Move Fast and Meet Deadlines: Fine-grained Real-time Dataflow Scheduling with Cameo

Le Xu, Shivaram Venkataraman, Indranil Gupta, Luo Mai, Rahul Potharaju

NSDI 2021

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

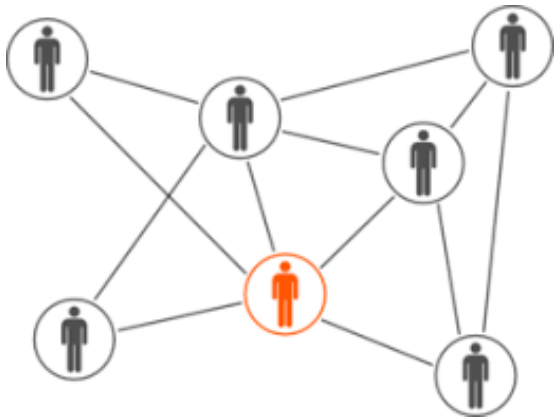
WISCONSIN UNIVERSITY OF WISCONSIN–MADISON

THE UNIVERSITY of EDINBURGH

Microsoft

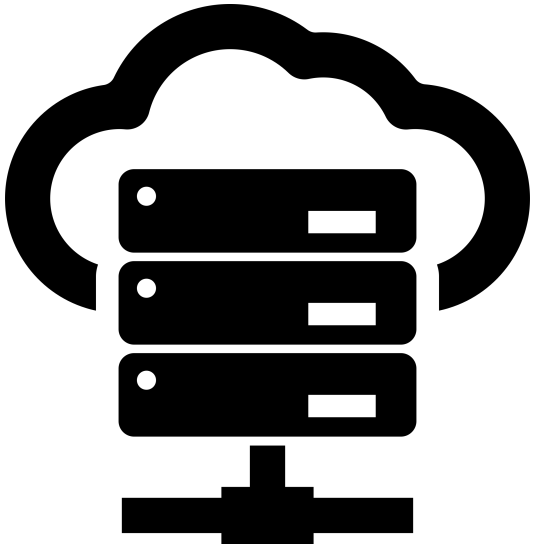
Forecast

Financial Services



Social Networks

Edge and IoT

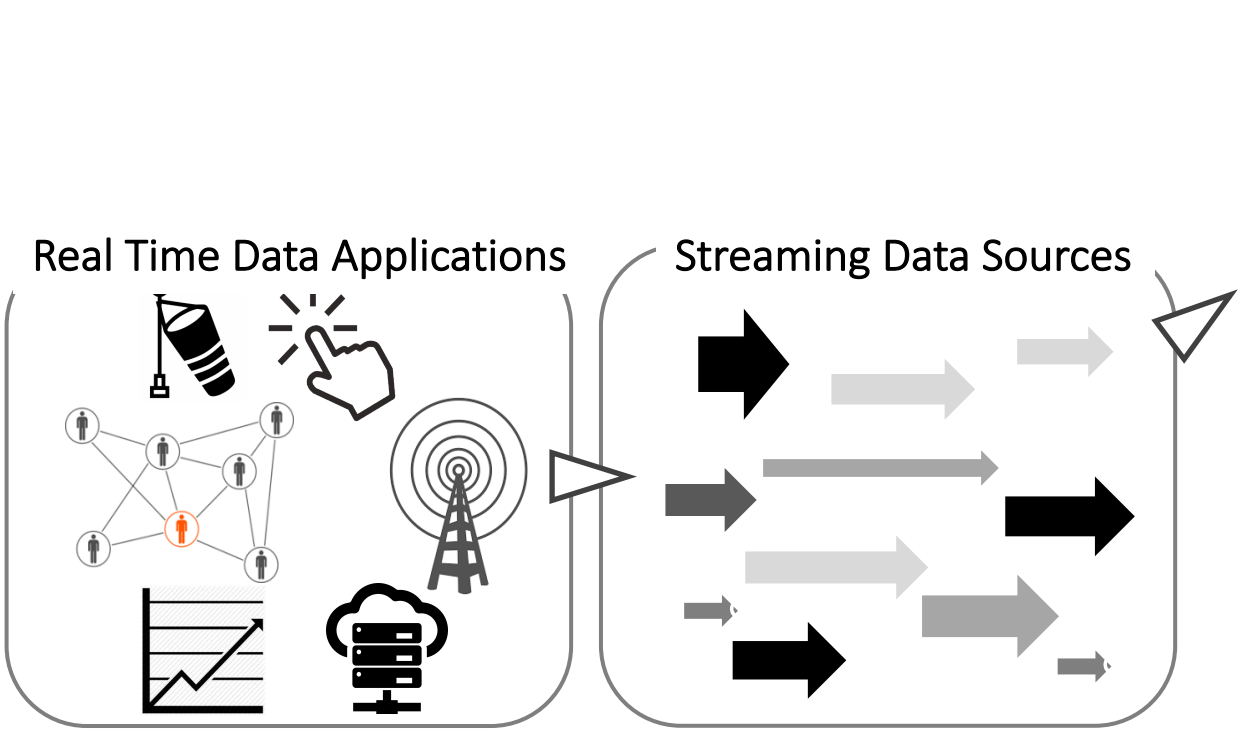


Cloud Management

Web Applications

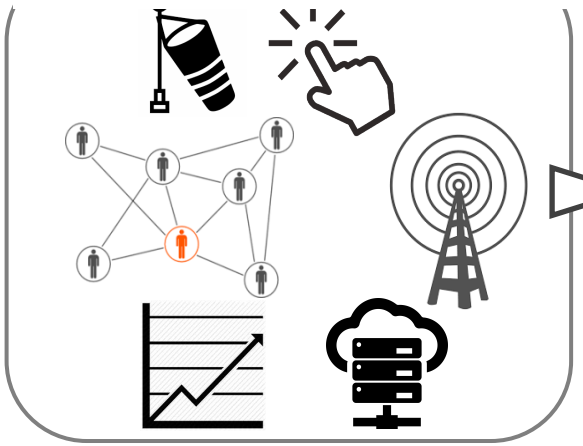
Real time data stream processing systems

Real time data stream processing systems

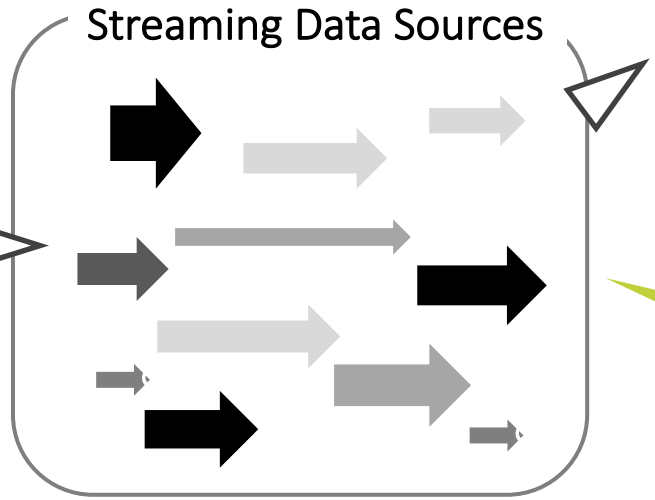


Real time data stream processing systems

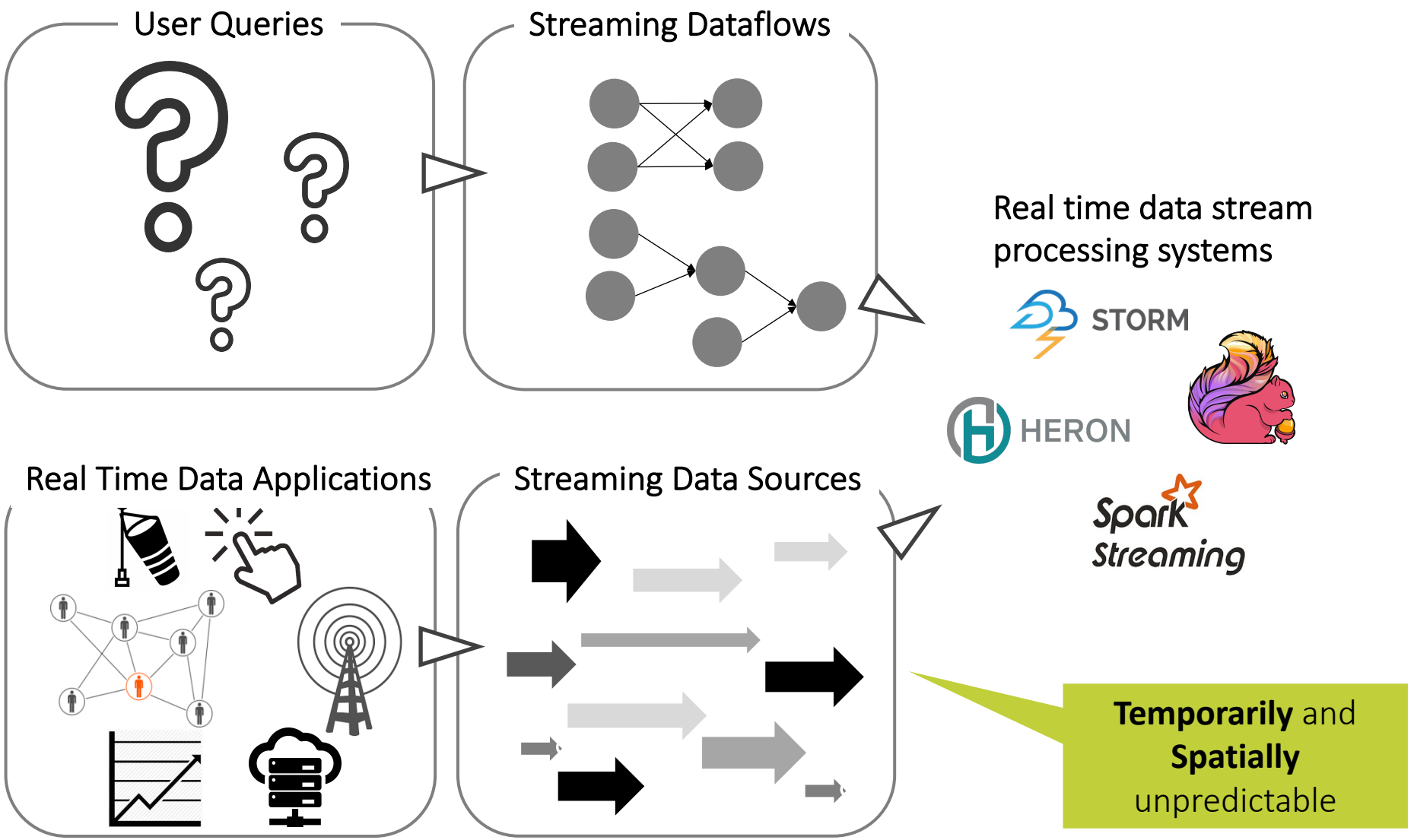
Real Time Data Applications

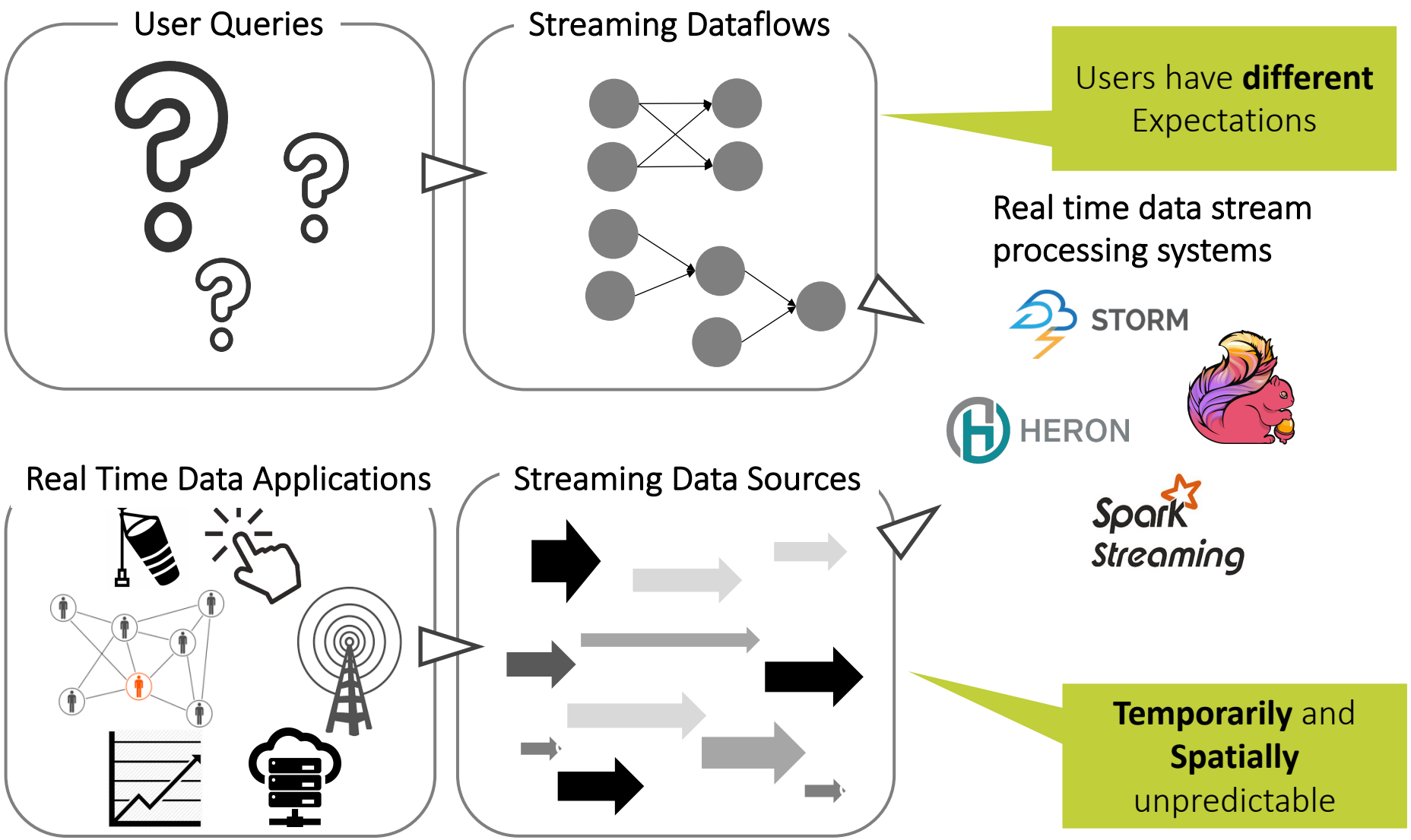


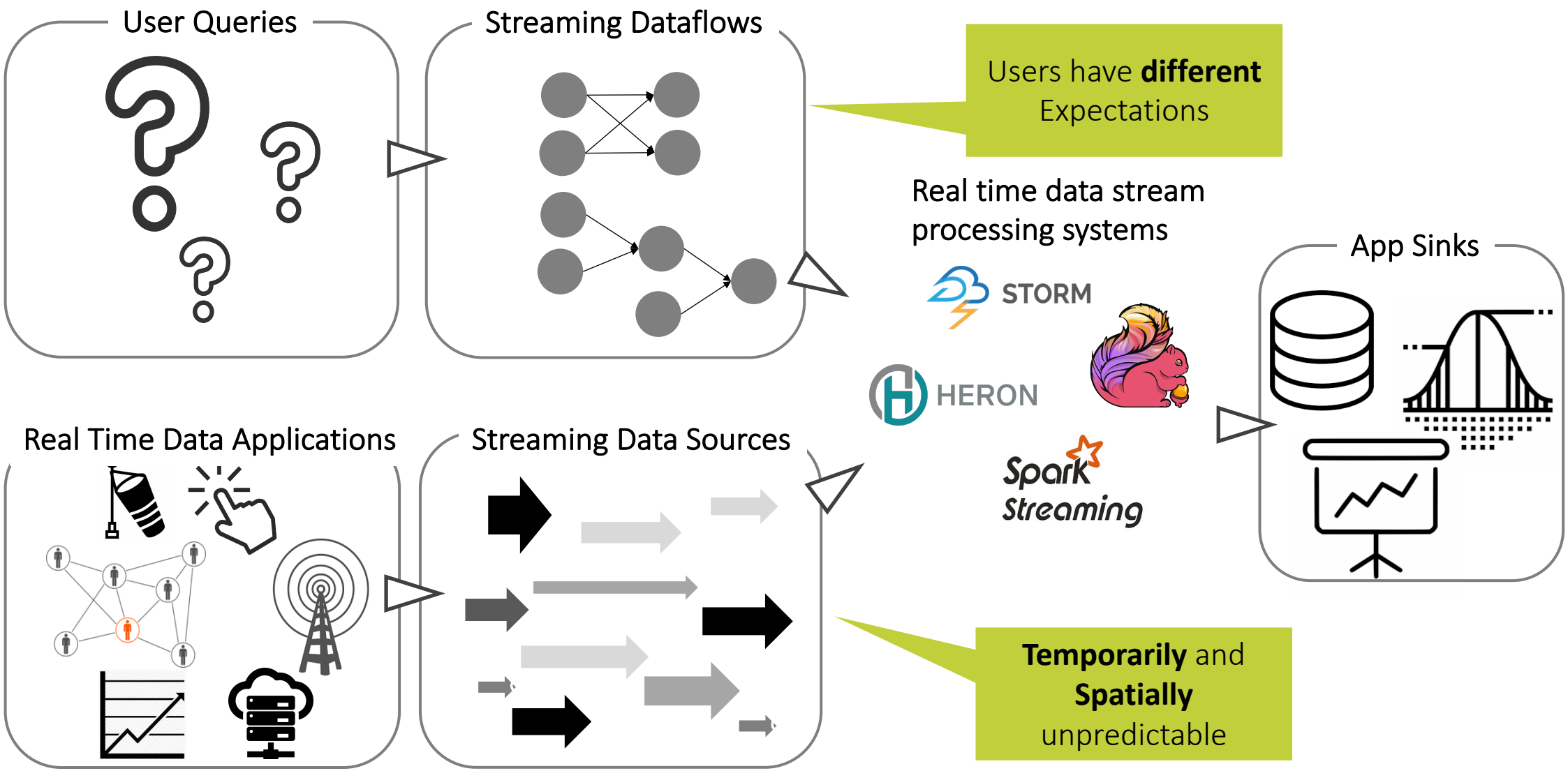
Streaming Data Sources

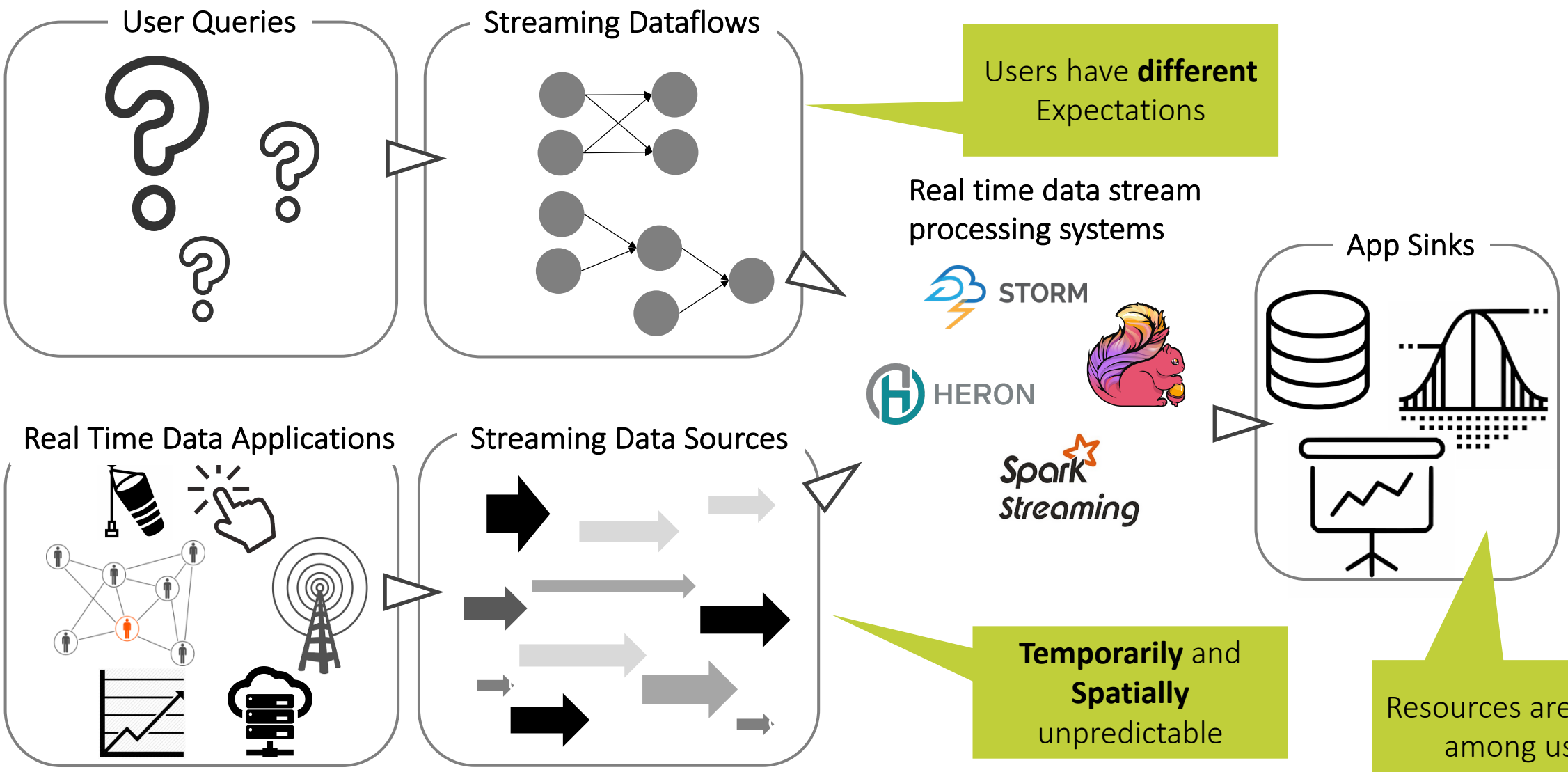


Temporarily and **Spatially** unpredictable


User Queries
Streaming Dataflows
Real time data stream processing systems
STORM
HERON
Spark Streaming
Real Time Data Applications
Streaming Data Sources
Temporarily and Spatially unpredictable


User Queries
Streaming Dataflows
Users have **different** Expectations
Real time data stream processing systems
STORM
HERON
Spark Streaming
Real Time Data Applications
Streaming Data Sources
Temporarily and **Spatially** unpredictable


User Queries
Streaming Dataflows
Users have **different** Expectations
Real time data stream processing systems
STORM
HERON
Spark Streaming
App Sinks
Real Time Data Applications
Streaming Data Sources
Temporarily and **Spatially** unpredictable


User Queries
Streaming Dataflows
Users have **different** Expectations
Real time data stream processing systems
STORM
HERON
Spark Streaming
App Sinks
Real Time Data Applications
Streaming Data Sources
Temporarily and **Spatially** unpredictable
Resources are **shared** among users

User Queries
Streaming Dataflows
Various performance expectations
STORM
HERON
Spark Streaming
App Sinks
Real Time Data Applications
Streaming Data Sources
Temporarily and spatially unpredictable
Users have different Expectations

User Queries

Streaming Dataflows

Various performance expectations

App Sinks

.filter(e.key == “Error”)

Target latency 50

Real Time Data Applications

Streaming Data Sources

STORM

HERON

Spark Streaming

Latency Sensitive Query

Temporarily and spatially unpredictable

Users have different Expectations

User Queries

Streaming Dataflows

Various performance expectations

App Sinks

Real Time Data Applications

Streaming Data Sources

.filter(e.key == “Error”)

Target latency 50

.window(1000)
.groupBy(e.key)
.sum()

