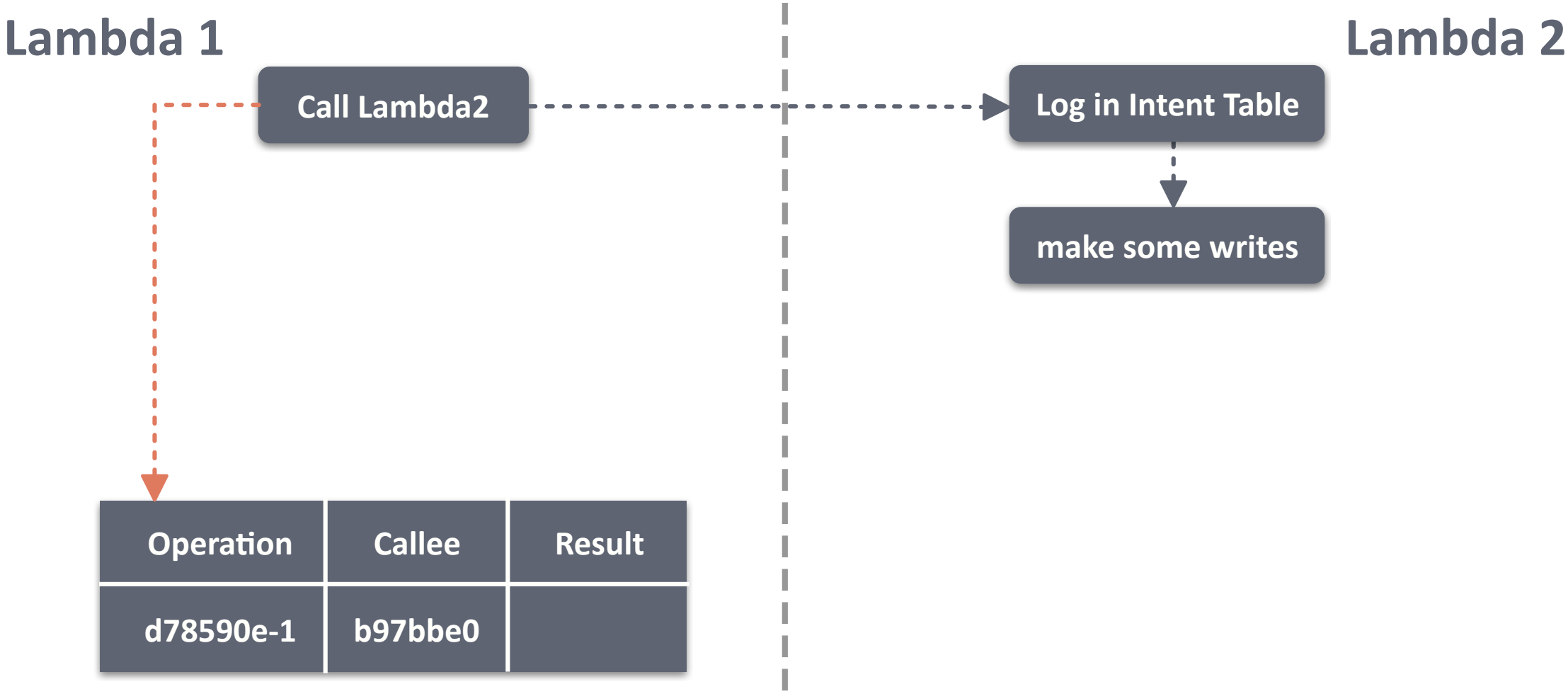
Invocation with exactly-once semantics



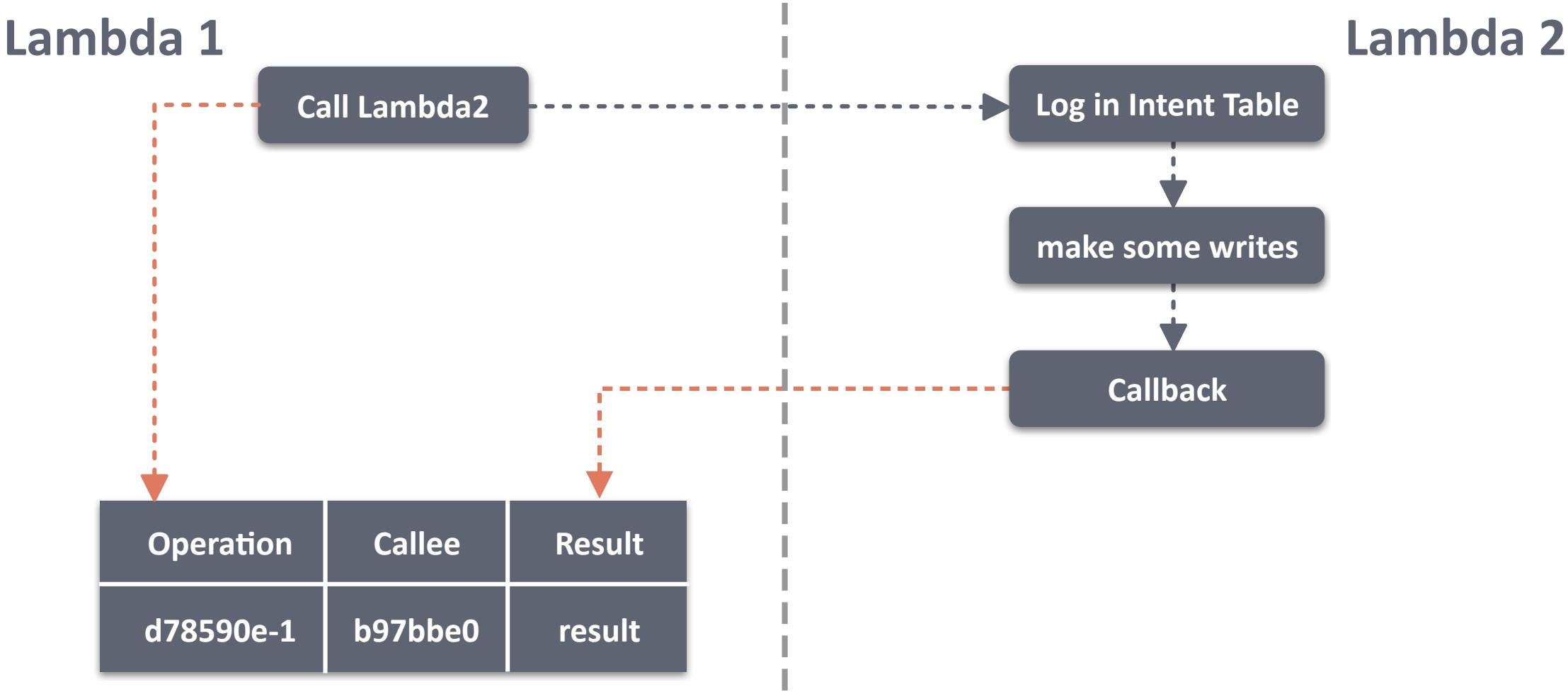
Invocation with exactly-once semantics



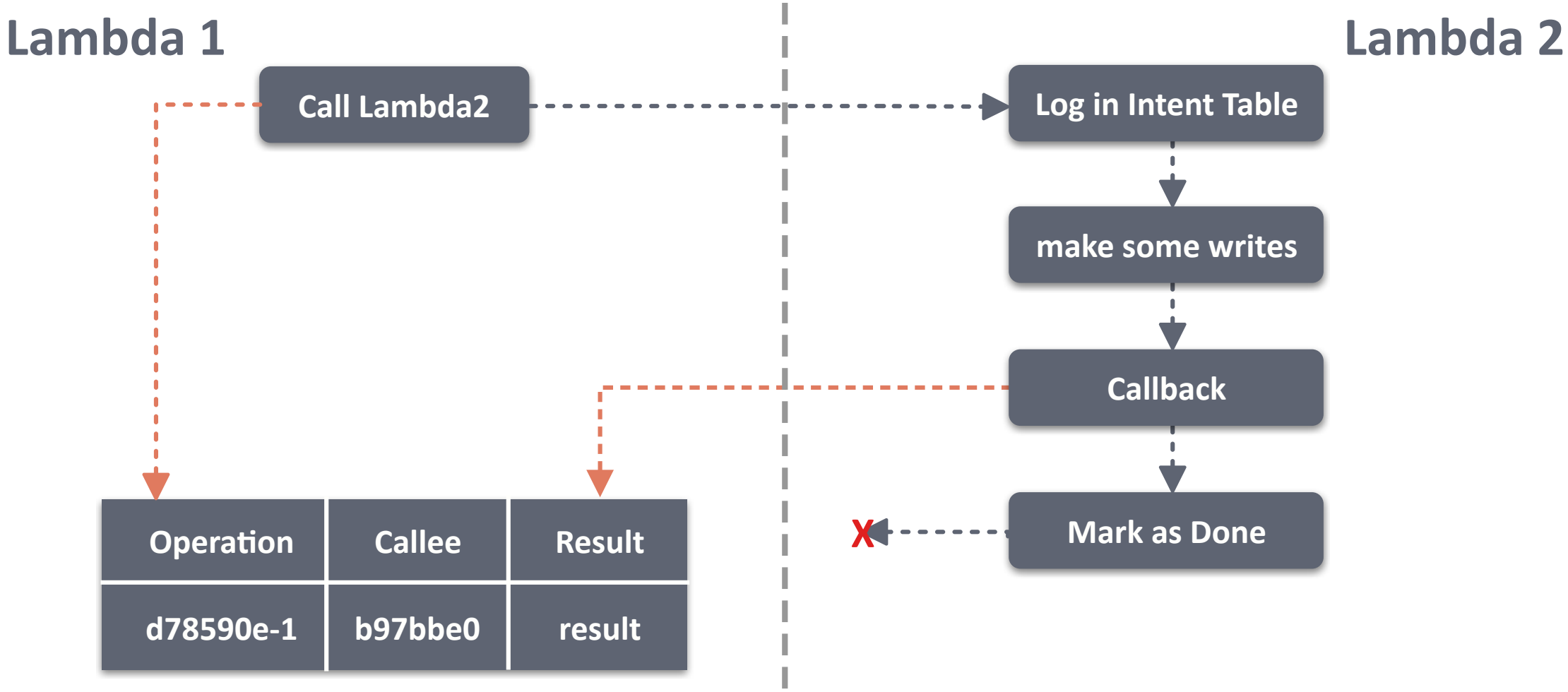
Invocation with exactly-once semantics