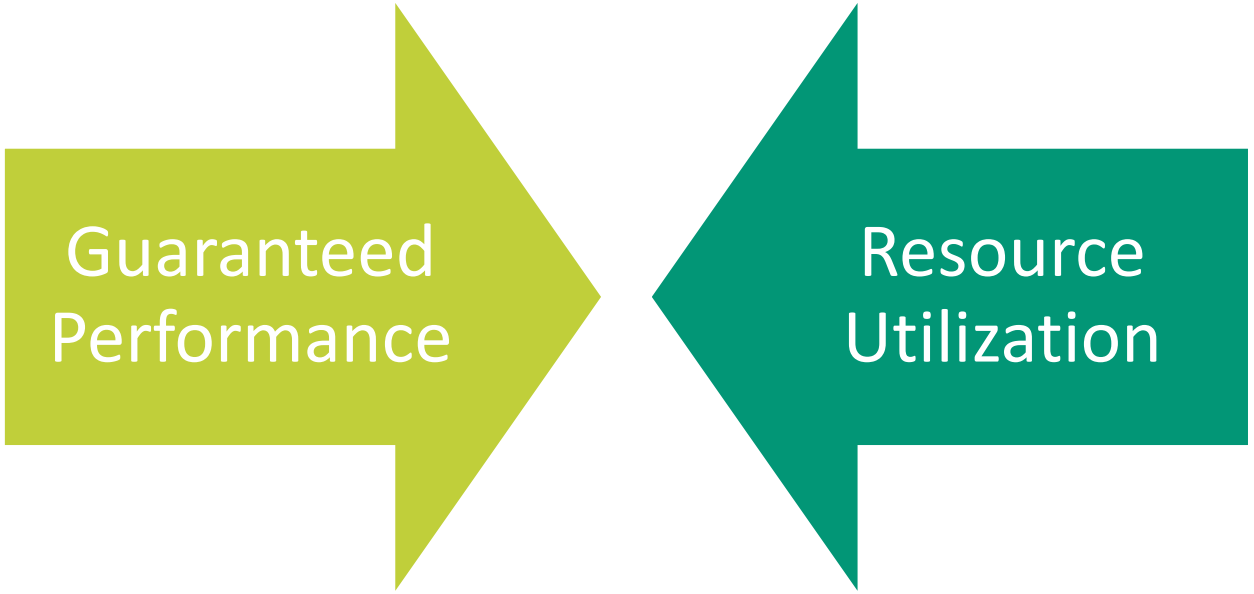
Target latency 100

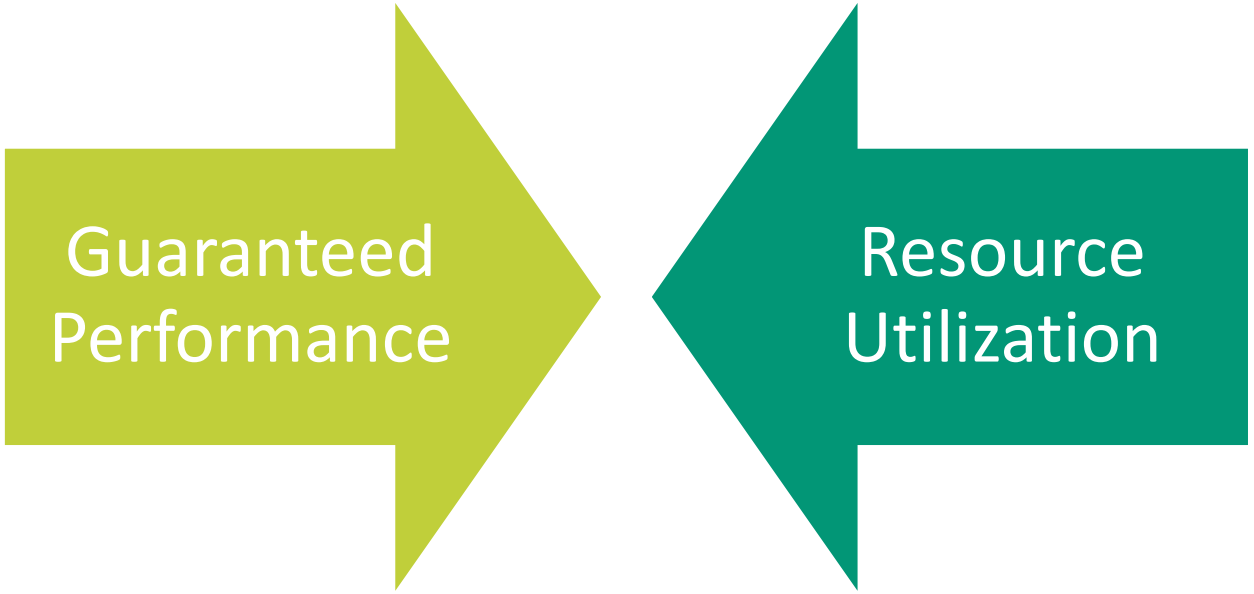
Latency Sensitive Query

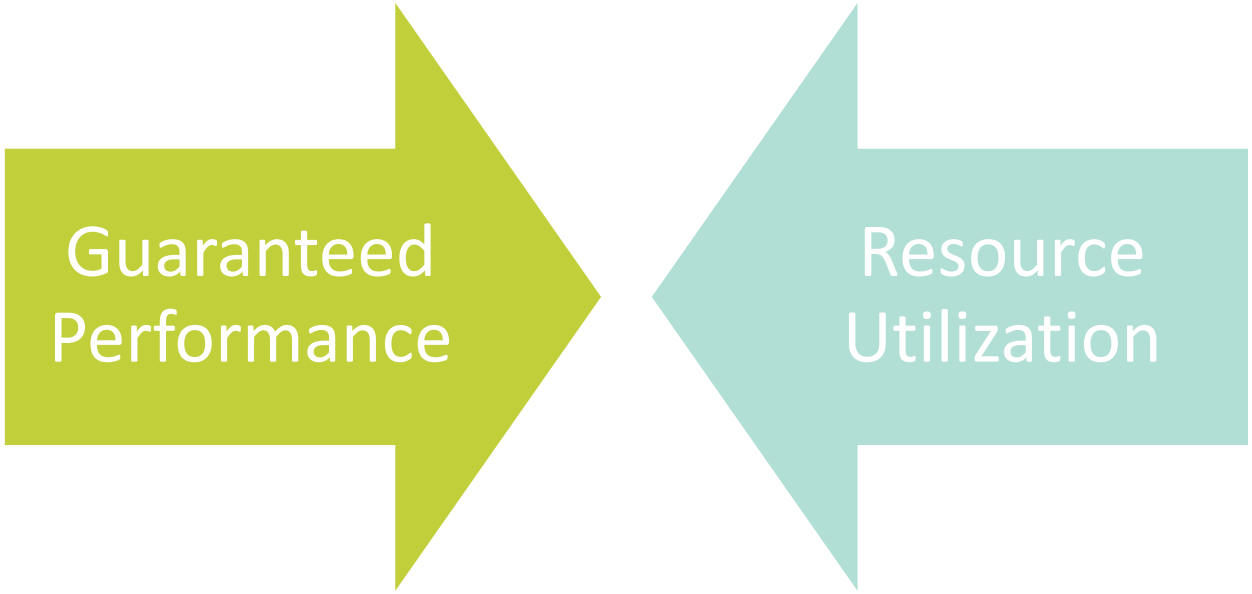
Bulk Analytics Query

Temporarily and spatially unpredictable

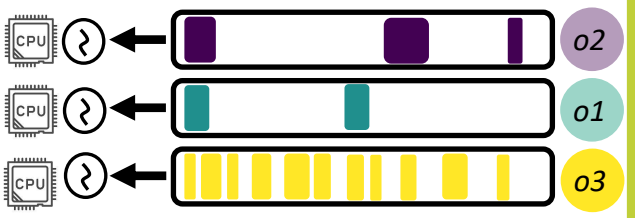
Users have different Expectations


Guaranteed
Performance
Resource
Utilization


Guaranteed
Performance
Resource
Utilization


Guaranteed Performance
Resource Utilization

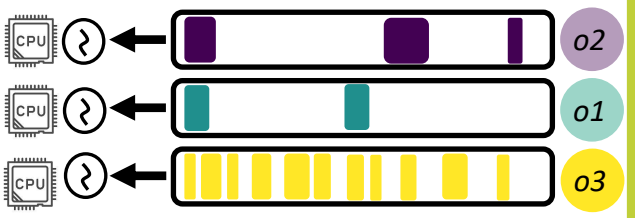
Full Resource Isolation



Guaranteed Performance

Resource Utilization

Full Resource Isolation



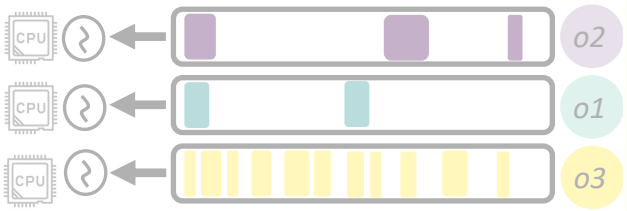
Guaranteed Performance

Resource Utilization

+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

Full Resource Isolation



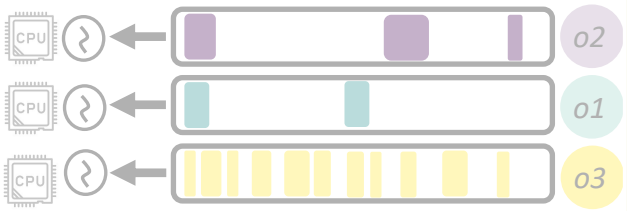
Guaranteed Performance

Resource Utilization

+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

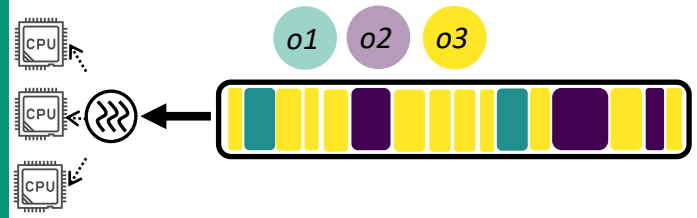
Full Resource Isolation



Guaranteed Performance

Resource Utilization

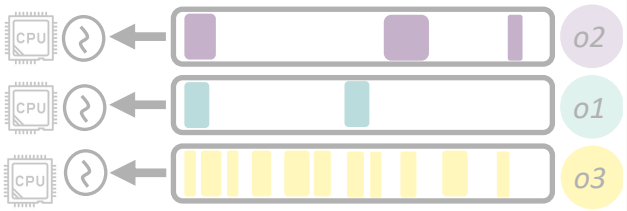
No Resource Isolation



+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

Full Resource Isolation



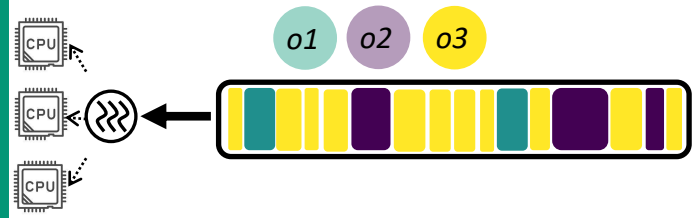
Guaranteed Performance

+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

Resource Utilization

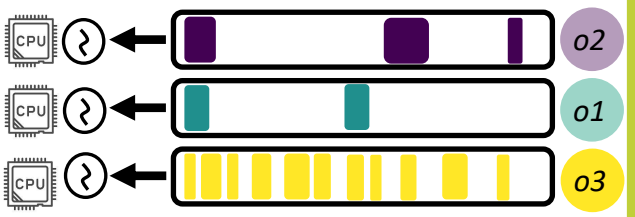
No Resource Isolation



+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)

Full Resource Isolation



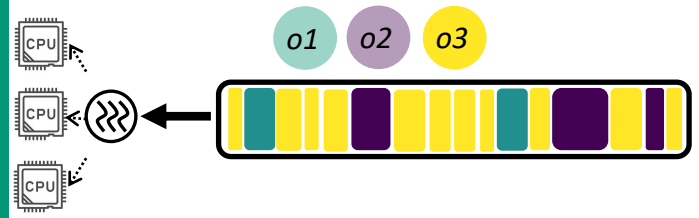
Guaranteed Performance

+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

Resource Utilization

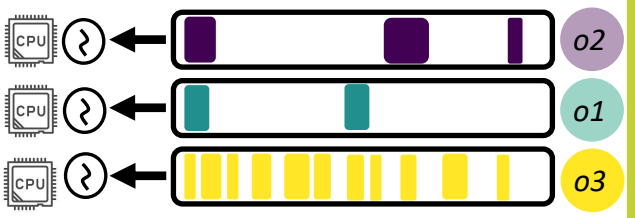
No Resource Isolation



+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)

Full Resource Isolation

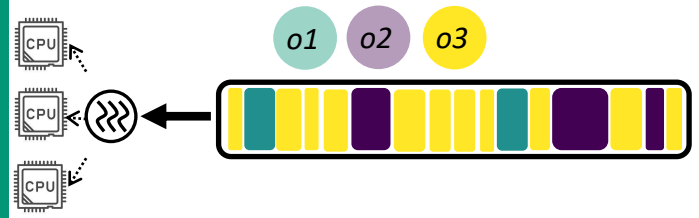


Guaranteed Performance

?

Resource Utilization

No Resource Isolation



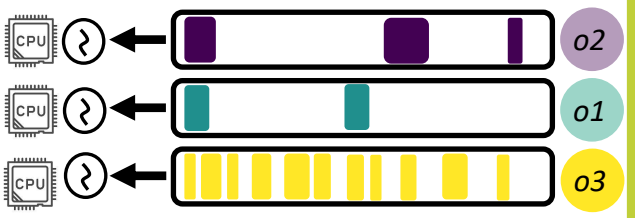
+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)

Full Resource Isolation



Guaranteed Performance

+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

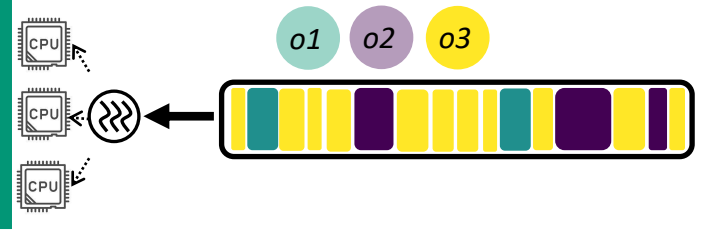
Fine-grained Resource Sharing

Resource Utilization

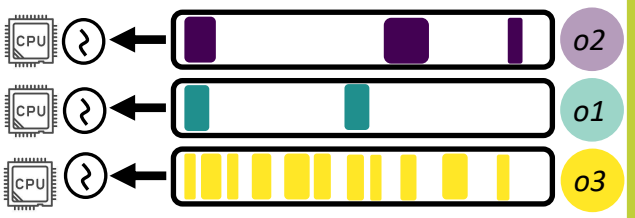
+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)

No Resource Isolation



Full Resource Isolation



Guaranteed Performance

Performance Isolation

+ Unaffected performance under workload peaks

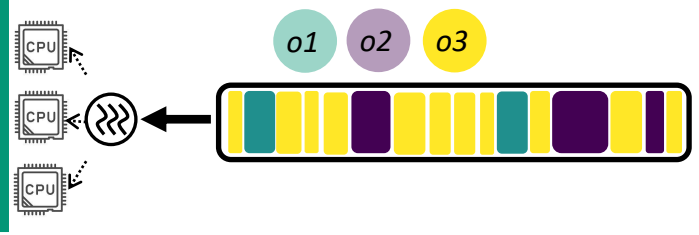
- Resource under-utilization and high cost

?

Fine-grained Resource Sharing

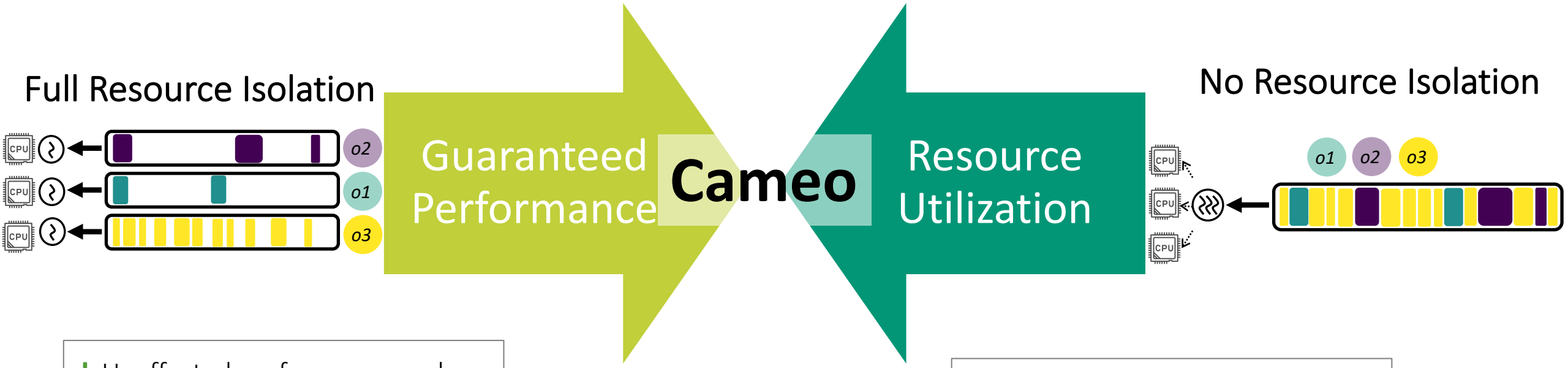
Resource Utilization

No Resource Isolation



+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)



+ Unaffected performance under workload peaks

- Resource under-utilization and high cost

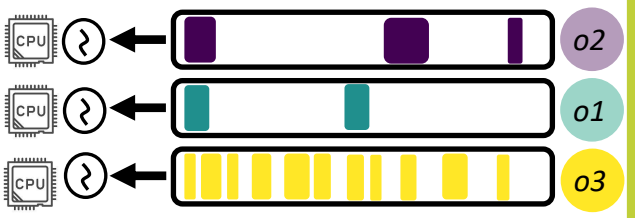
Performance Isolation

Fine-grained Resource Sharing

+ Uses resource efficiently

- Compromised performance (e.g., high processing latency)

Full Resource Isolation



Guaranteed Performance

Cameo

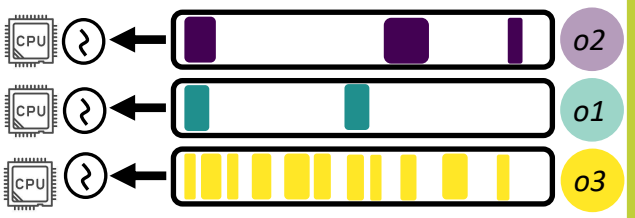
Resource Utilization

No Resource Isolation

Event-driven Architecture

Treating operator invocation as serverless functions

Full Resource Isolation



Guaranteed Performance

Cameo

Resource Utilization

No Resource Isolation

Event-driven Architecture

Treating operator invocation as serverless functions

Orleans

Guaranteed Performance

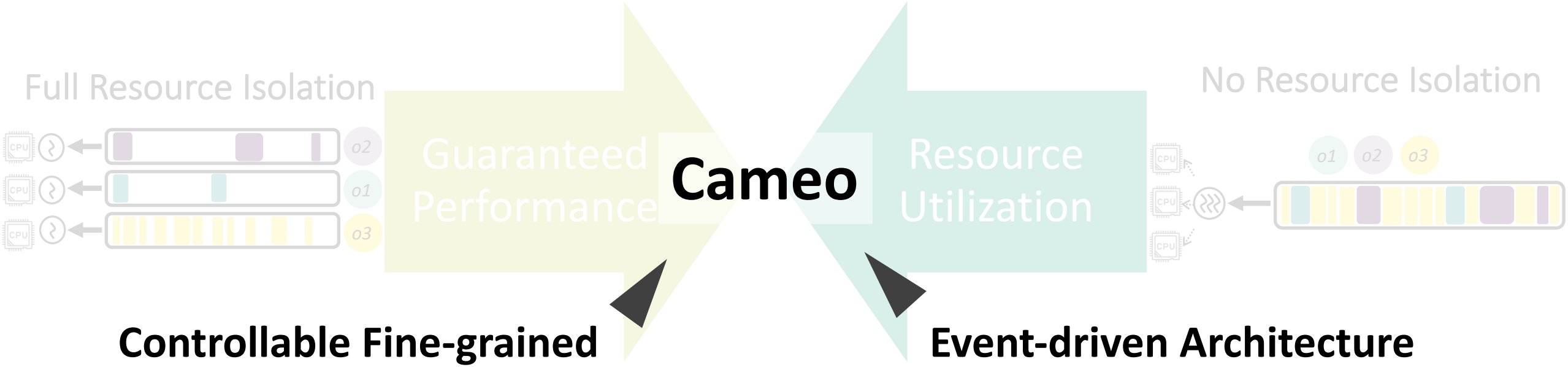
Cameo

Resource Utilization

Event-driven Architecture

Treating operator invocation as serverless functions

Orleans



Controllable Fine-grained Scheduling

Identifying and prioritizing important work

Event-driven Architecture

Treating operator invocation as serverless functions

Orleans

Cameo

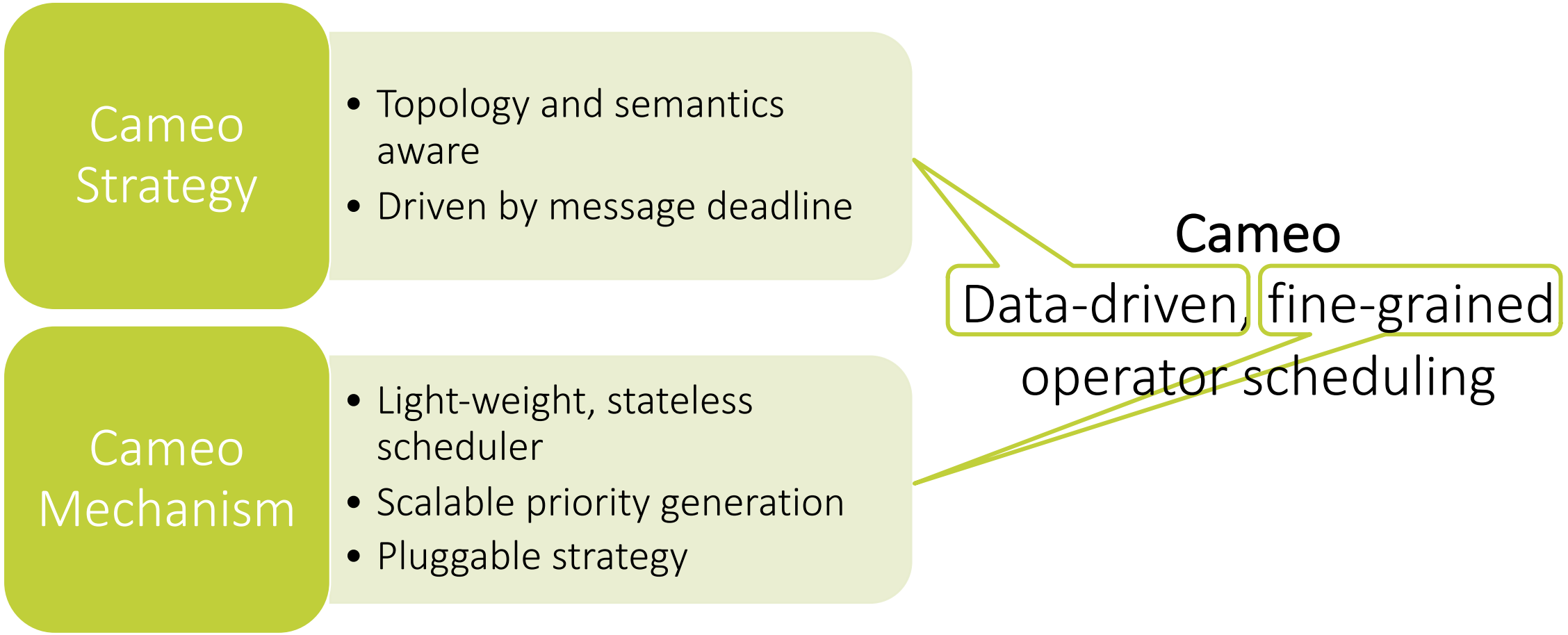
Data-driven, fine-grained operator scheduling

Cameo Strategy

- Topology and semantics aware
- Driven by message deadline

Cameo

Data-driven, fine-grained operator scheduling


Cameo Strategy
• Topology and semantics aware
• Driven by message deadline
Cameo Mechanism
• Light-weight, stateless scheduler
• Scalable priority generation
• Pluggable strategy
Cameo
Data-driven, fine-grained operator scheduling

Cameo Strategy

- Topology and semantics aware
- Driven by message deadline

Cameo Mechanism

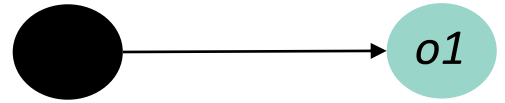
- Light-weight, stateless scheduler
- Scalable priority generation
- Pluggable strategy

Cameo

Data-driven, fine-grained operator scheduling

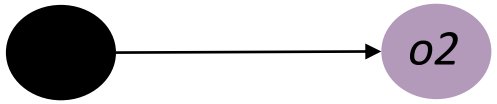
Cameo Strategy

```
.window(1000)
.groupBy(e.key)
.sum()
```



Bulk Analytics Query

```
.filter(e.key == “Error”)
```

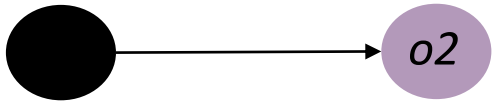


Latency Sensitive Query

```
.window(1000)
.groupBy(e.key)
.sum()
```

Bulk Analytics Query

```
.filter(e.key == “Error”)
```



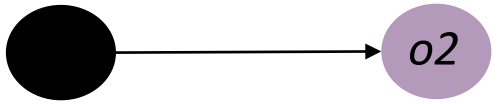
Latency Sensitive Query

.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)



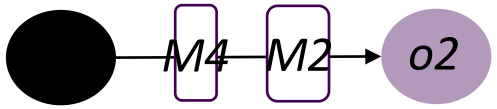
Latency Sensitive Query

.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)



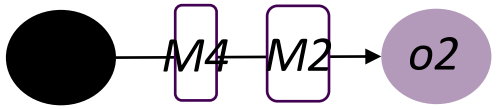
Latency Sensitive Query

.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

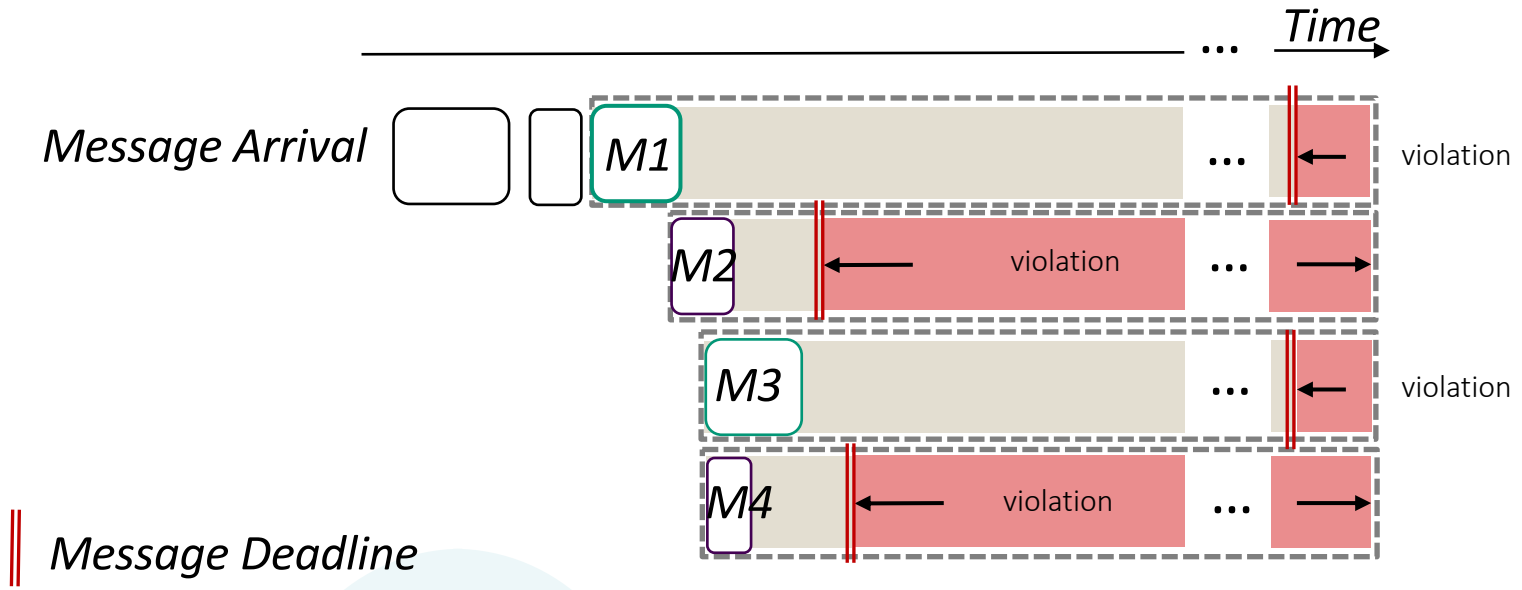
Bulk Analytics Query

.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

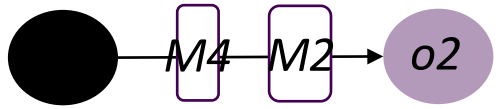


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

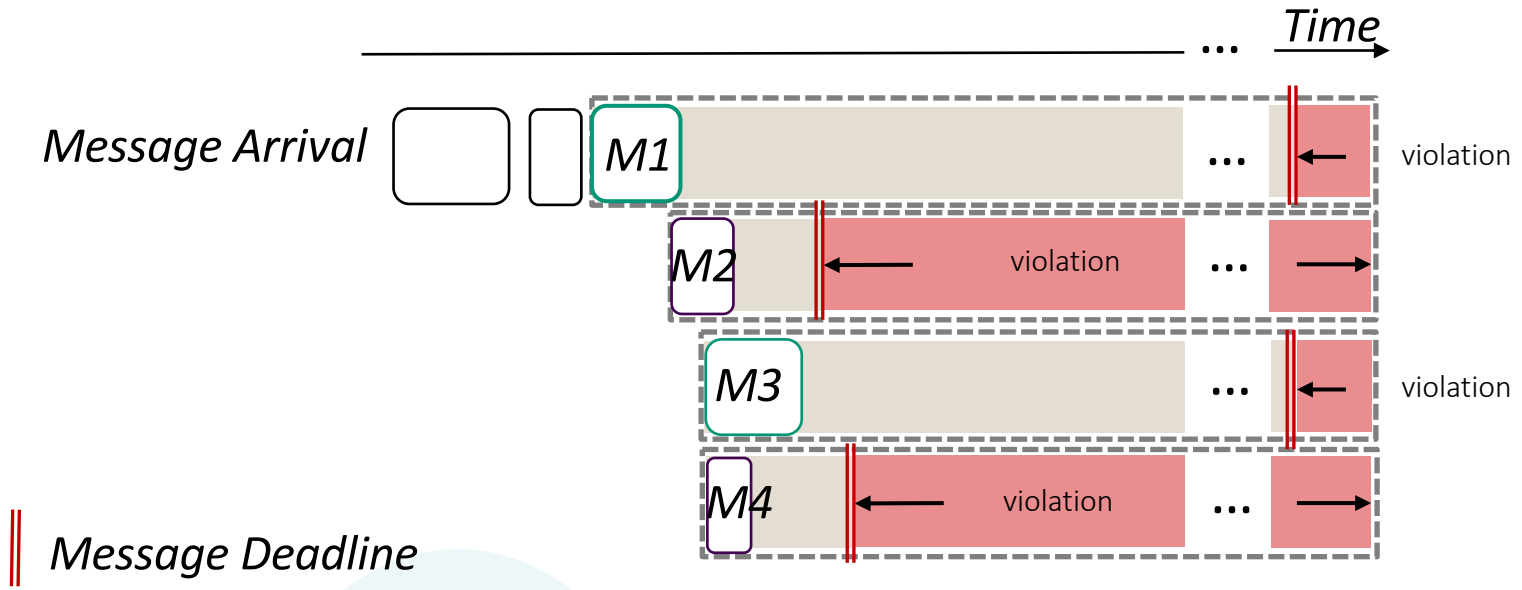
Bulk Analytics Query

.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

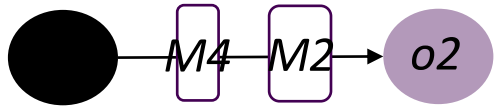


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

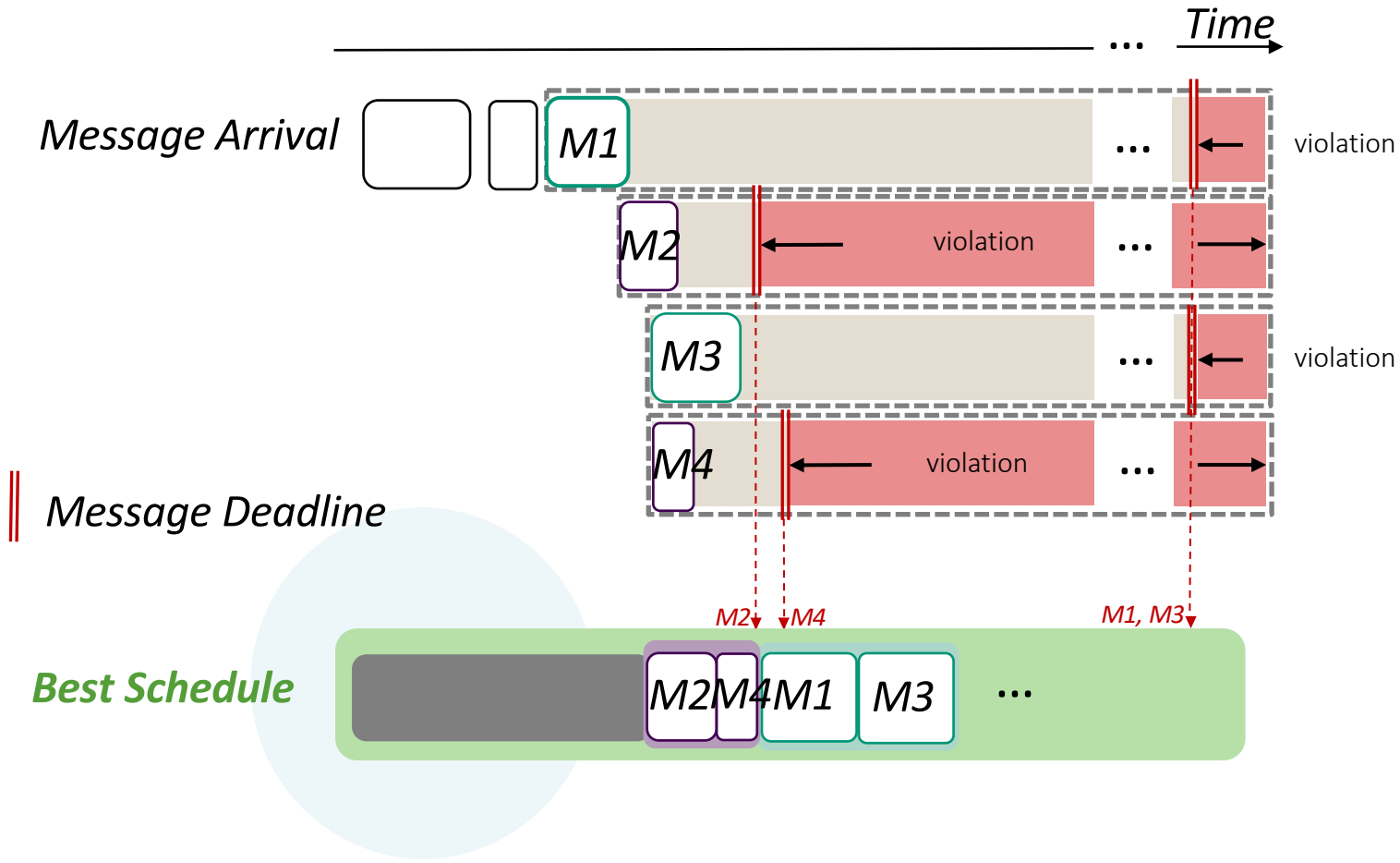
Bulk Analytics Query

.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

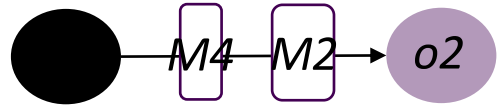


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

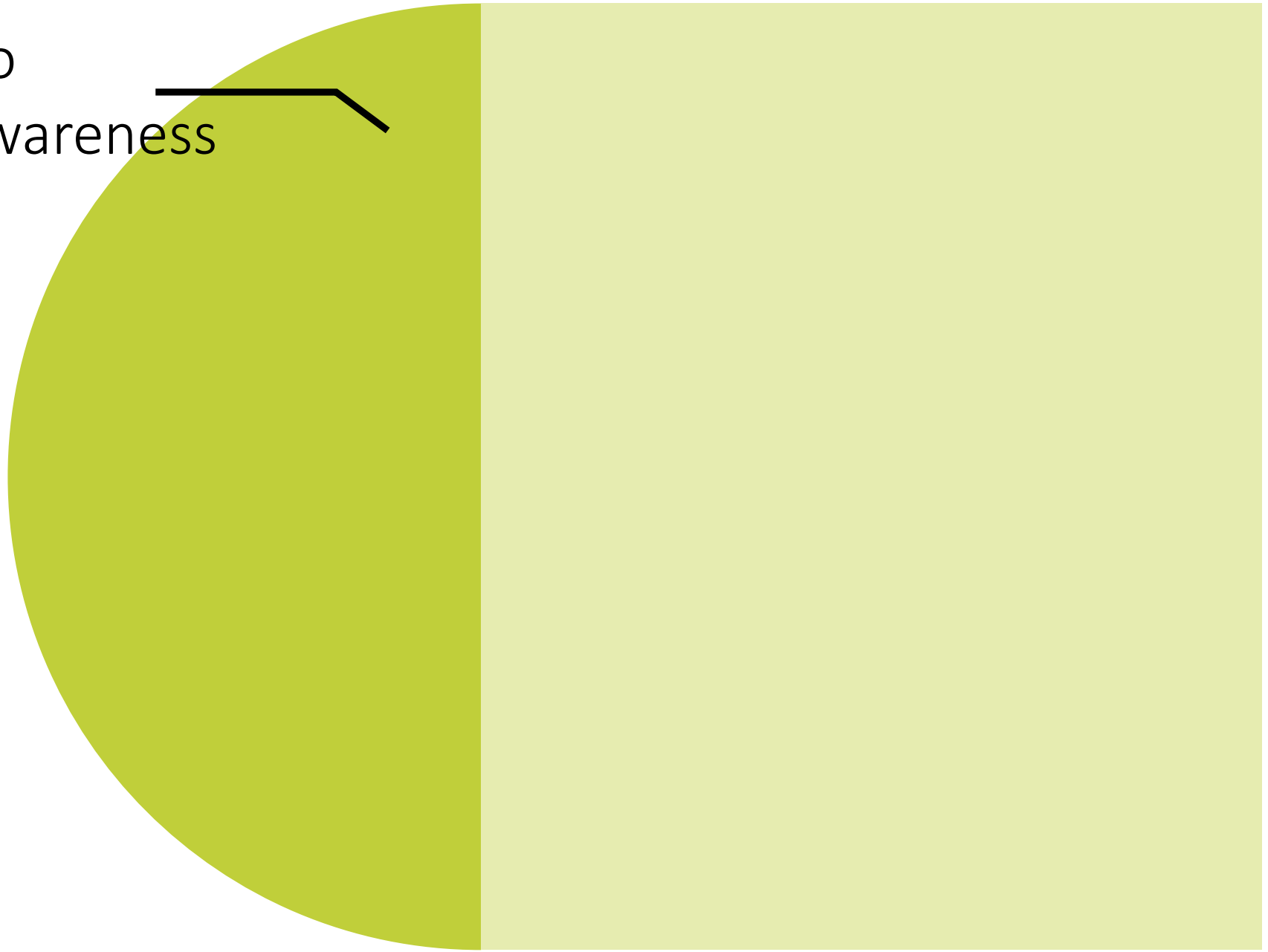
.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

No Awareness



.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

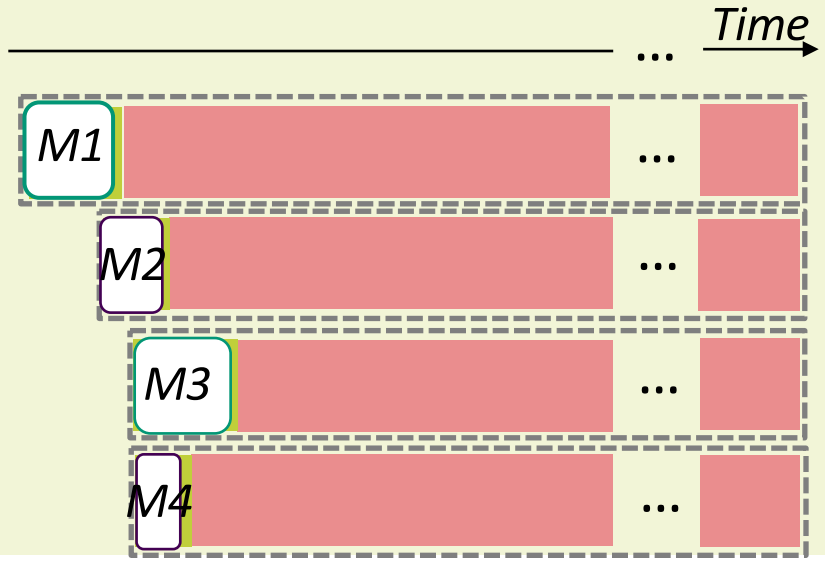
.filter(e.key == “Error”)

Target latency **50**

Latency Sensitive Query

violation

No Awareness



.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

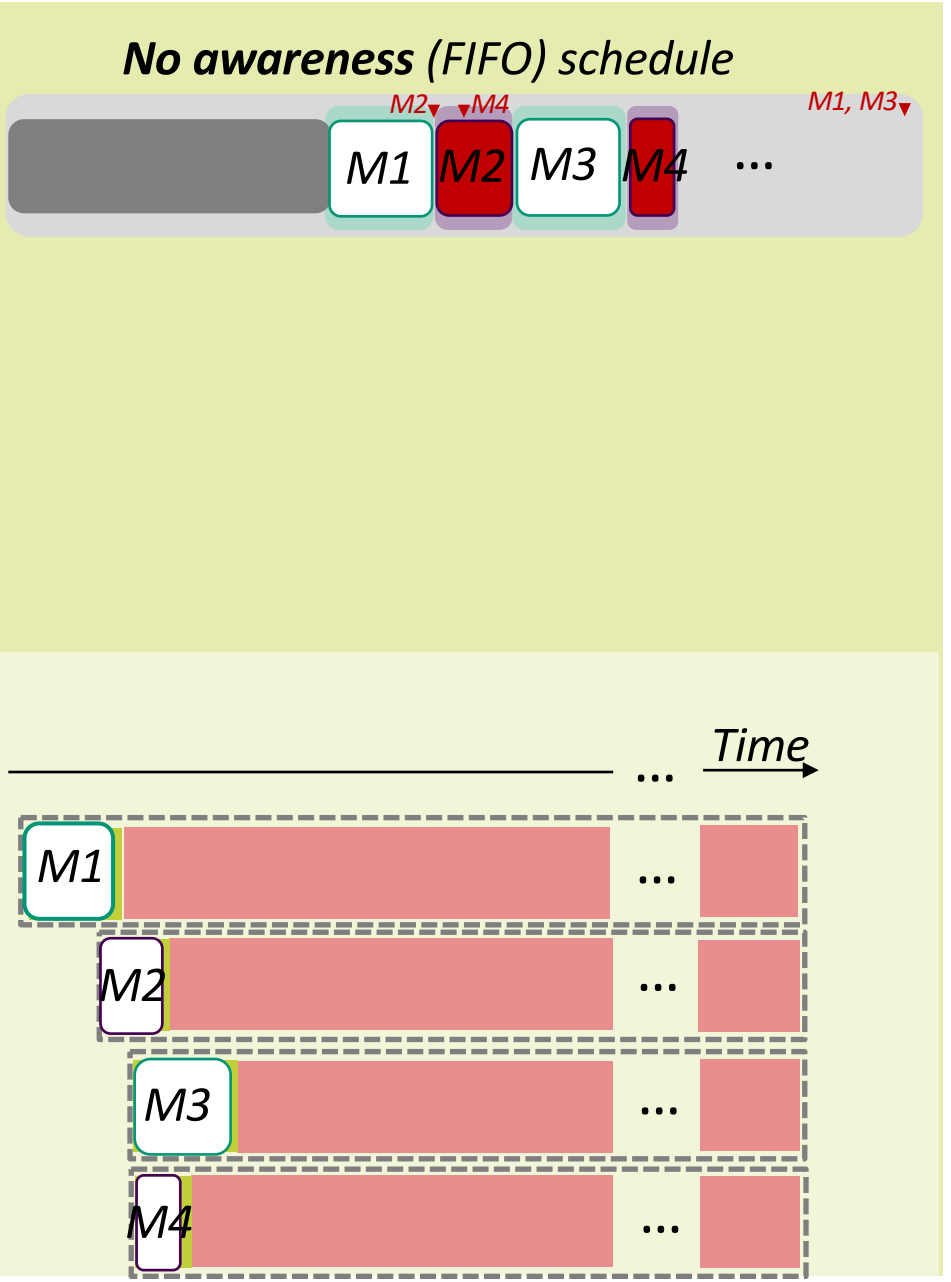
.filter(e.key == “Error”)

Target latency **50**

Latency Sensitive Query

violation

No Awareness



.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)

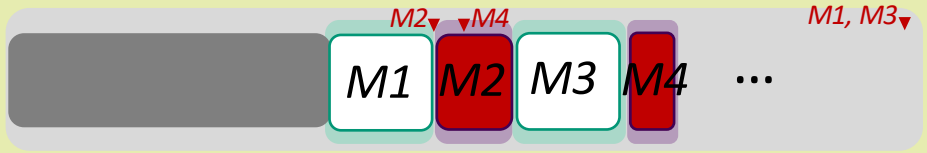
Target latency **50**

Latency Sensitive Query

violation

No Awareness

Topology-aware



.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)

Target latency **50**

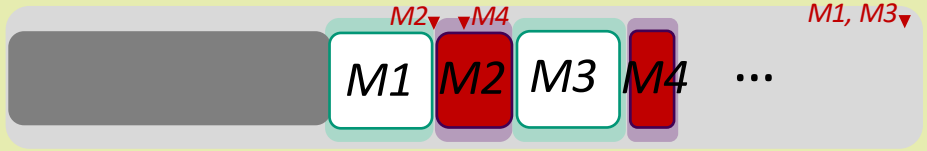
Latency Sensitive Query

violation

No Awareness

Topology-aware

***No awareness** (FIFO) schedule*

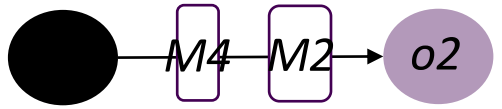


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)



Target latency **50**

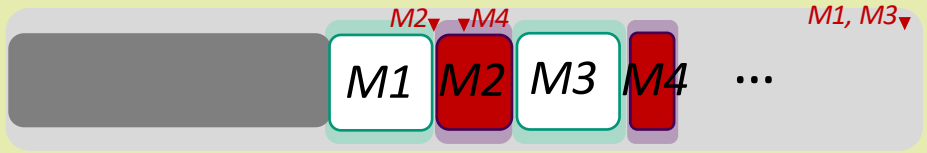
Latency Sensitive Query

violation

No Awareness

Topology-aware

***No awareness** (FIFO) schedule*

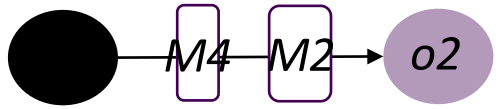


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == “Error”)



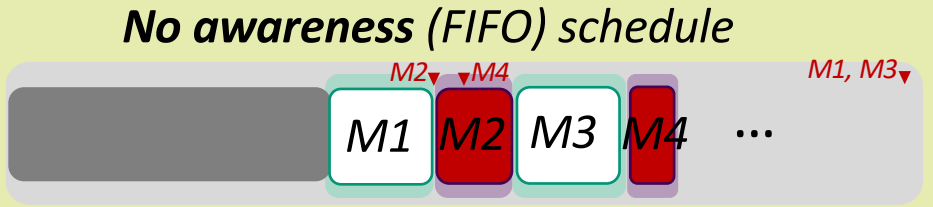
Target latency **50**

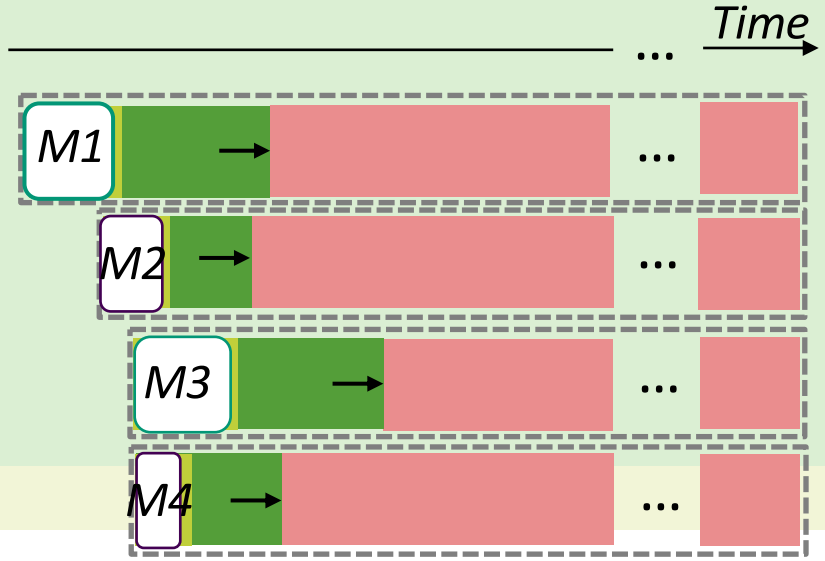
Latency Sensitive Query

violation

No Awareness

Topology-aware



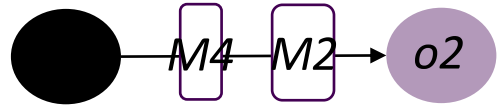


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

.filter(e.key == "Error")



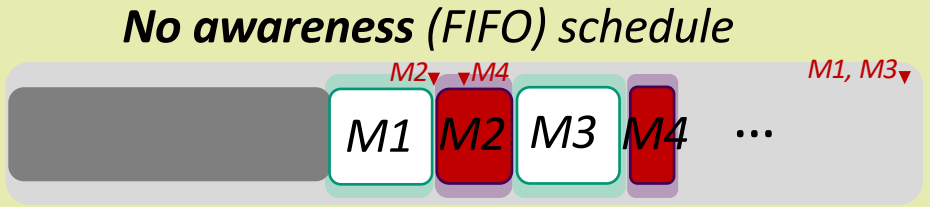
Target latency **50**

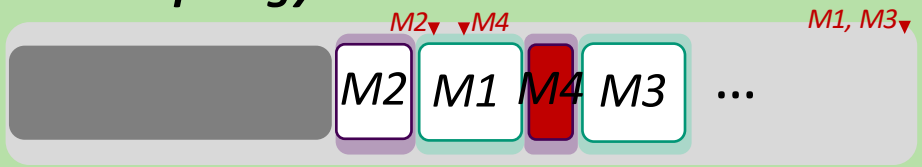
Latency Sensitive Query

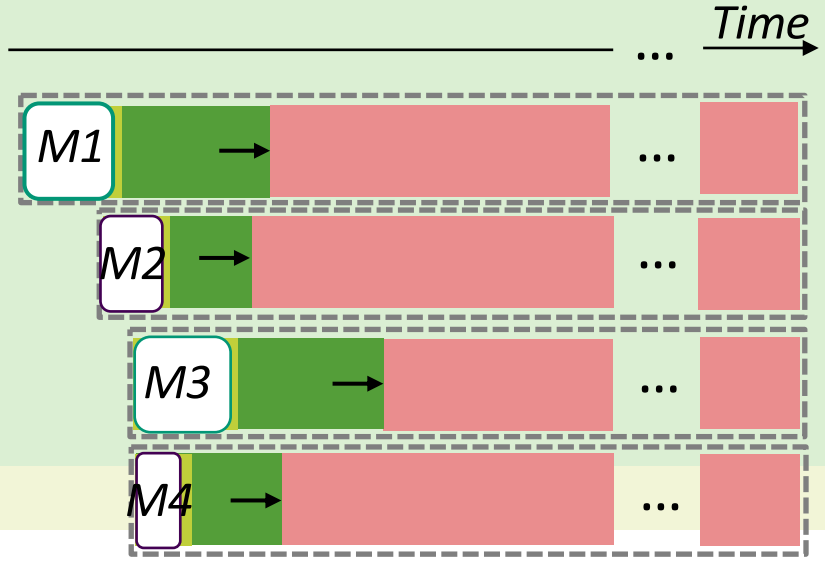
violation

No Awareness

Topology-aware





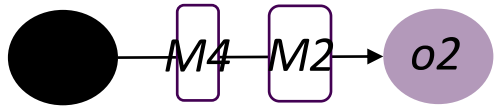



```
.window(1000)
.groupBy(e.key)
.sum()
```