Invocation with exactly-once semantics



Invocation with exactly-once semantics



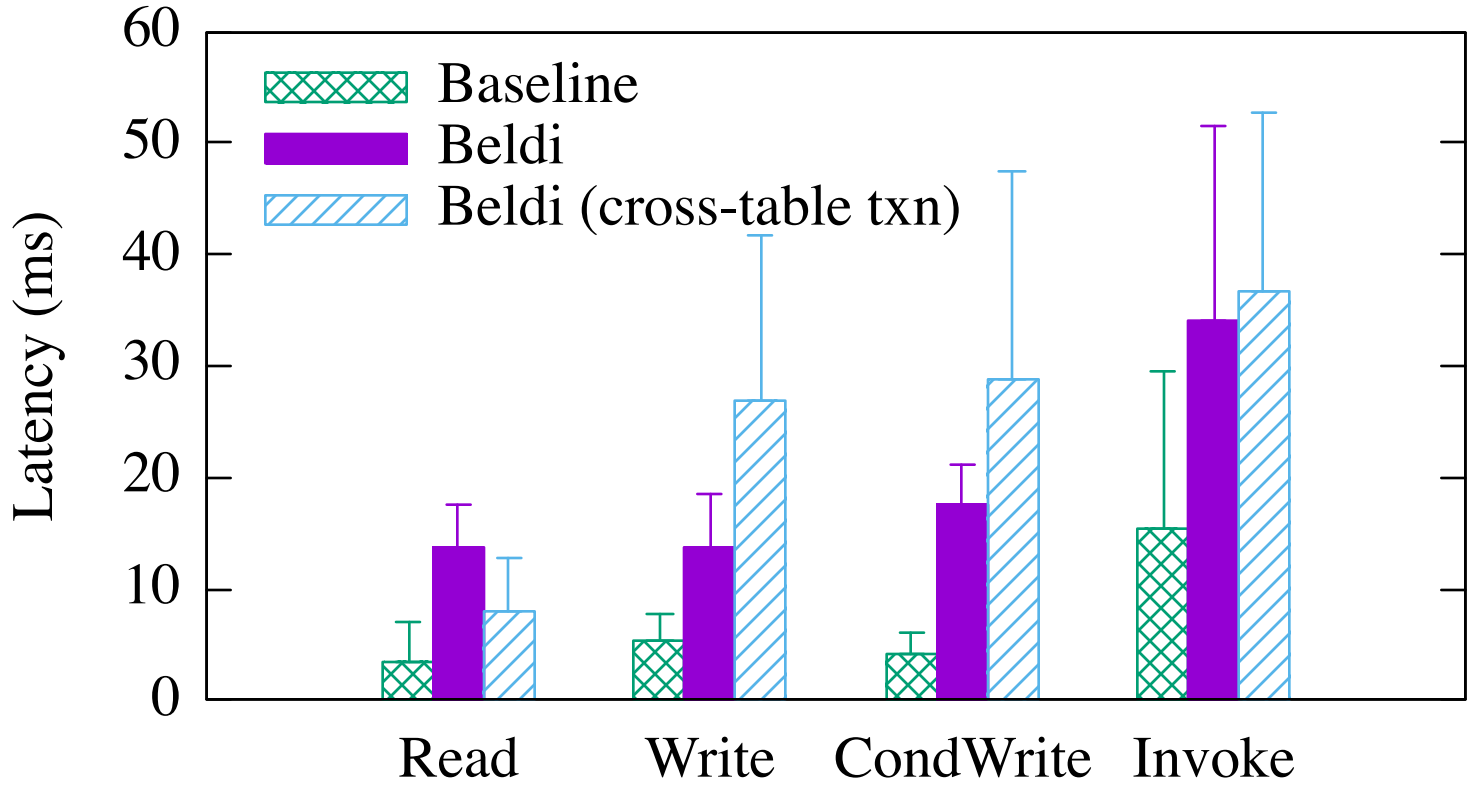
Outline

- Beldi’s Infrastructure
- Linked DAAL
- Invocation with exactly-once semantics
- **Evaluation**
- Conclusion

Evaluation

1. What are the costs of Beldi's API operations?
2. How does Beldi perform in real-world applications?
3. What is the effect of garbage collection?

What are the costs of Beldi's API operations?