Target latency **100**

Bulk Analytics Query

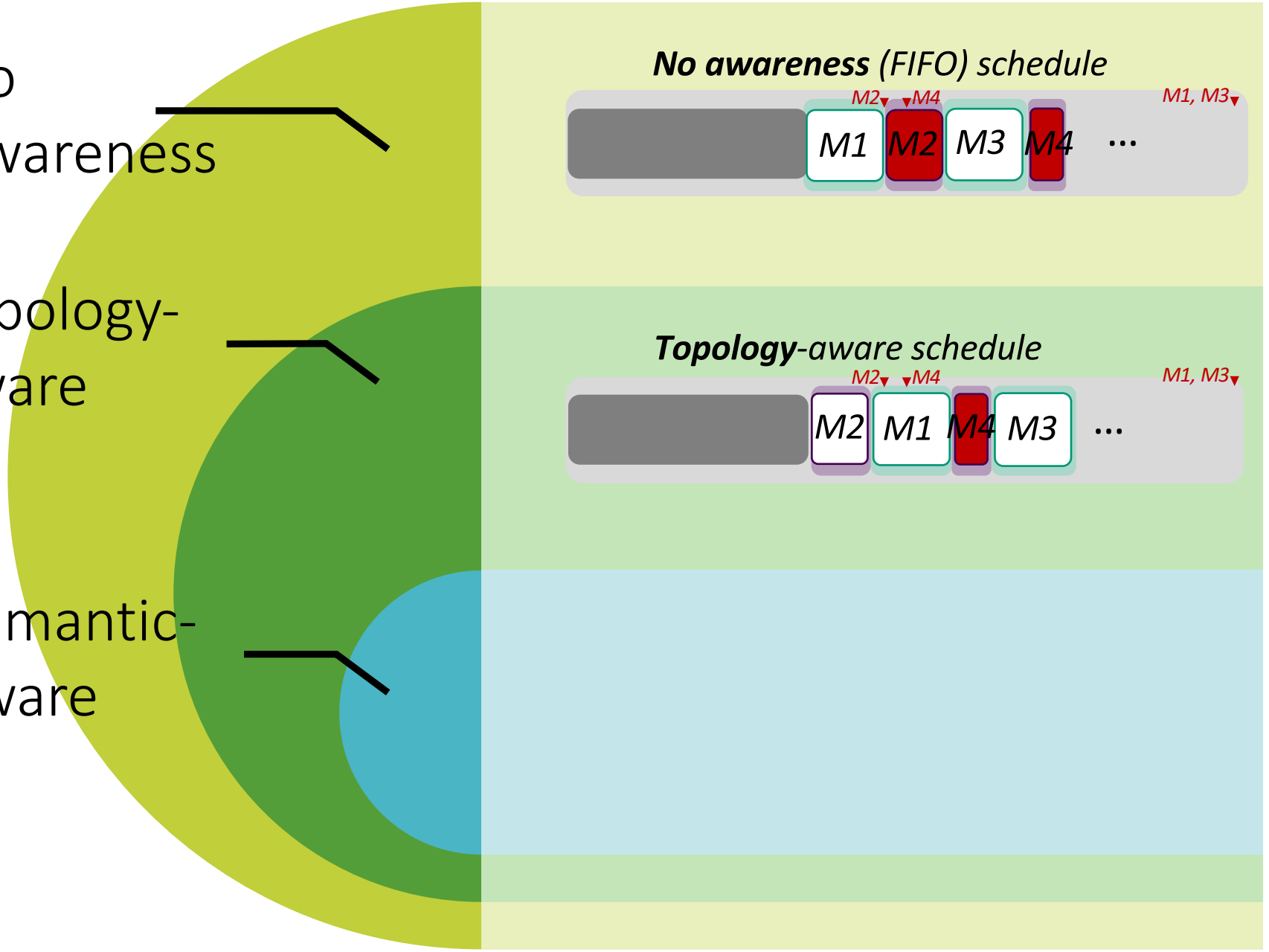
```
.filter(e.key == "Error")
```



Target latency **50**

Latency Sensitive Query

violation

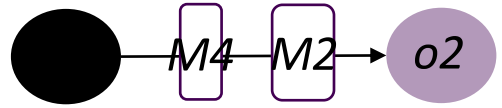


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

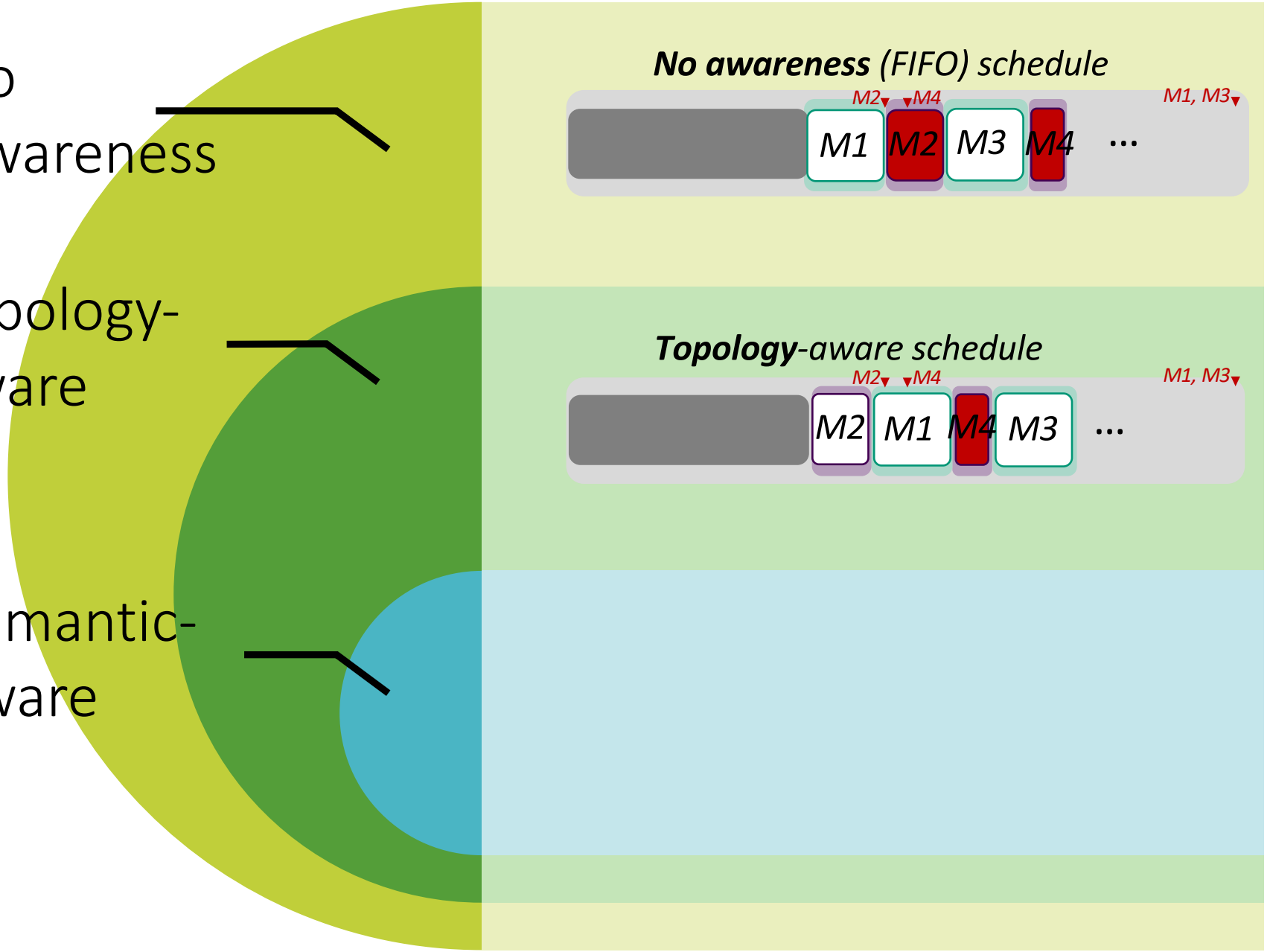
.filter(e.key == "Error")



Target latency **50**

Latency Sensitive Query

violation

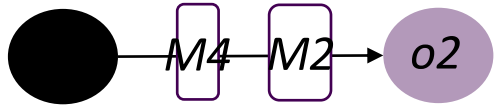


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

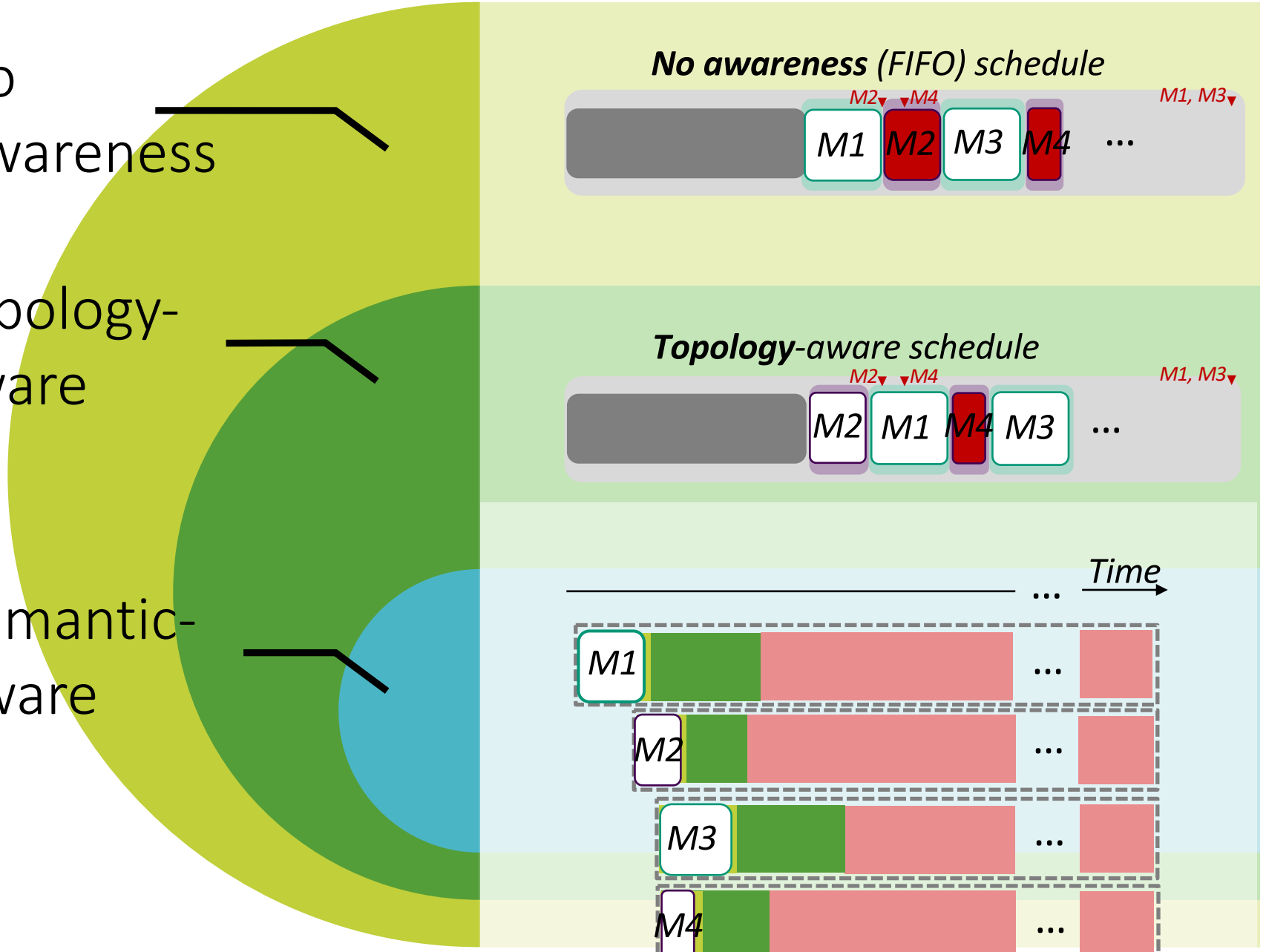
.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

violation

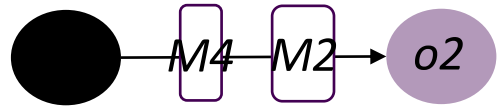


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

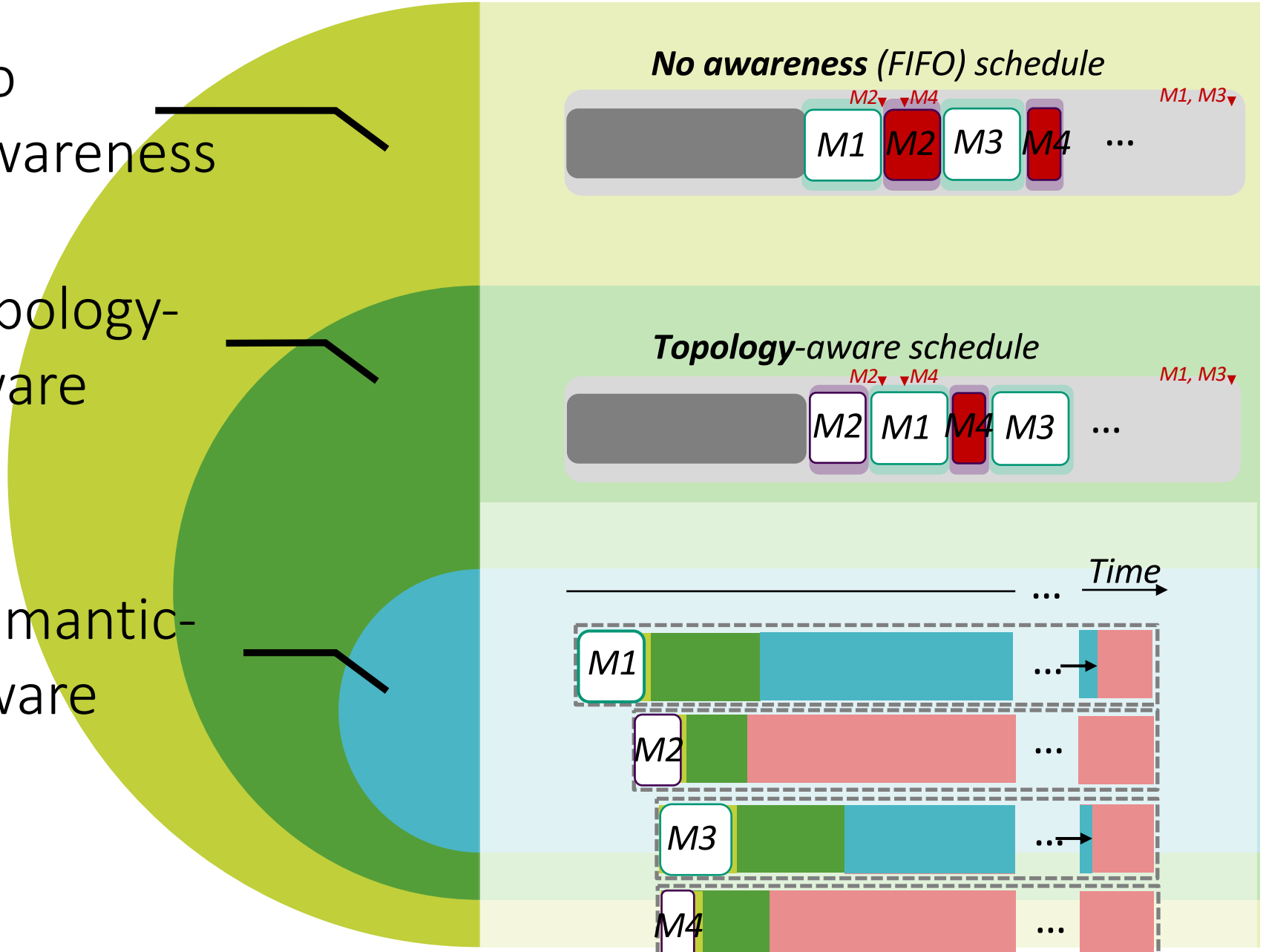
.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

violation

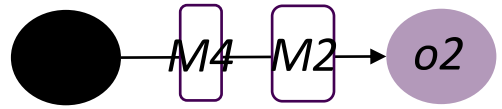


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

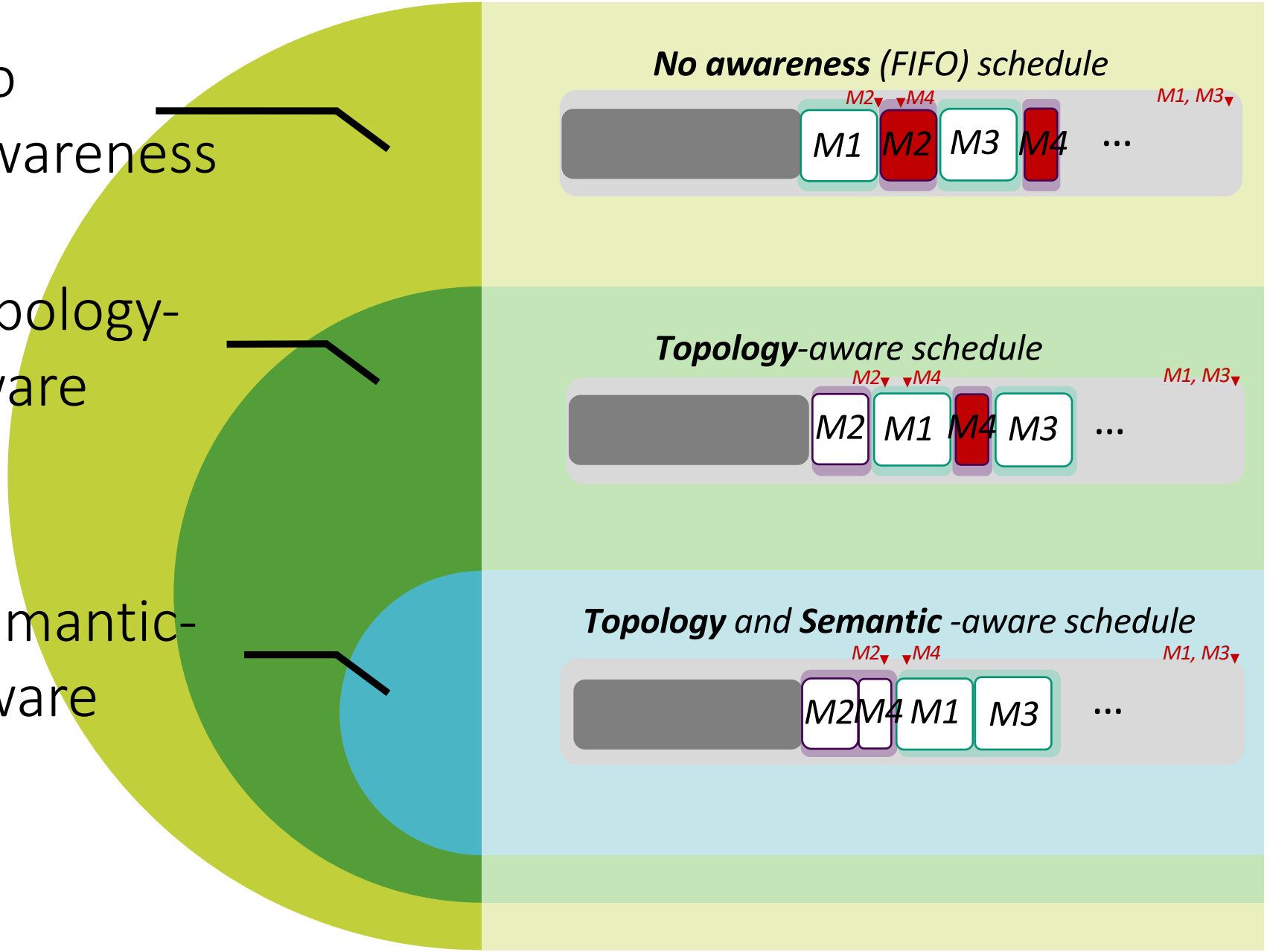
.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

violation

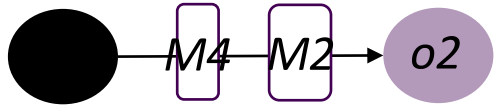


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

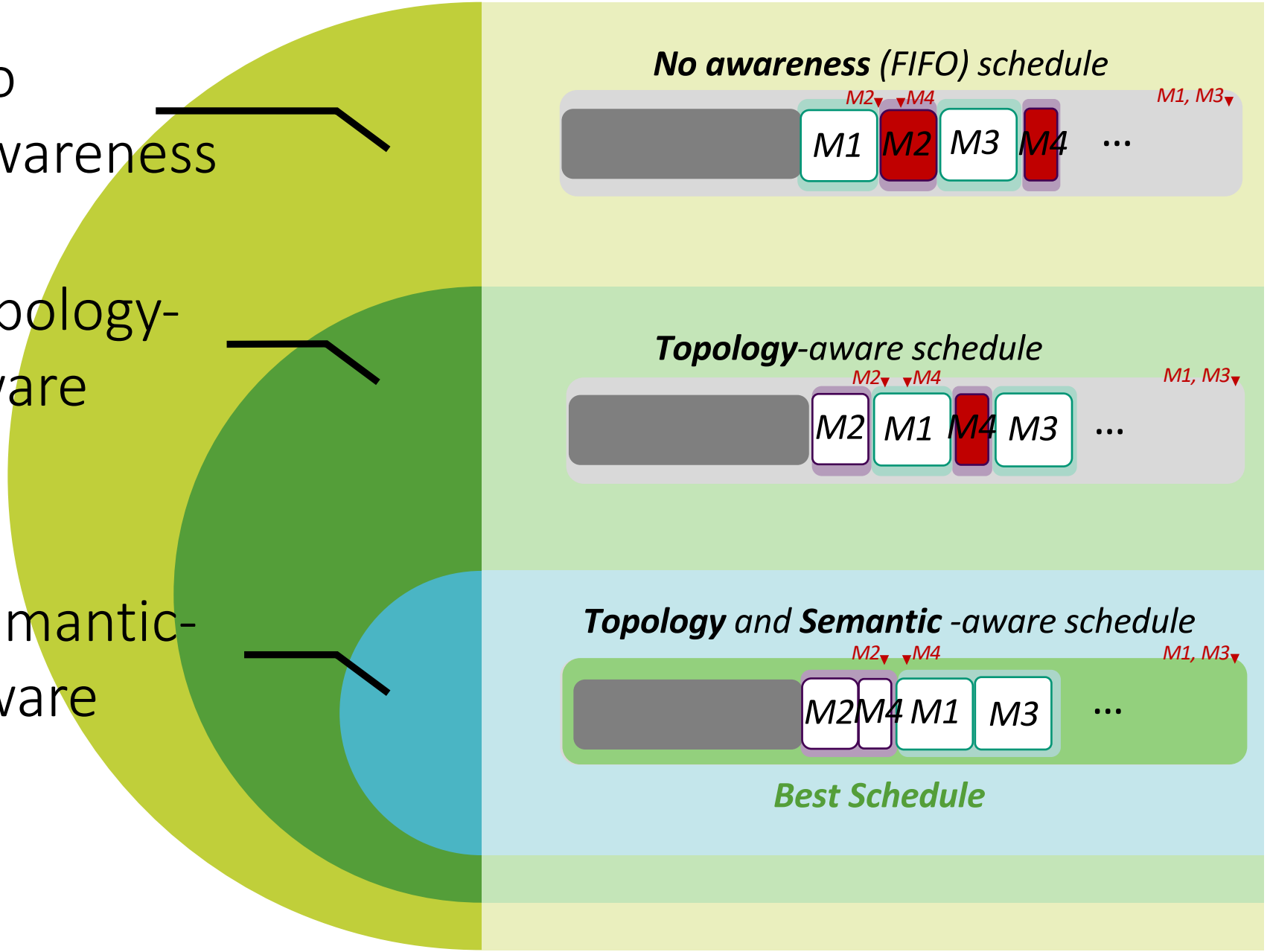
.filter(e.key == “Error”)



Target latency **50**

Latency Sensitive Query

violation

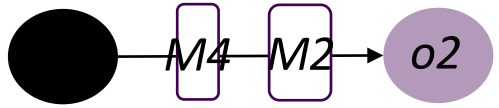


.window(1000)
.groupBy(e.key)
.sum()

Target latency **100**

Bulk Analytics Query

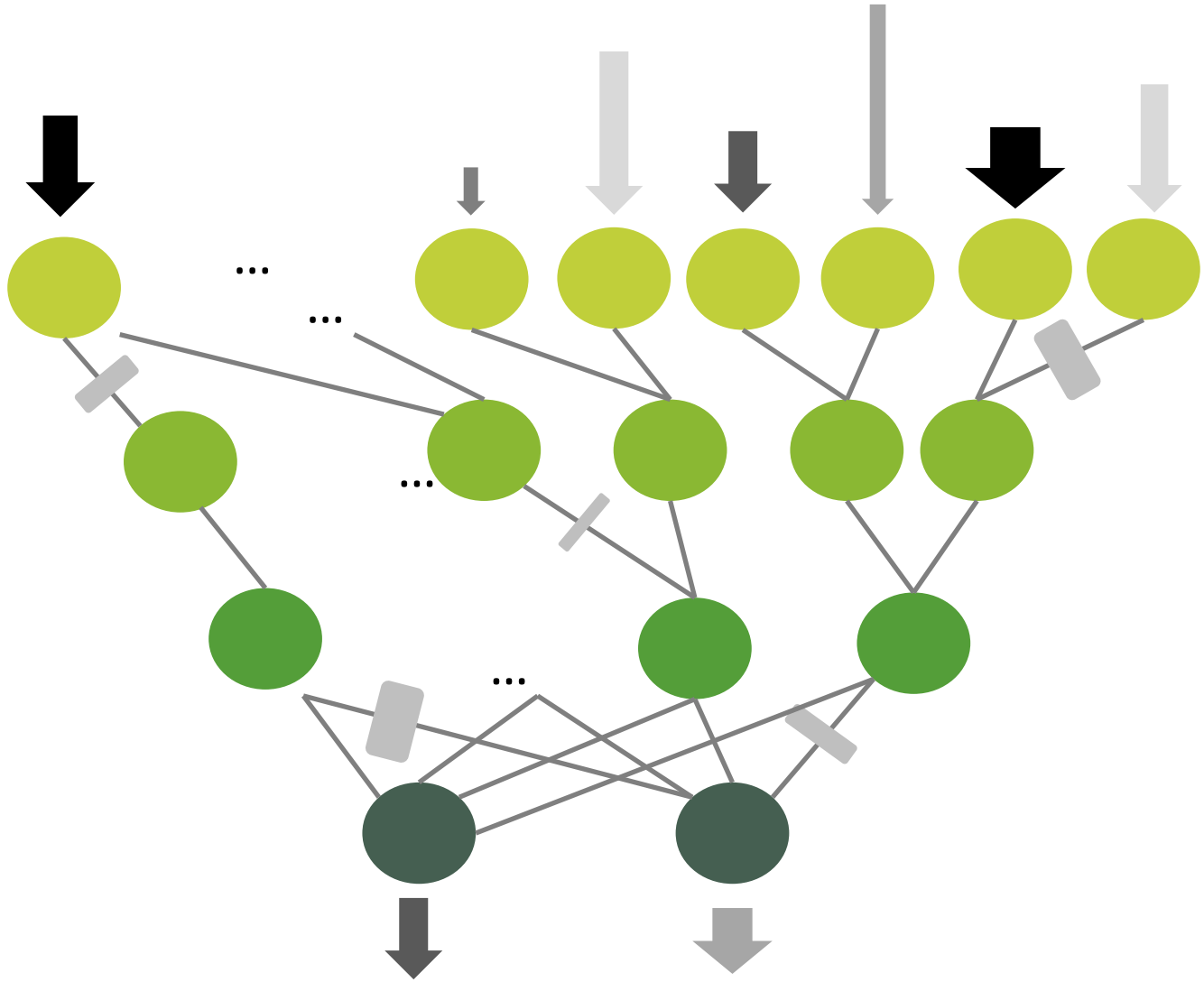
.filter(e.key == “Error”)



Target latency **50**

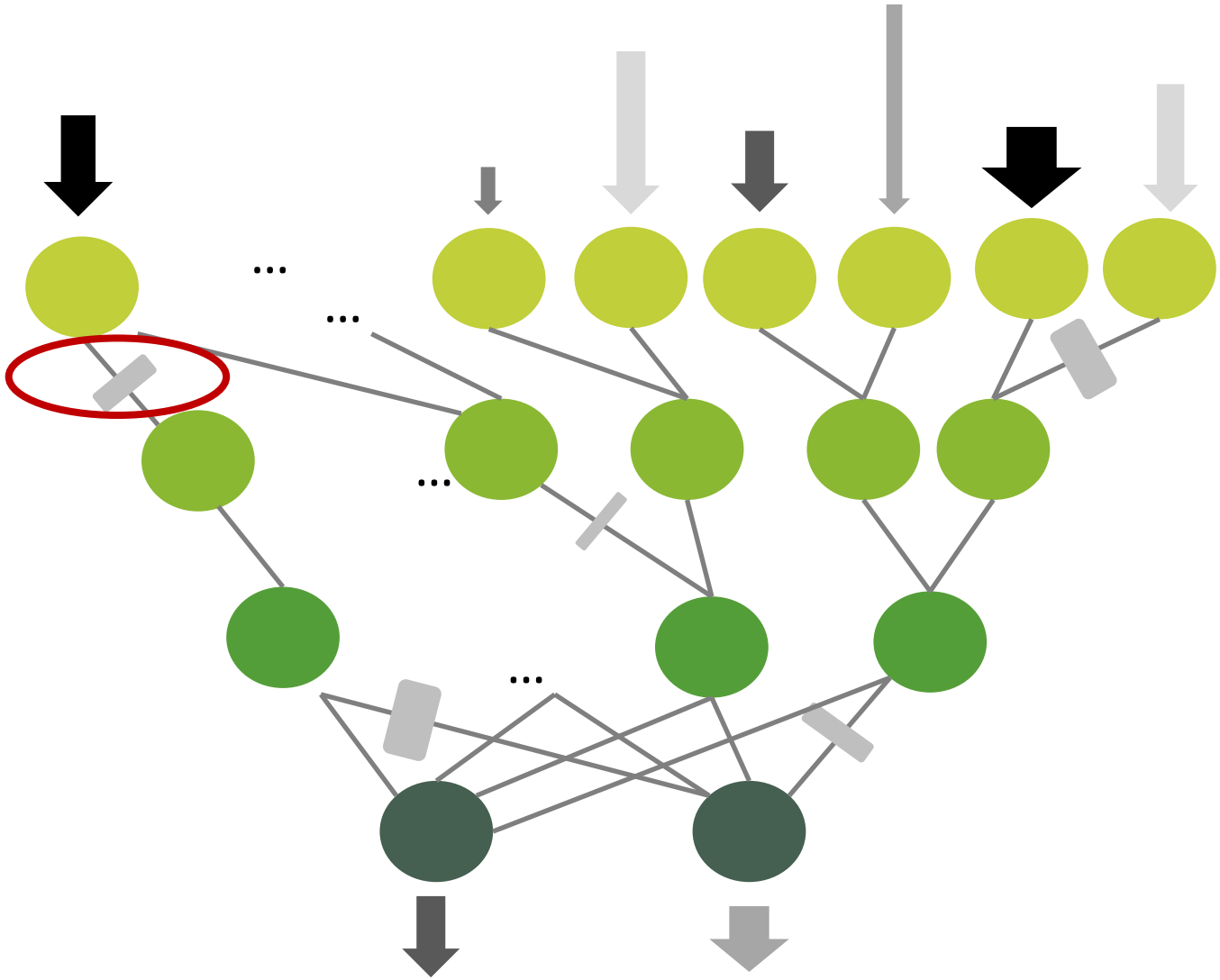
Latency Sensitive Query

violation



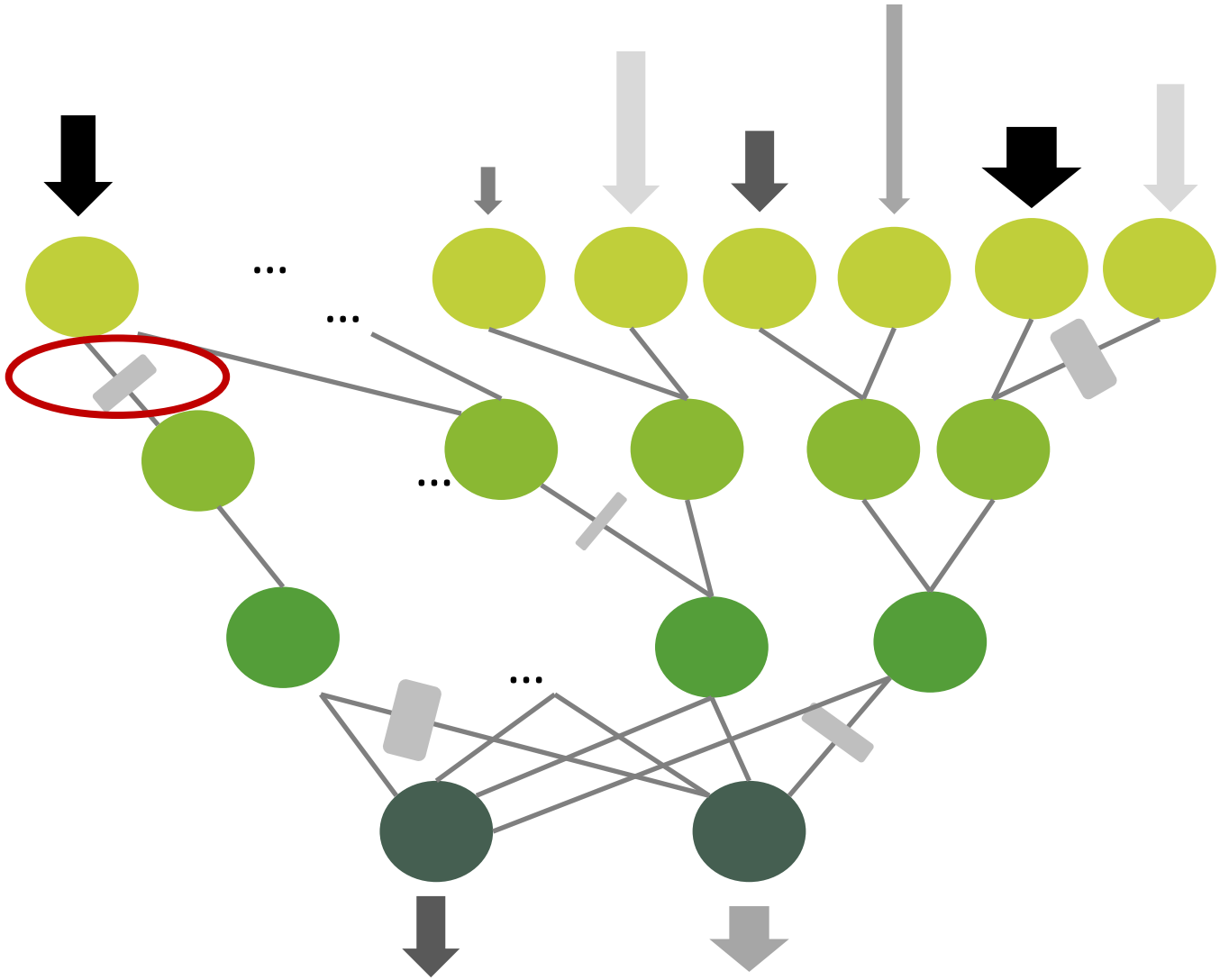
Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



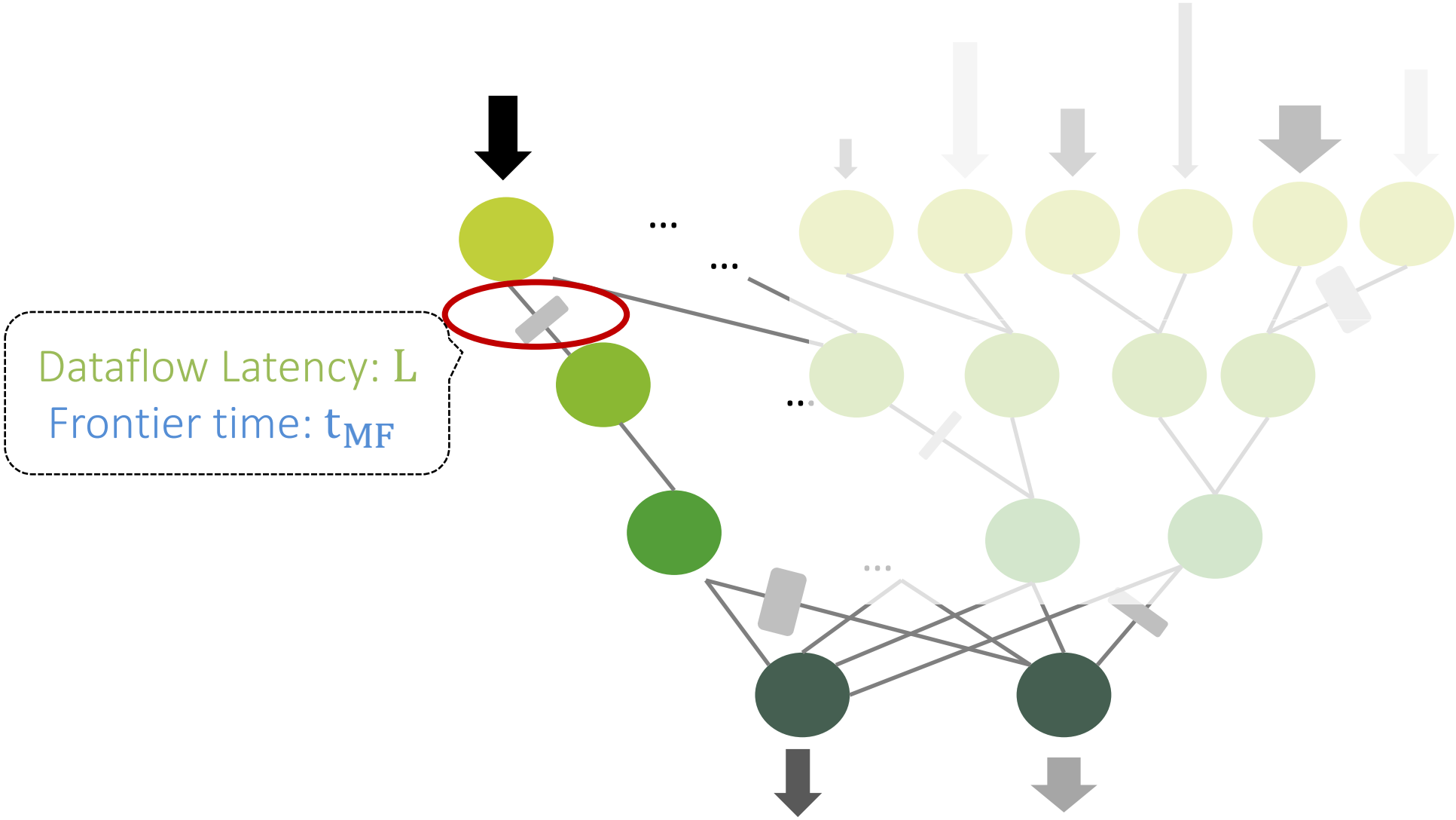
Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



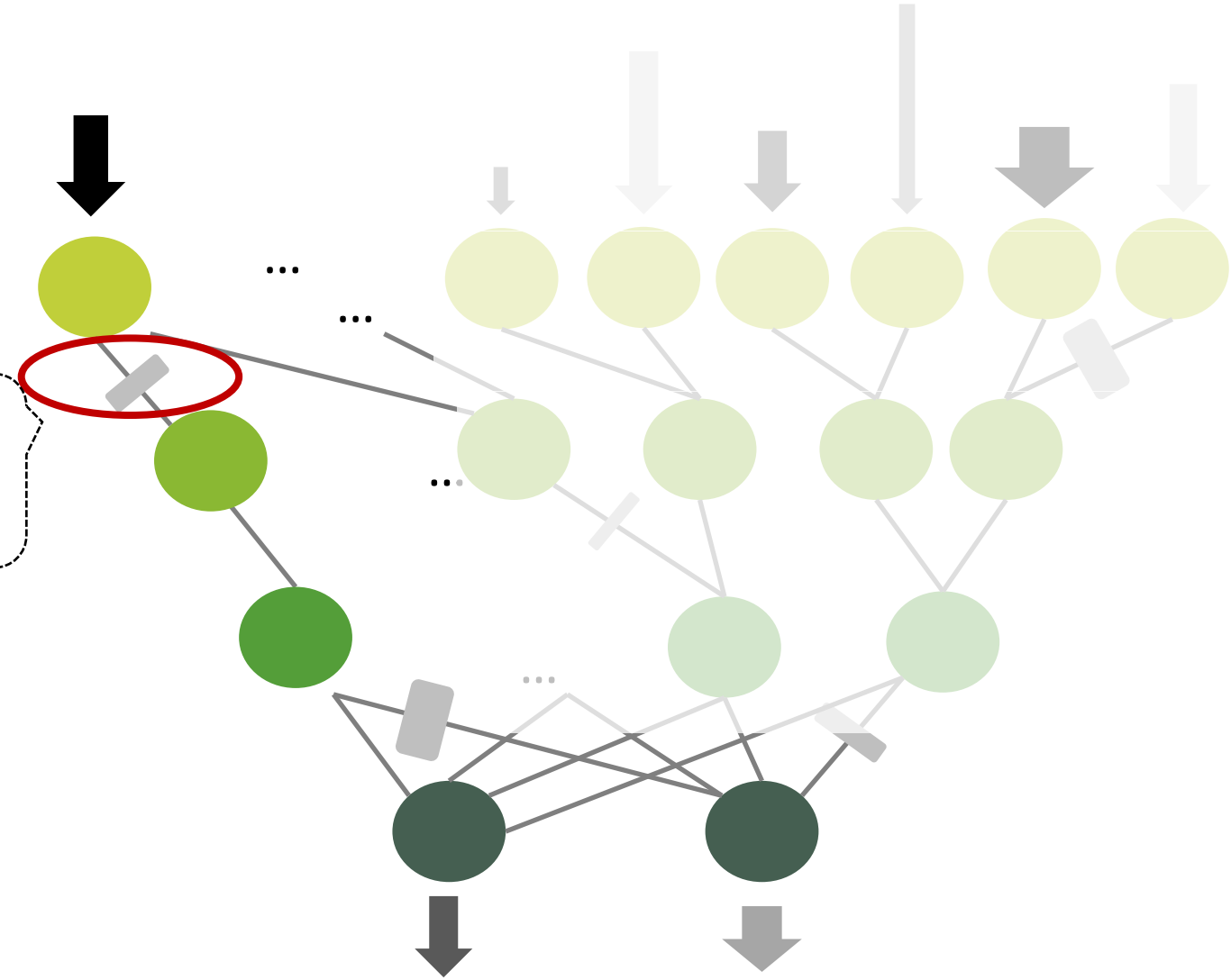
Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



Message Priority (Deadline)

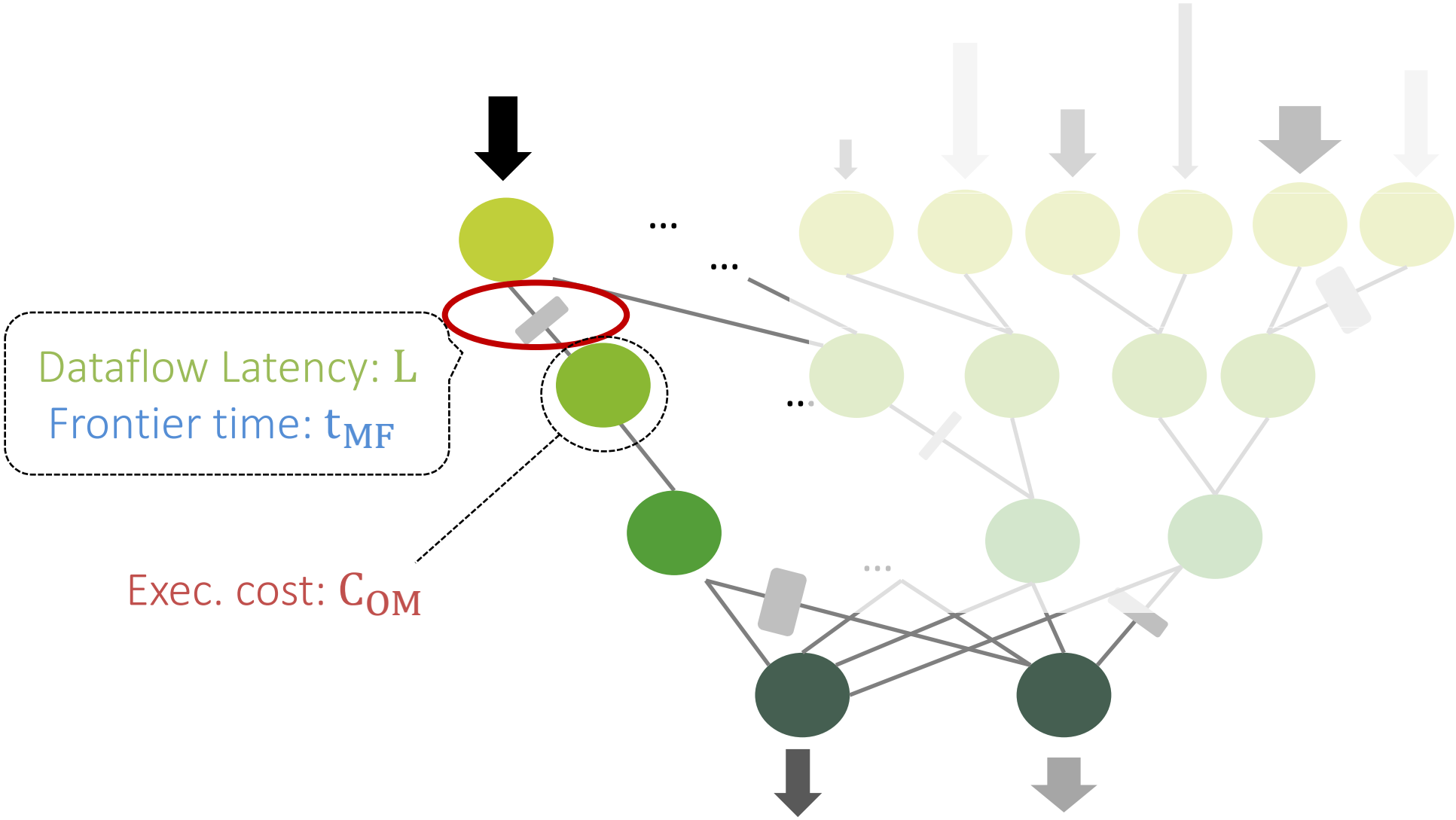
$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



Dataflow Latency: L
Frontier time: t_{MF}

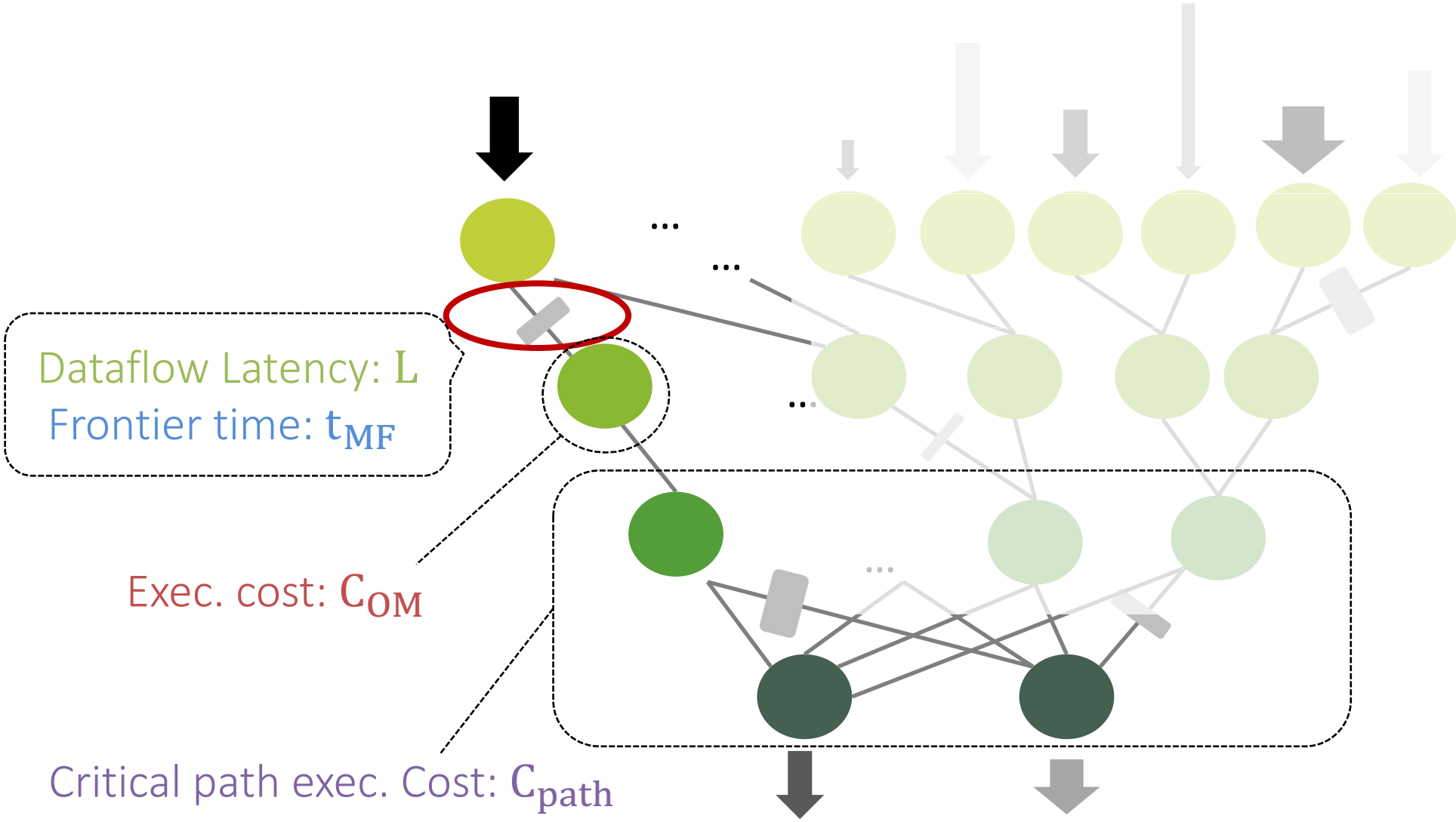
Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