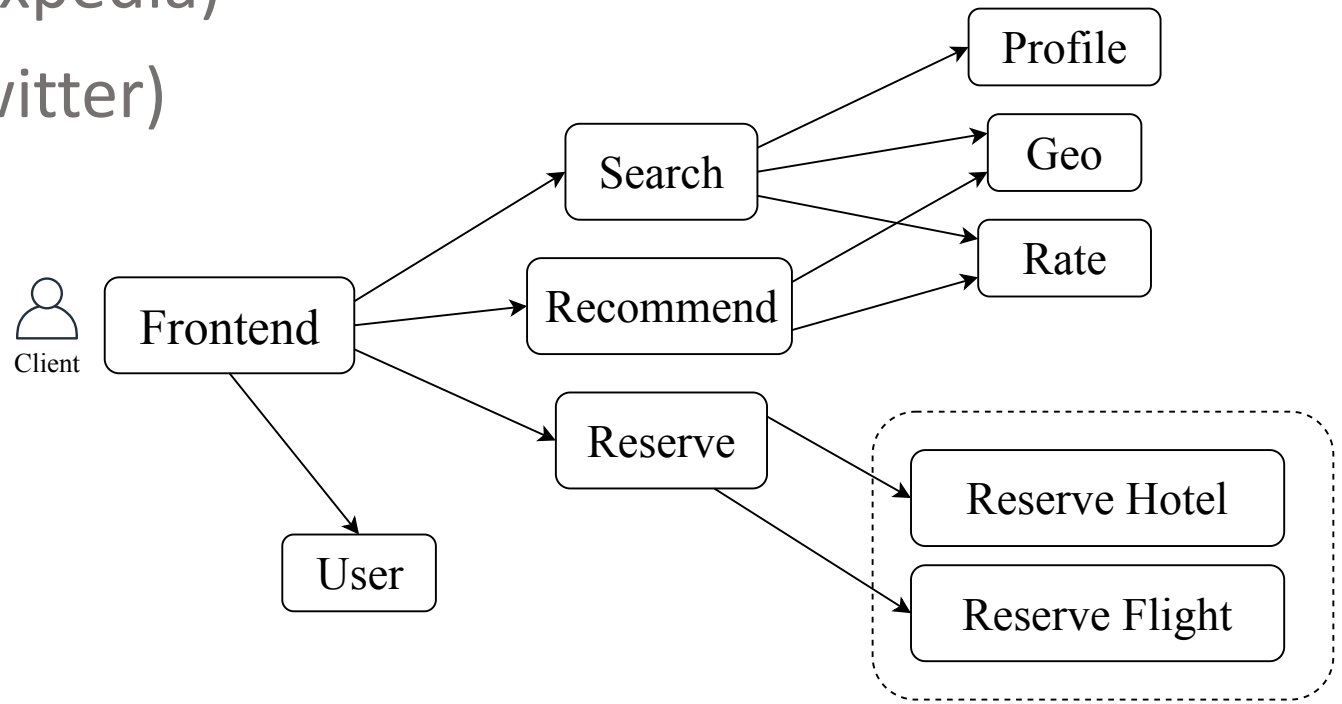


20 rows in Linked DAAL, 2 - 4x more expensive than baseline

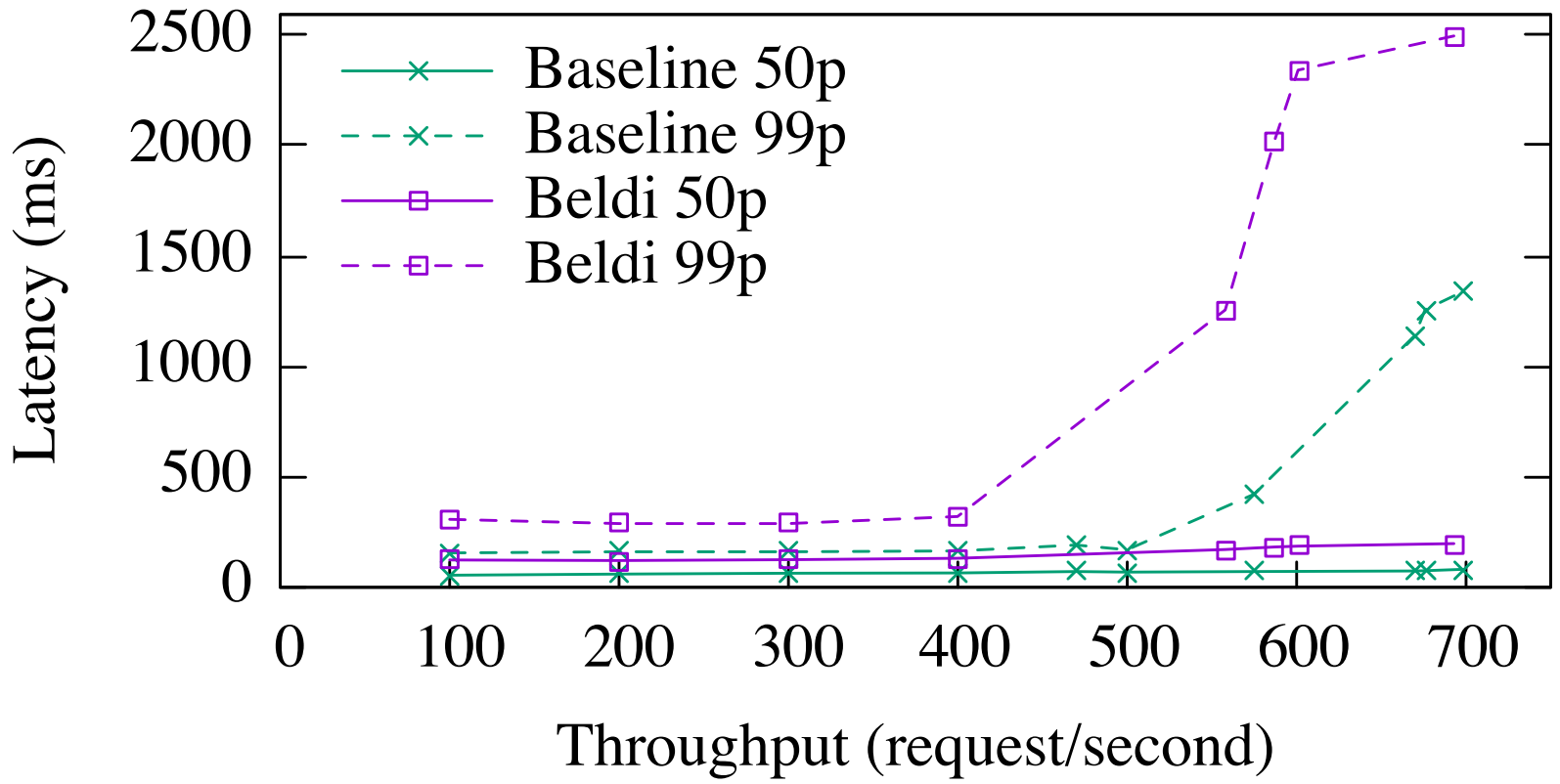
How does Beldi perform in real-world applications?

DeathStarBench (*ASPLOS 19*): open-source microservices benchmark

- Movie review service (Cf. IMDB)
- Travel reservation (Cf. Expedia)
- Social media site (Cf. Twitter)



How does Beldi perform in real-world applications?



<400 req/s:

2× higher than baseline

700 req/s (saturation):

3.3 × higher than baseline

Outline

- Beldi’s Infrastructure
- Linked DAAL
- Invocation with exactly-once semantics
- Evaluation
- **Conclusion**

Conclusion

1. A framework to write transactional and fault-tolerant applications on serverless.
2. A lock-free data structure (Linked DAAL) to support fast logging and exactly-once semantics
3. A collaborative distributed transaction protocol across multiple lambdas
4. An efficient garbage collection algorithm that runs independently without affecting running lambdas or requiring any pauses.

https://github.com/eniac/beldi

Thank you!