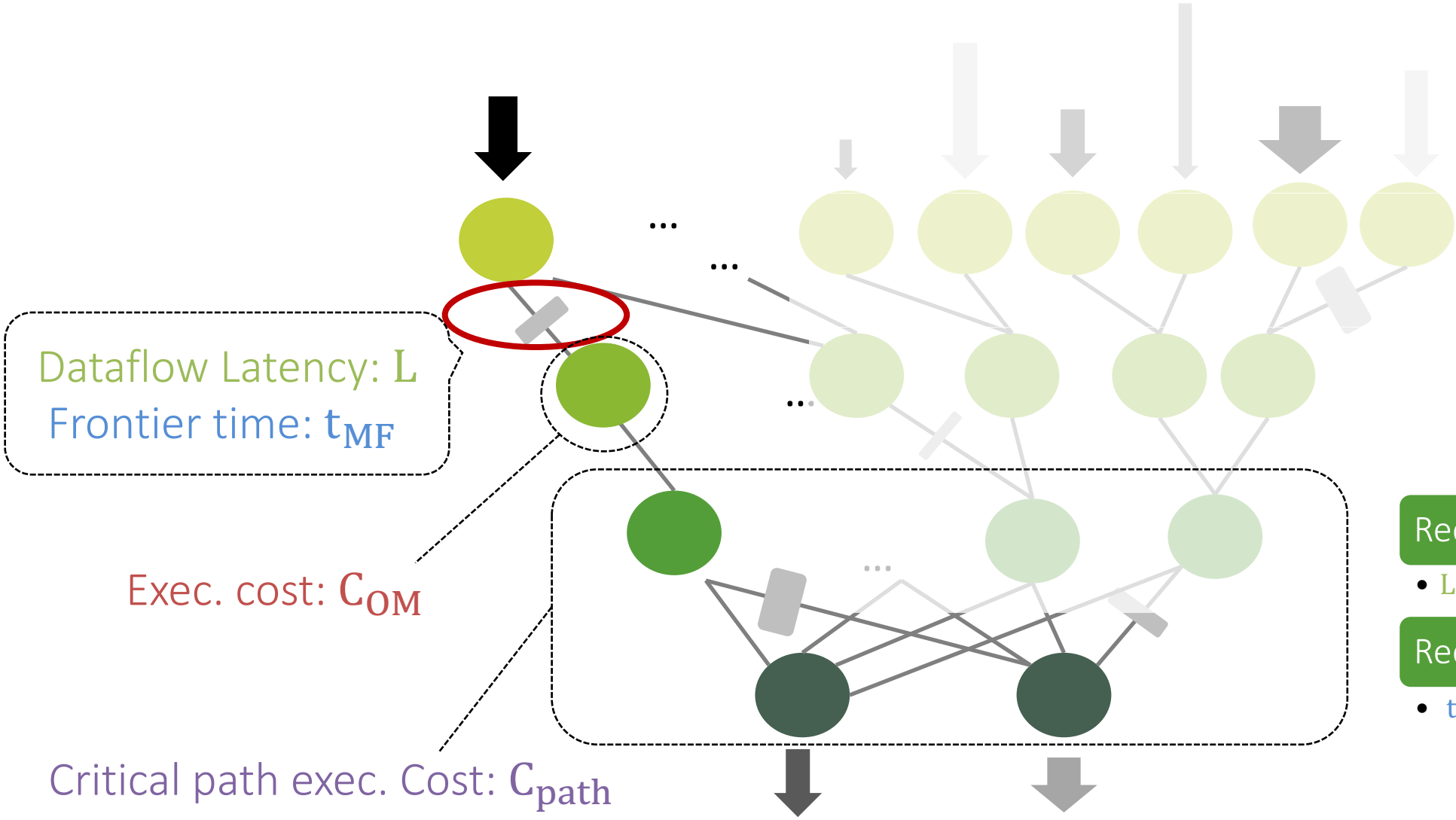
Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$



Requires Static Information

- L, t_{MF}

Requires Dynamic Information

- t_{MF}, C_{OM} , C_{path}

Message Priority (Deadline)

$$ddl_M = t_{MF} + L - C_{OM} - C_{path}$$

Cameo Strategy

- Topology and semantics aware
- Driven by message deadline

Cameo Mechanism

- Light-weight, stateless scheduler
- Scalable priority generation
- Pluggable strategy

Cameo

Data-driven, fine-grained operator scheduling

Cameo Strategy

- Topology and semantics aware
- Driven by message deadline

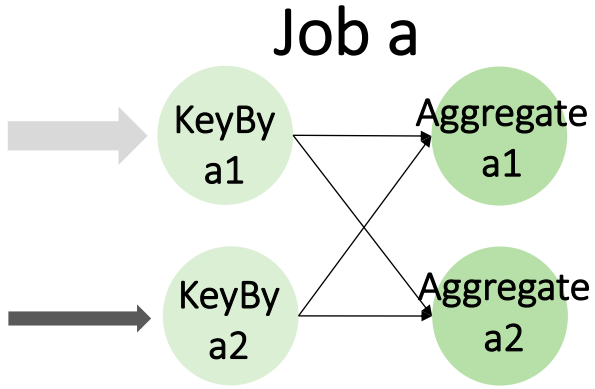
Cameo Mechanism

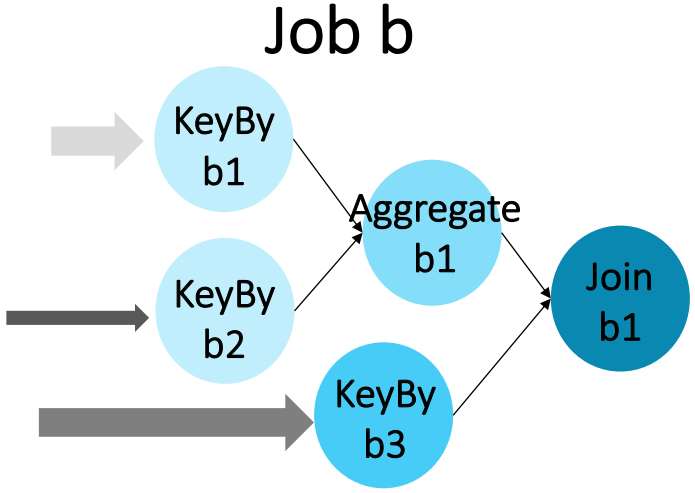
- Light-weight, stateless scheduler
- Scalable priority generation
- Pluggable strategy

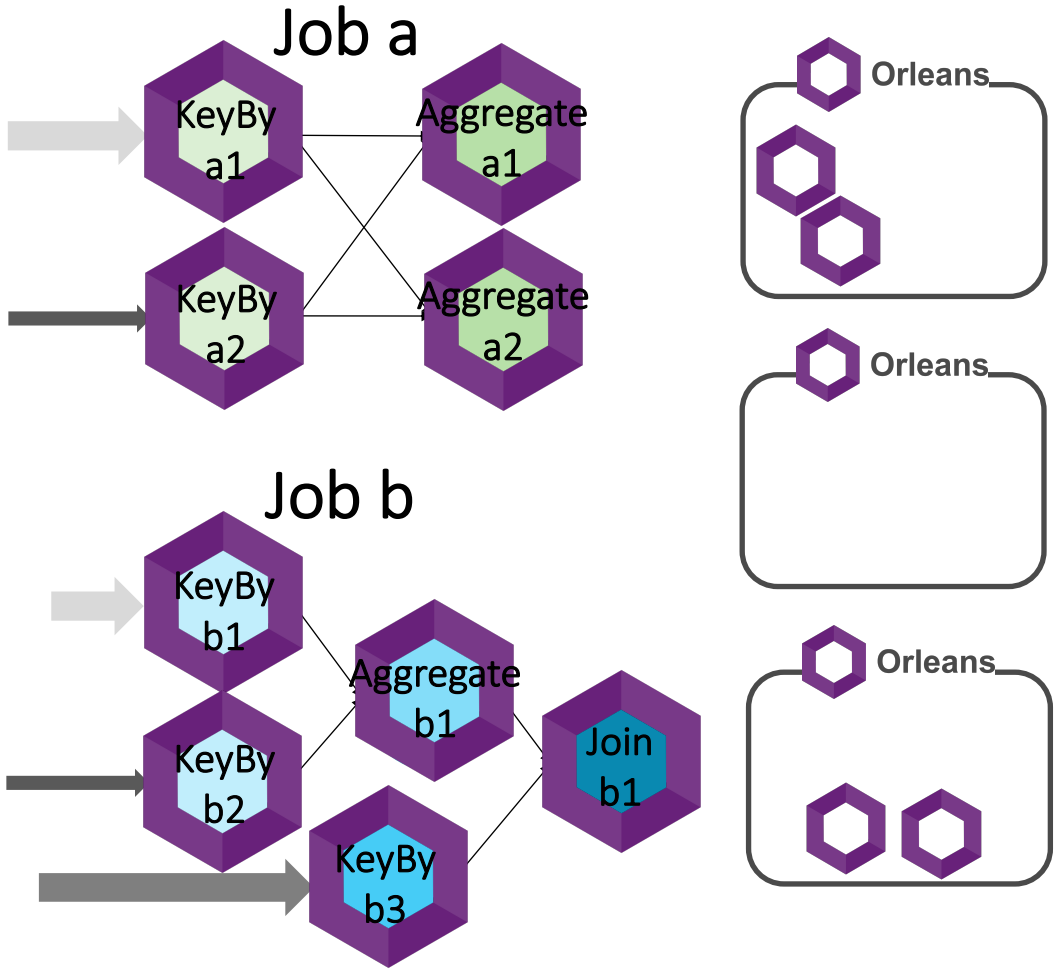
Cameo

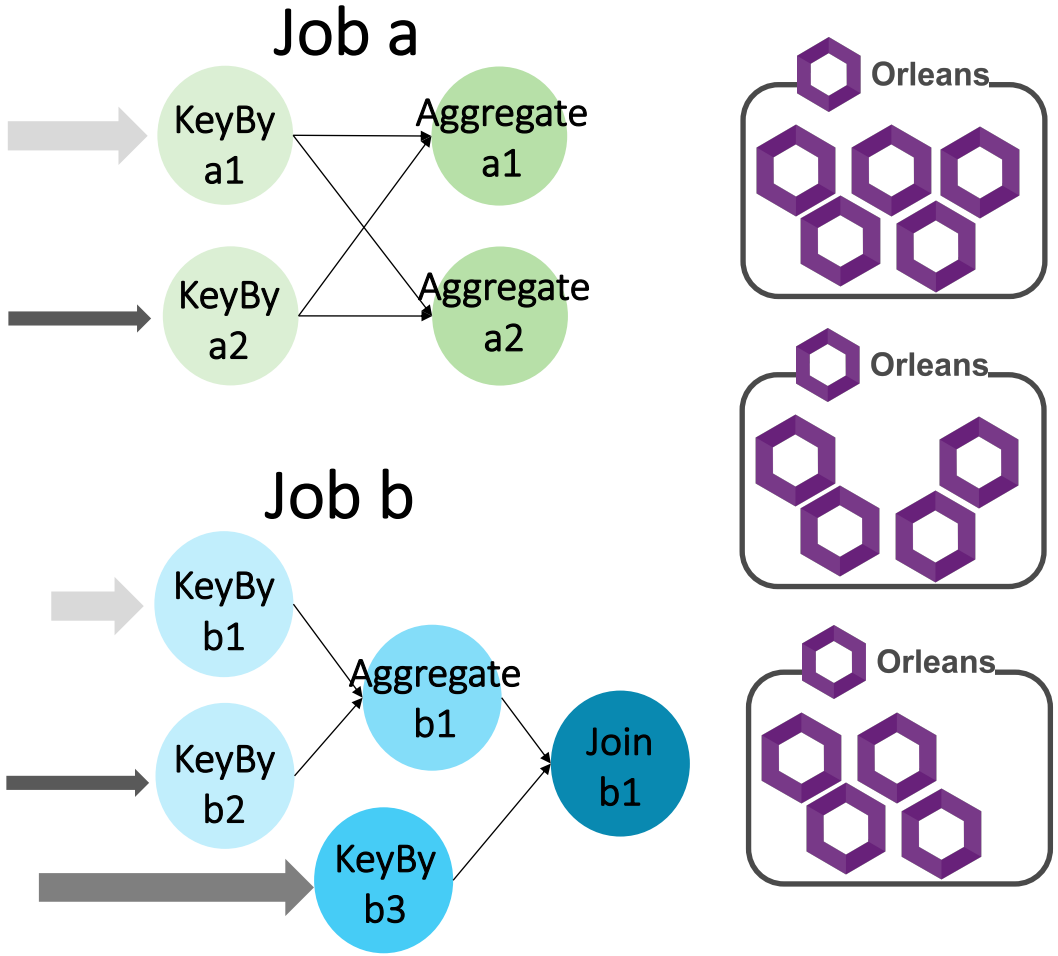
Data-driven, fine-grained operator scheduling

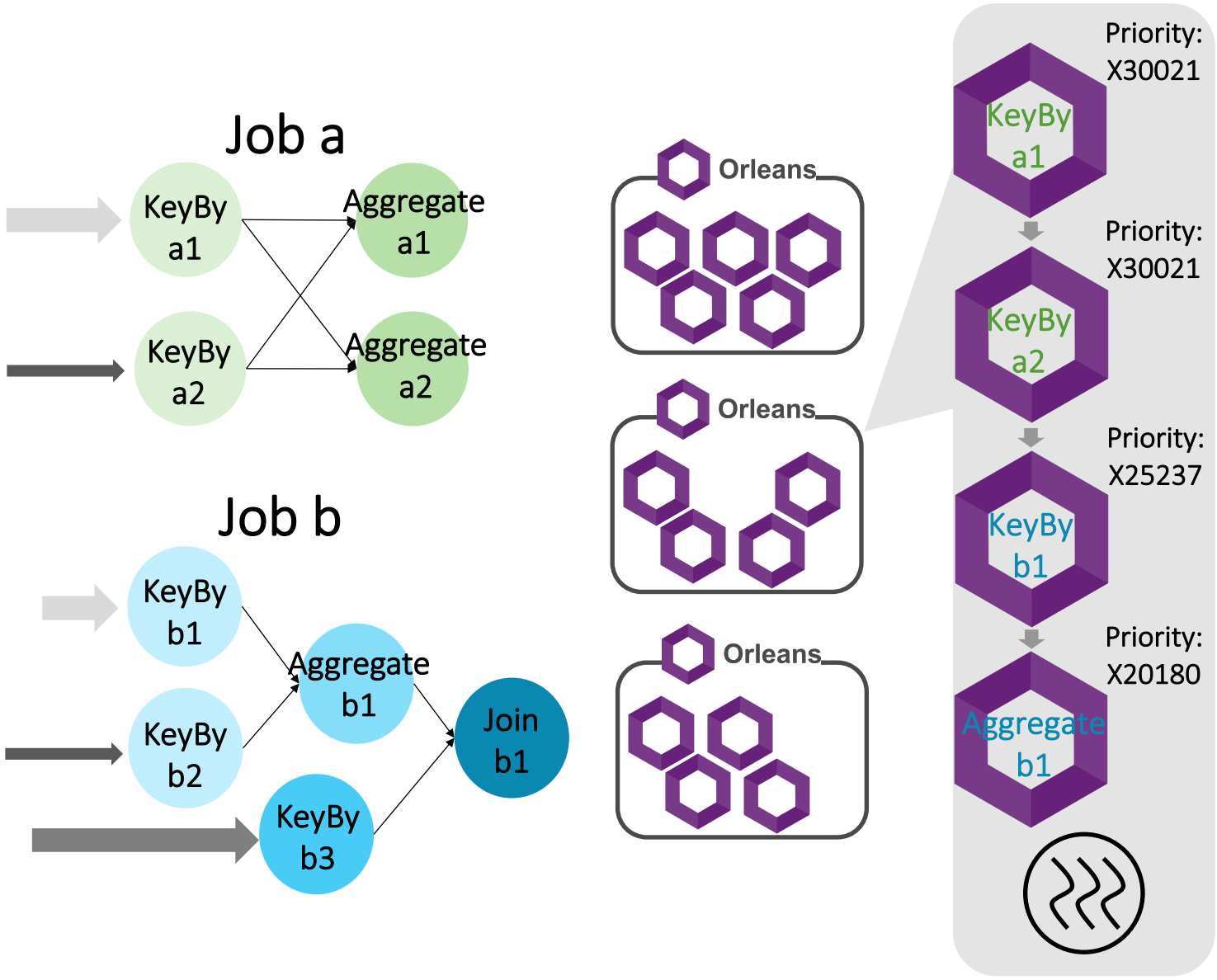
Cameo Mechanism

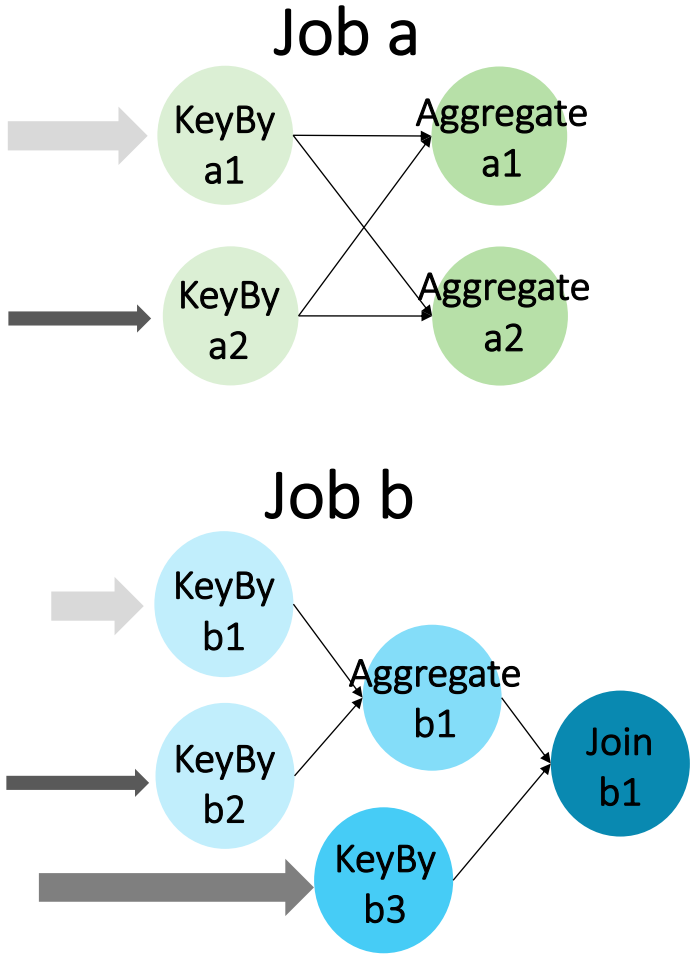

Job a
KeyBy
a1
KeyBy
a2
Aggregate
a1
Aggregate
a2

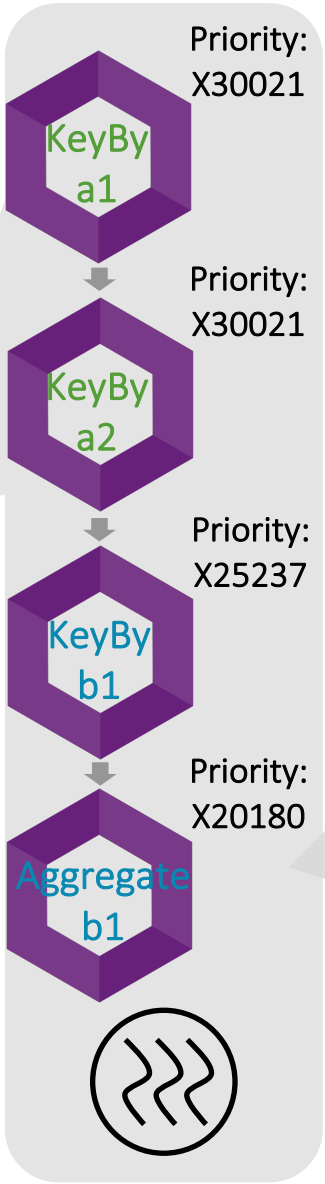

Job b
KeyBy
b1
KeyBy
b2
KeyBy
b3
Aggregate
b1
Join
b1


Job a
KeyBy a1
KeyBy a2
Aggregate a1
Aggregate a2
Job b
KeyBy b1
KeyBy b2
KeyBy b3
Aggregate b1
Join b1
Orleans
Orleans
Orleans

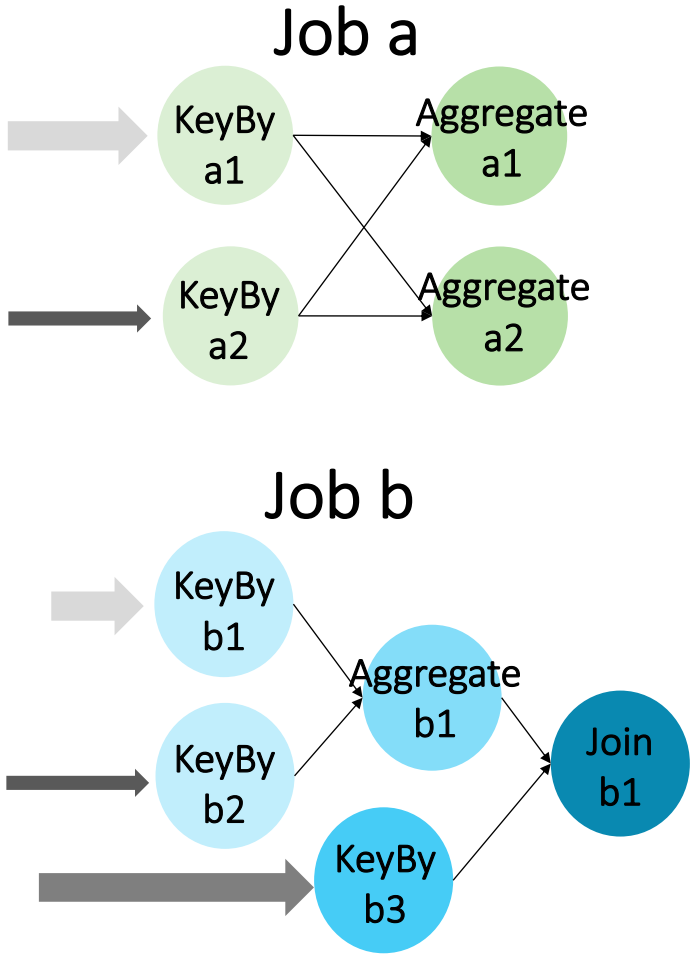

Job a
KeyBy a1
KeyBy a2
Aggregate a1
Aggregate a2
Job b
KeyBy b1
KeyBy b2
KeyBy b3
Aggregate b1
Join b1
Orleans
Orleans
Orleans

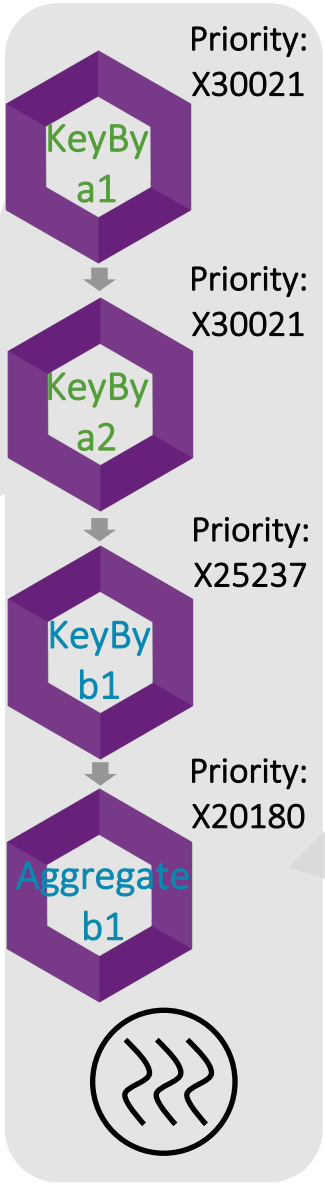

Job a
KeyBy a1
KeyBy a2
Aggregate a1
Aggregate a2
Job b
KeyBy b1
KeyBy b2
KeyBy b3
Aggregate b1
Join b1
Orleans
Orleans
Orleans
Priority: X30021
KeyBy a1
Priority: X30021
KeyBy a2
Priority: X25237
KeyBy b1
Priority: X20180
Aggregate b1



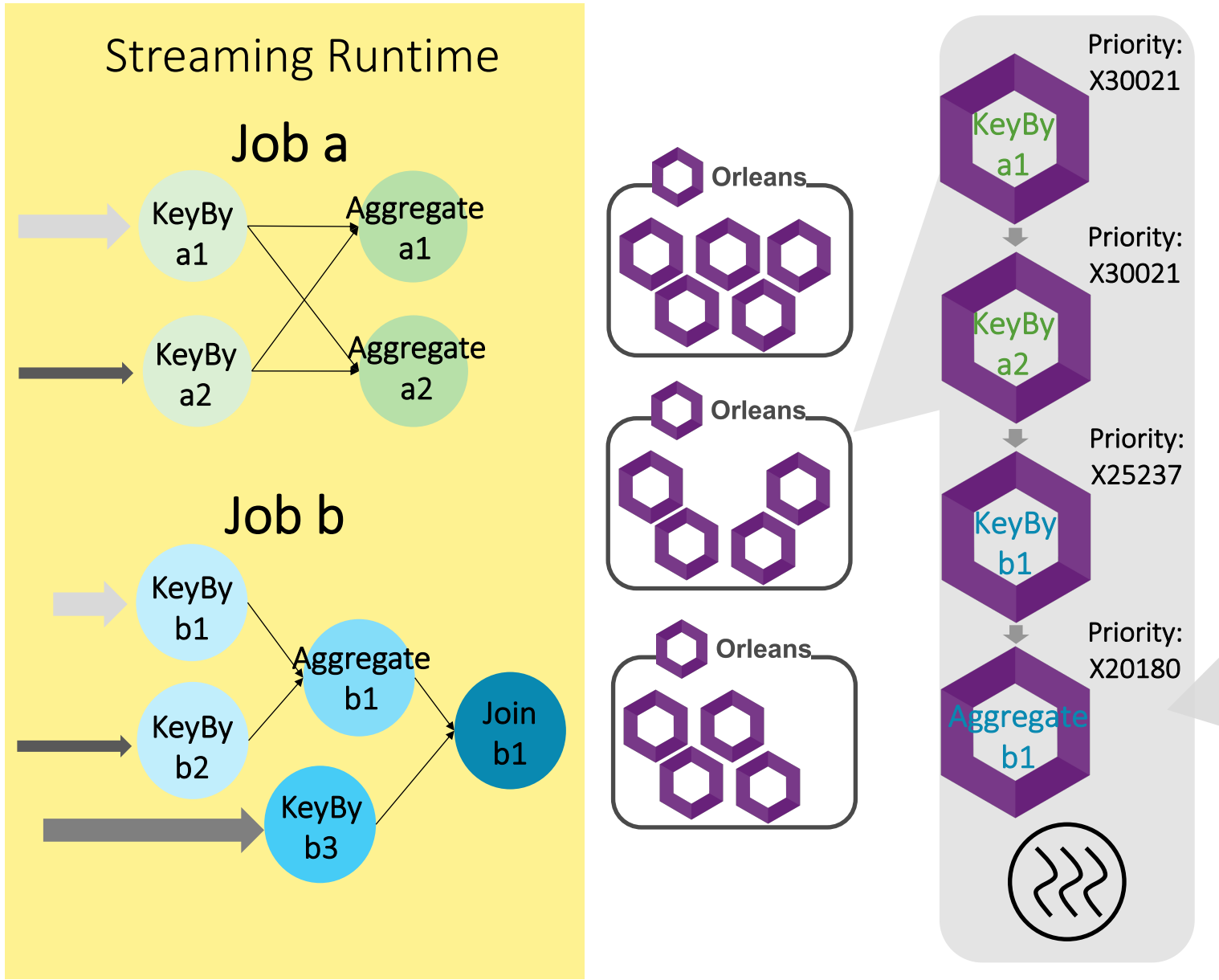


Global Priority	X20180	X30176
Local Priority	X20000	X30000
Pending Messages	Message<RID: 7216>, Message<RID: 7217>	Message<RID: 7218>, Message<RID: 7219>

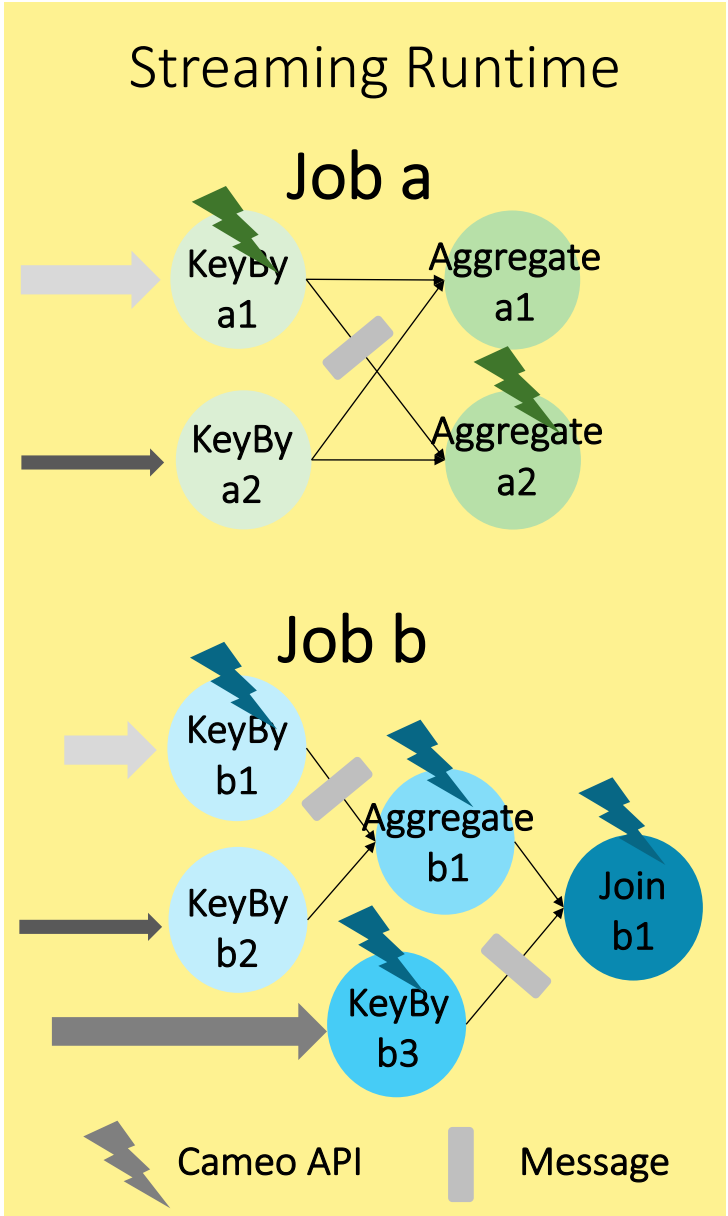


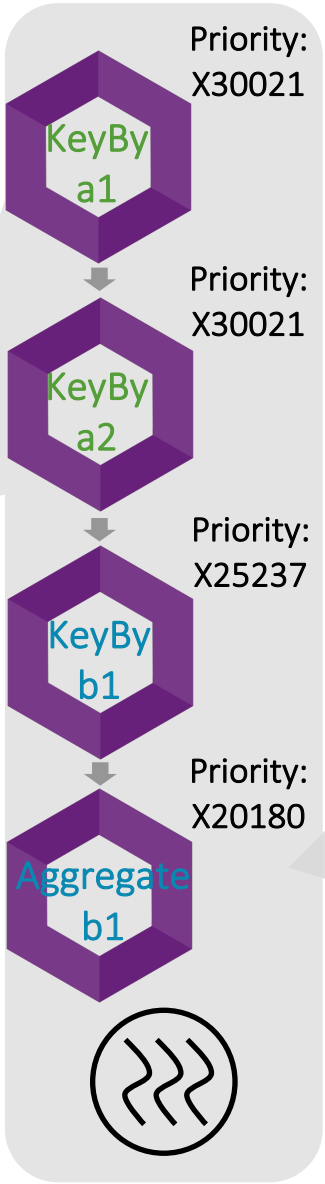


Global Priority	X20180	X30176
Local Priority	X20000	X30000
Pending Messages	Message<RID: 7216>, Message<RID: 7217>	Message<RID: 7218>, Message<RID: 7219>

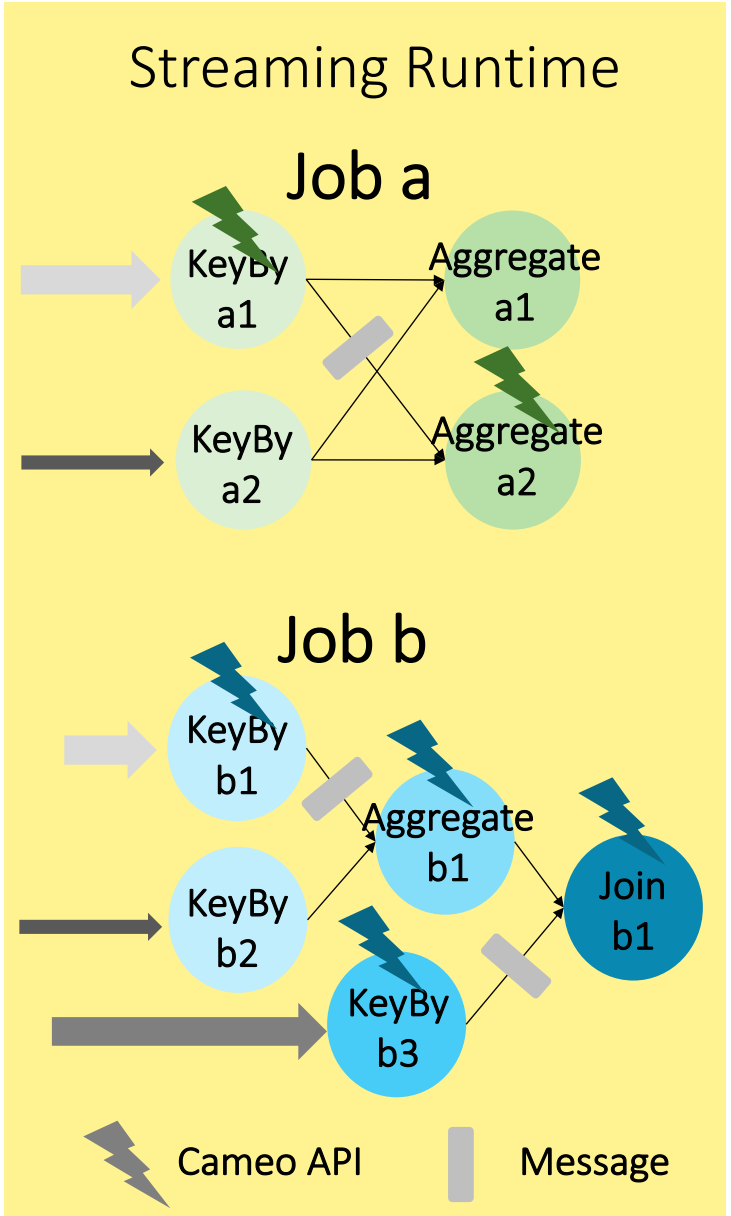


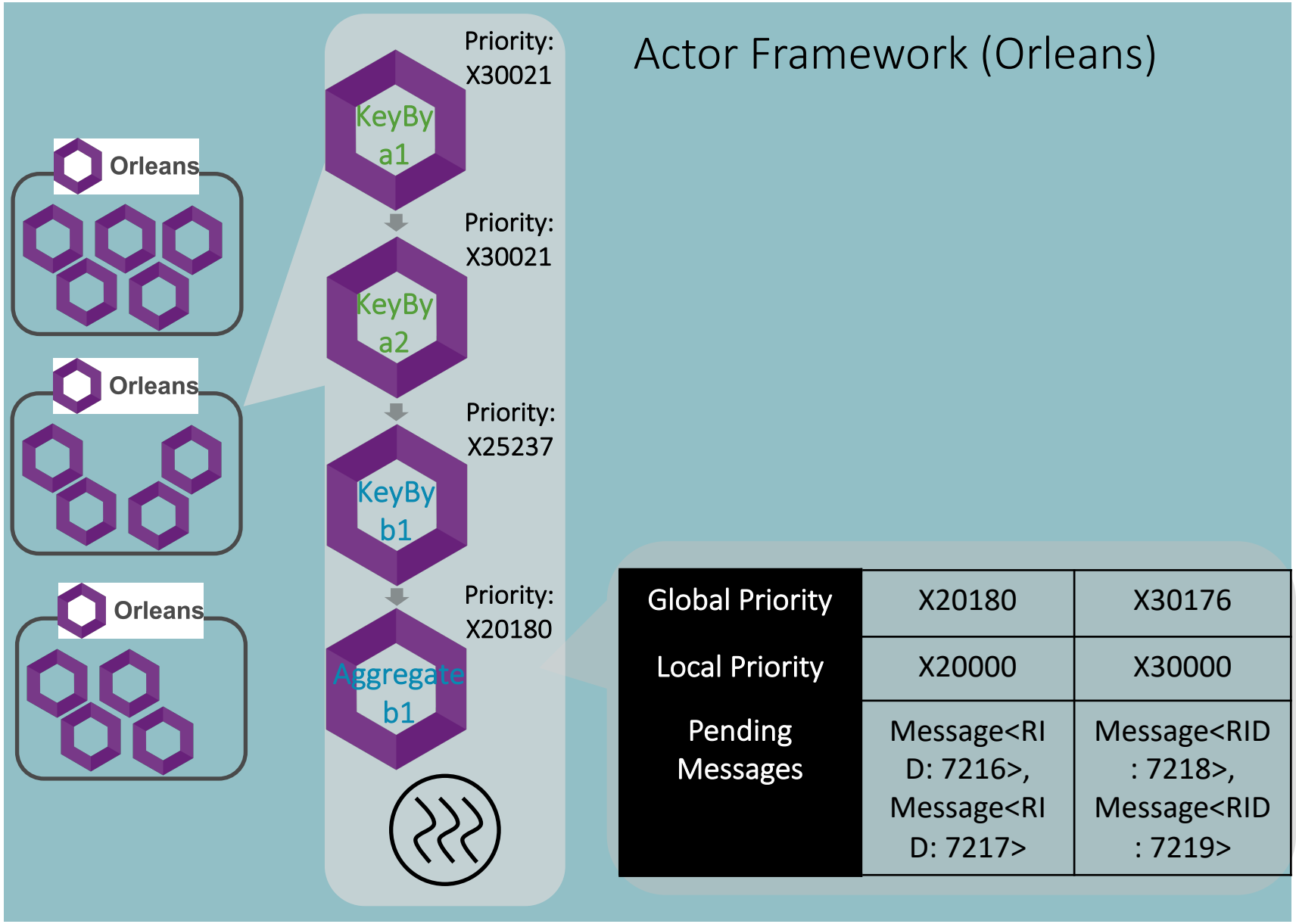
Global Priority	X20180	X30176
Local Priority	X20000	X30000
Pending Messages	Message<RID: 7216>, Message<RID: 7217>	Message<RID: 7218>, Message<RID: 7219>

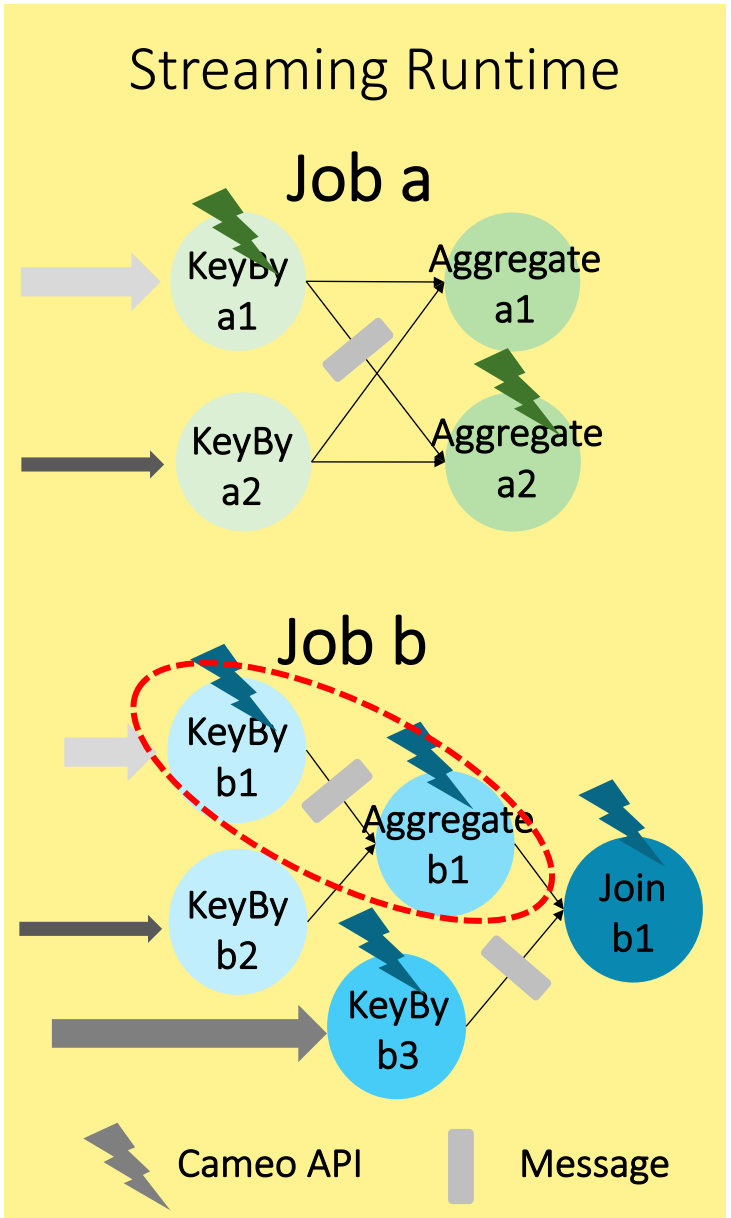


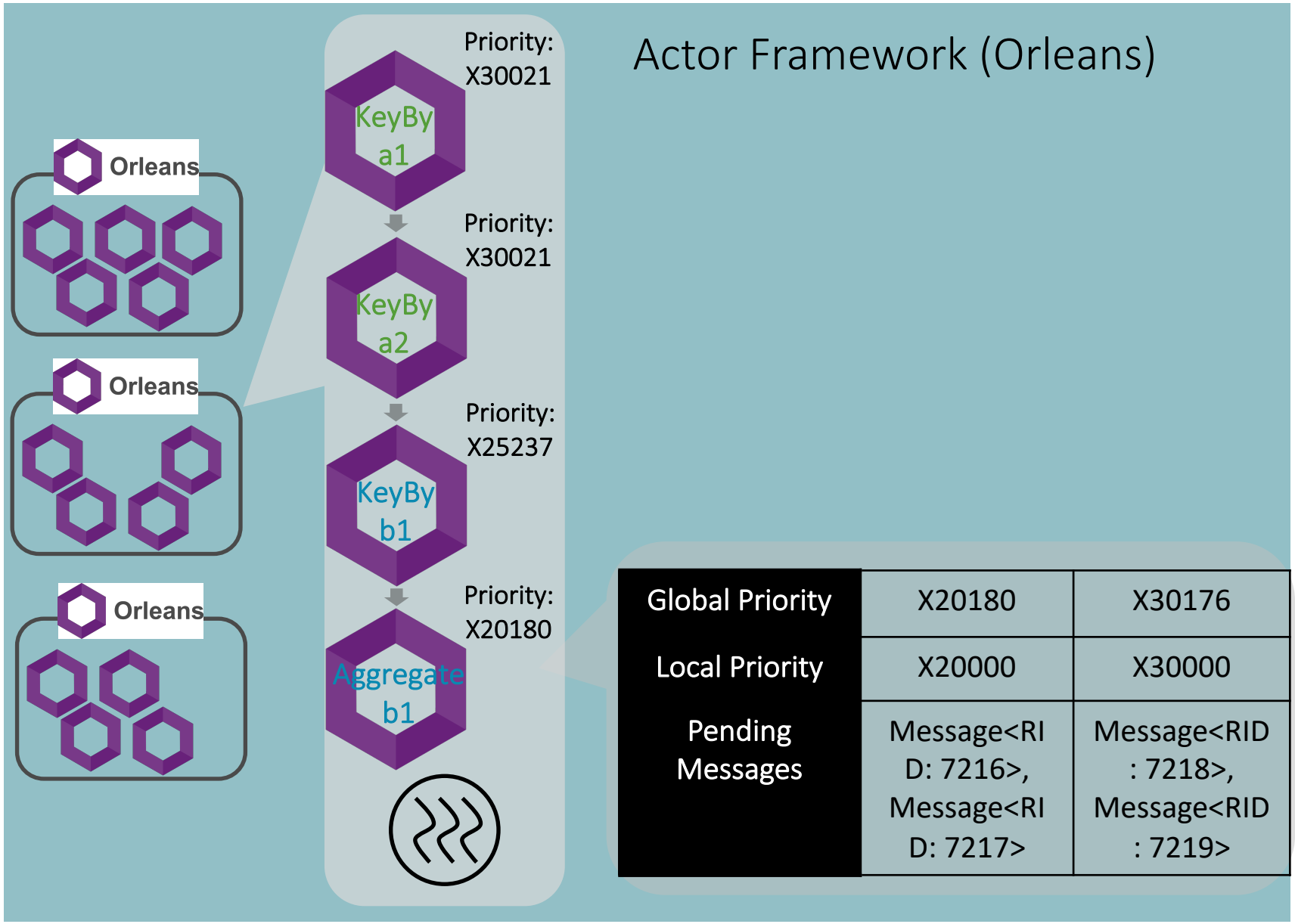


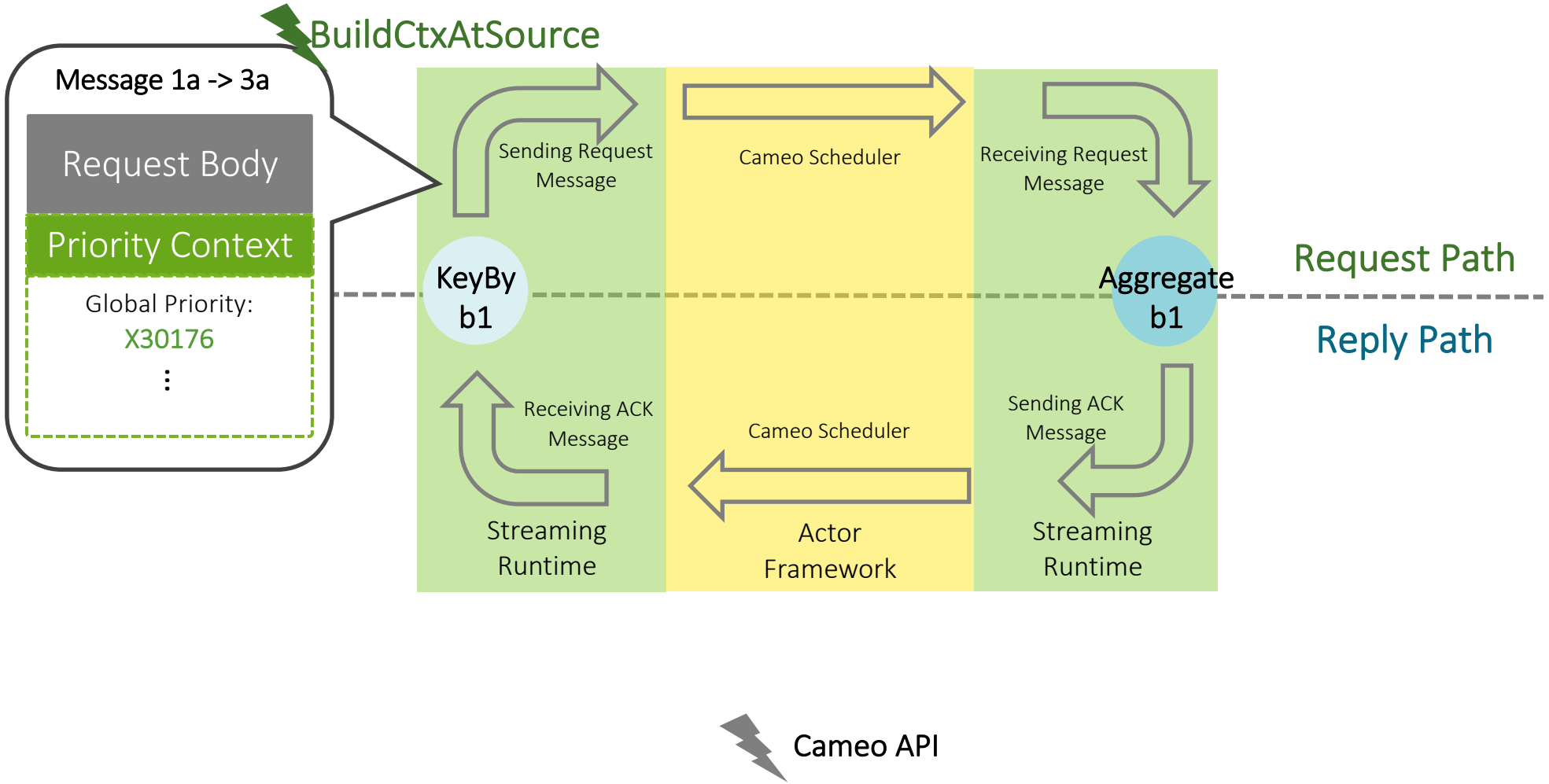
Global Priority	X20180	X30176
Local Priority	X20000	X30000
Pending Messages	Message<RID: 7216>, Message<RID: 7217>	Message<RID: 7218>, Message<RID: 7219>

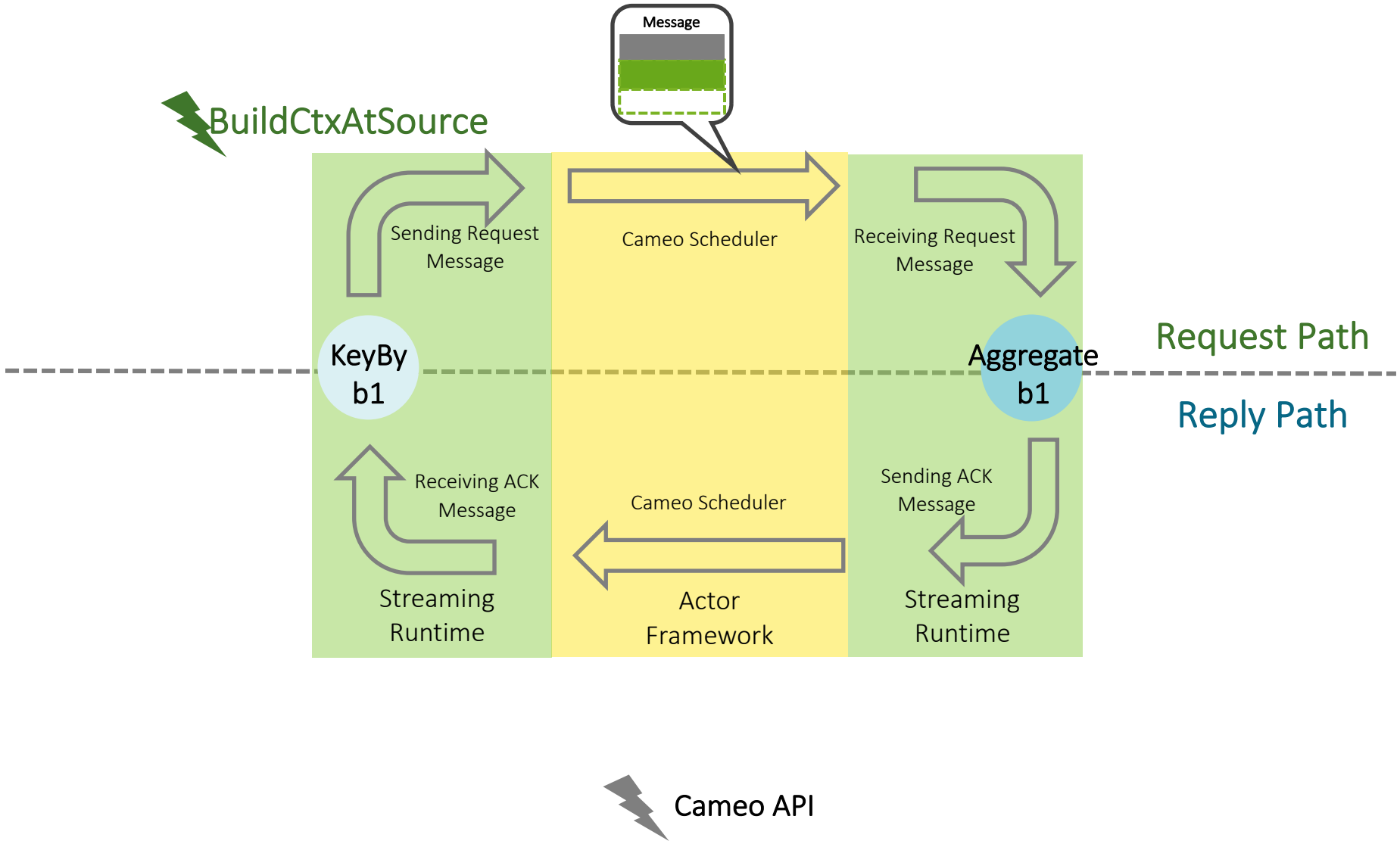

Streaming Runtime
Job a
KeyBy a1
KeyBy a2
Aggregate a1
Aggregate a2
Job b
KeyBy b1
KeyBy b2
KeyBy b3
Aggregate b1
Join b1
Cameo API
Message

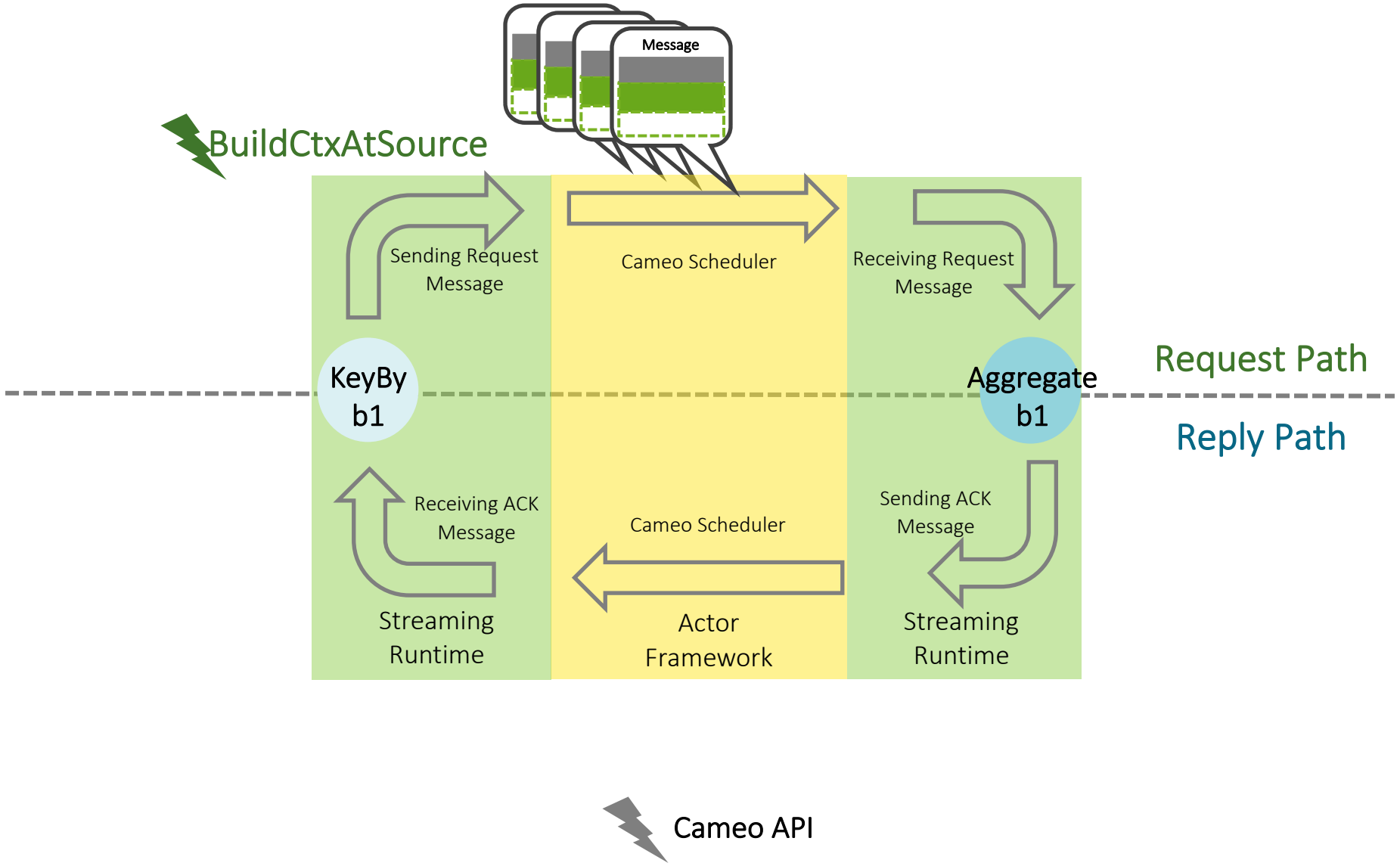

Actor Framework (Orleans)
Orleans
Orleans
Orleans
Priority: X30021
KeyBy a1
Priority: X30021
KeyBy a2
Priority: X25237
KeyBy b1
Priority: X20180
Aggregate b1
Global Priority
X20180
X30176
Local Priority
X20000
X30000
Pending Messages
Message<RID: 7216>, Message<RID: 7217>
Message<RID: 7218>, Message<RID: 7219>

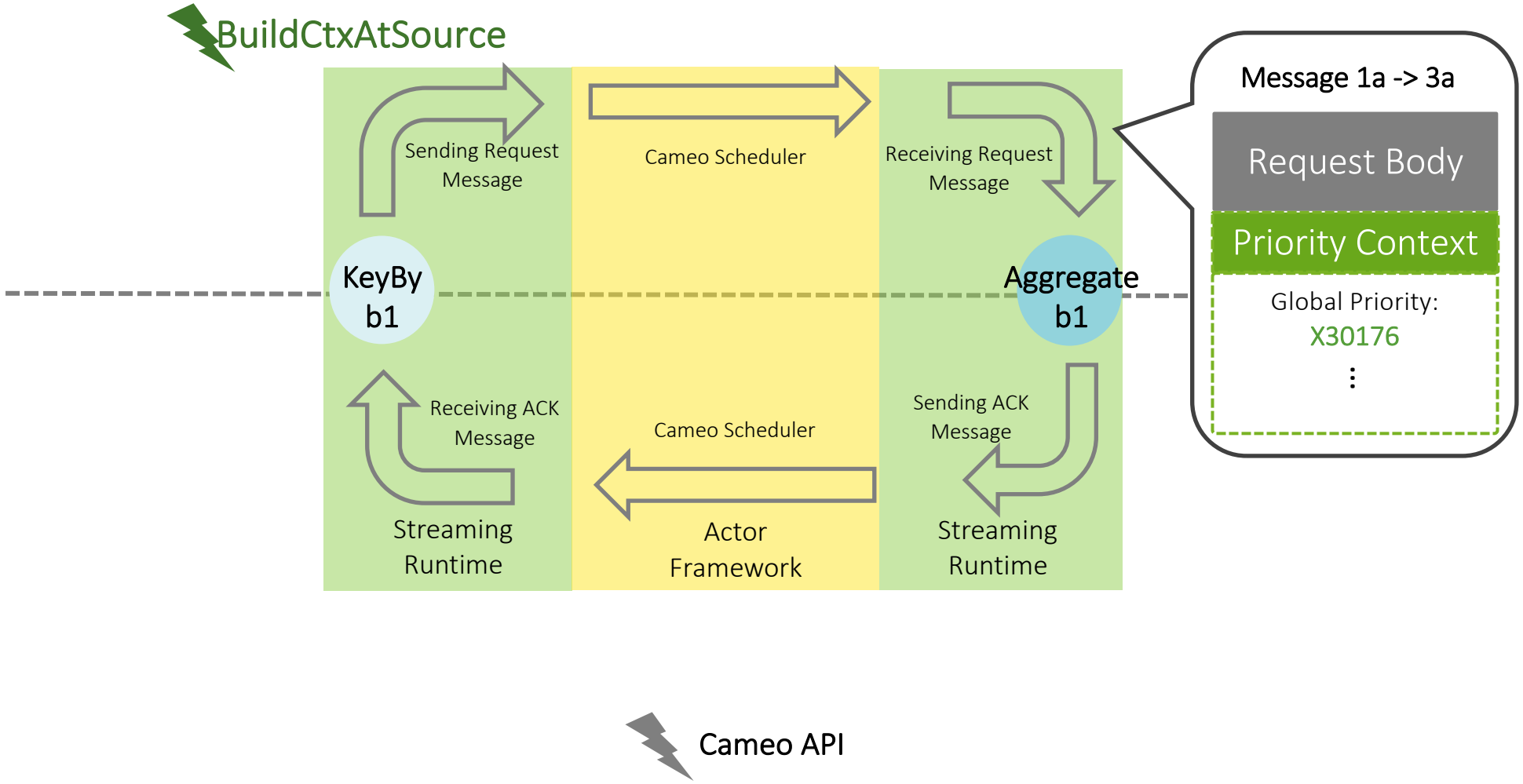

Streaming Runtime
Job a
KeyBy a1
KeyBy a2
Aggregate a1
Aggregate a2
Job b
KeyBy b1
KeyBy b2
KeyBy b3
Aggregate b1
Join b1
Cameo API
Message

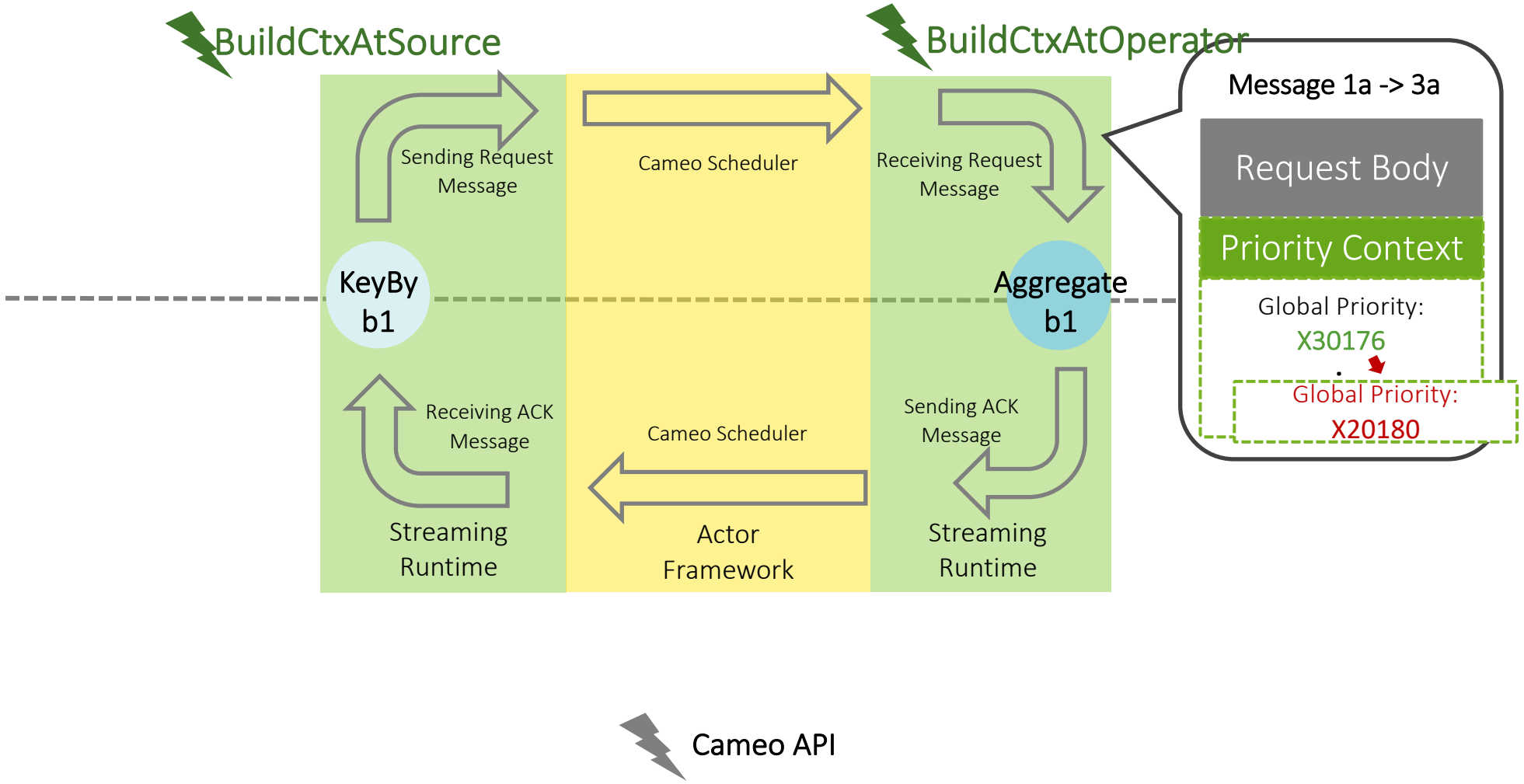

Actor Framework (Orleans)
Orleans
Orleans
Orleans
Priority: X30021
KeyBy a1
Priority: X30021
KeyBy a2
Priority: X25237
KeyBy b1
Priority: X20180
Aggregate b1
Global Priority
X20180
X30176
Local Priority
X20000
X30000
Pending Messages
Message<RID: 7216>, Message<RID: 7217>
Message<RID: 7218>, Message<RID: 7219>

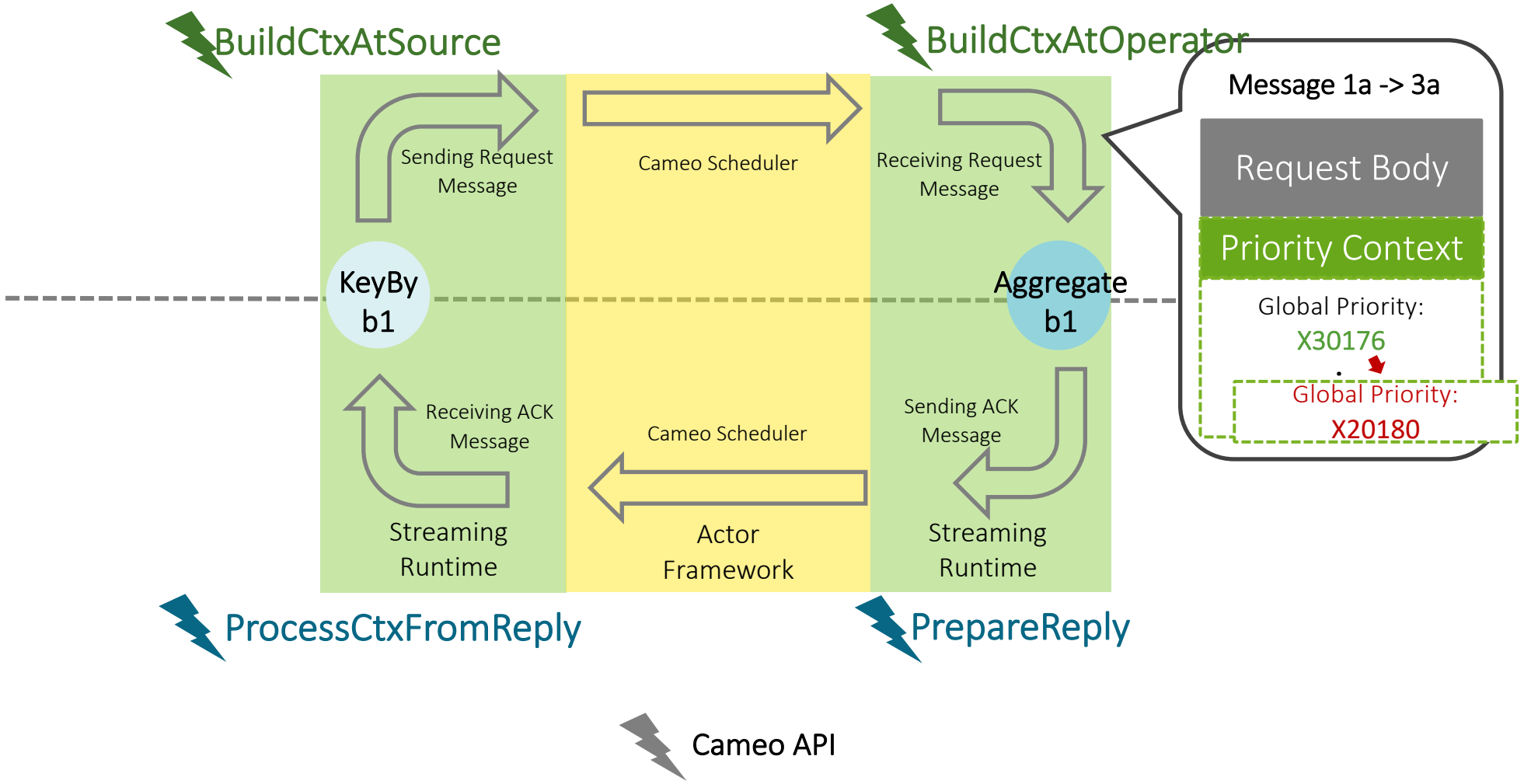

BuildCtxAtSource
Message 1a -> 3a
Request Body
Priority Context
Global Priority:
X30176
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Request Path
Reply Path
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
Cameo API


Message
BuildCtxAtSource
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Request Path
Reply Path
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
Cameo API


Message
BuildCtxAtSource
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Request Path
Reply Path
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
Cameo API

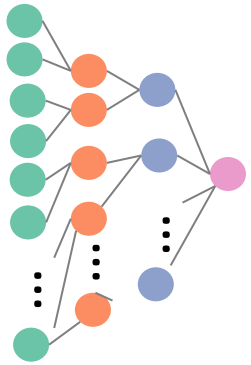

BuildCtxAtSource
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
Message 1a -> 3a
Request Body
Priority Context
Global Priority:
X30176
Cameo API

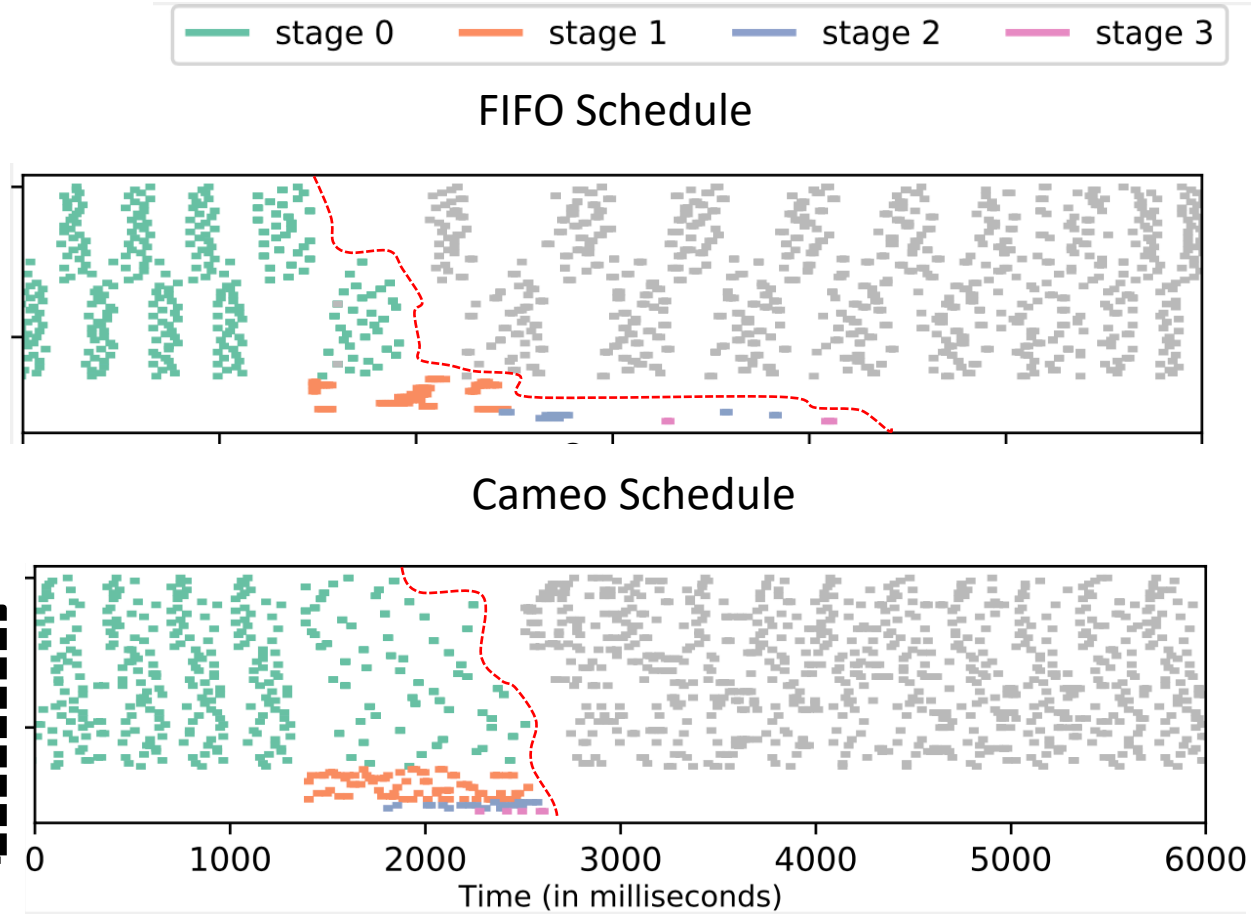

BuildCtxAtSource
BuildCtxAtOperator
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
Message 1a -> 3a
Request Body
Priority Context
Global Priority:
X30176
Global Priority:
X20180
Cameo API


BuildCtxAtSource
BuildCtxAtOperator
Sending Request Message
Cameo Scheduler
Receiving Request Message
KeyBy b1
Aggregate b1
Receiving ACK Message
Cameo Scheduler
Sending ACK Message
Streaming Runtime
Actor Framework
Streaming Runtime
ProcessCtxFromReply
PrepareReply
Cameo API
Message 1a -> 3a
Request Body
Priority Context
Global Priority:
X30176
Global Priority:
X20180

Evaluation

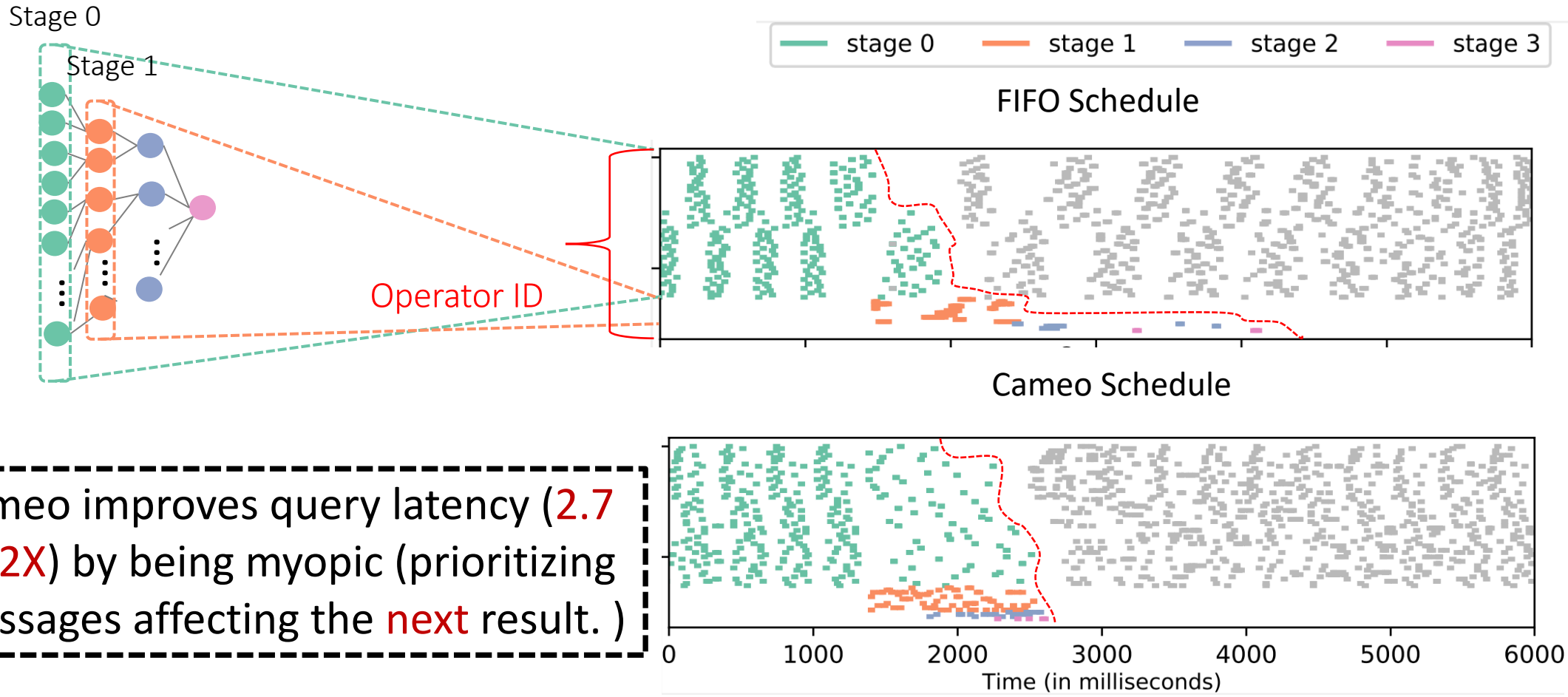
Cameo improves single-query performance





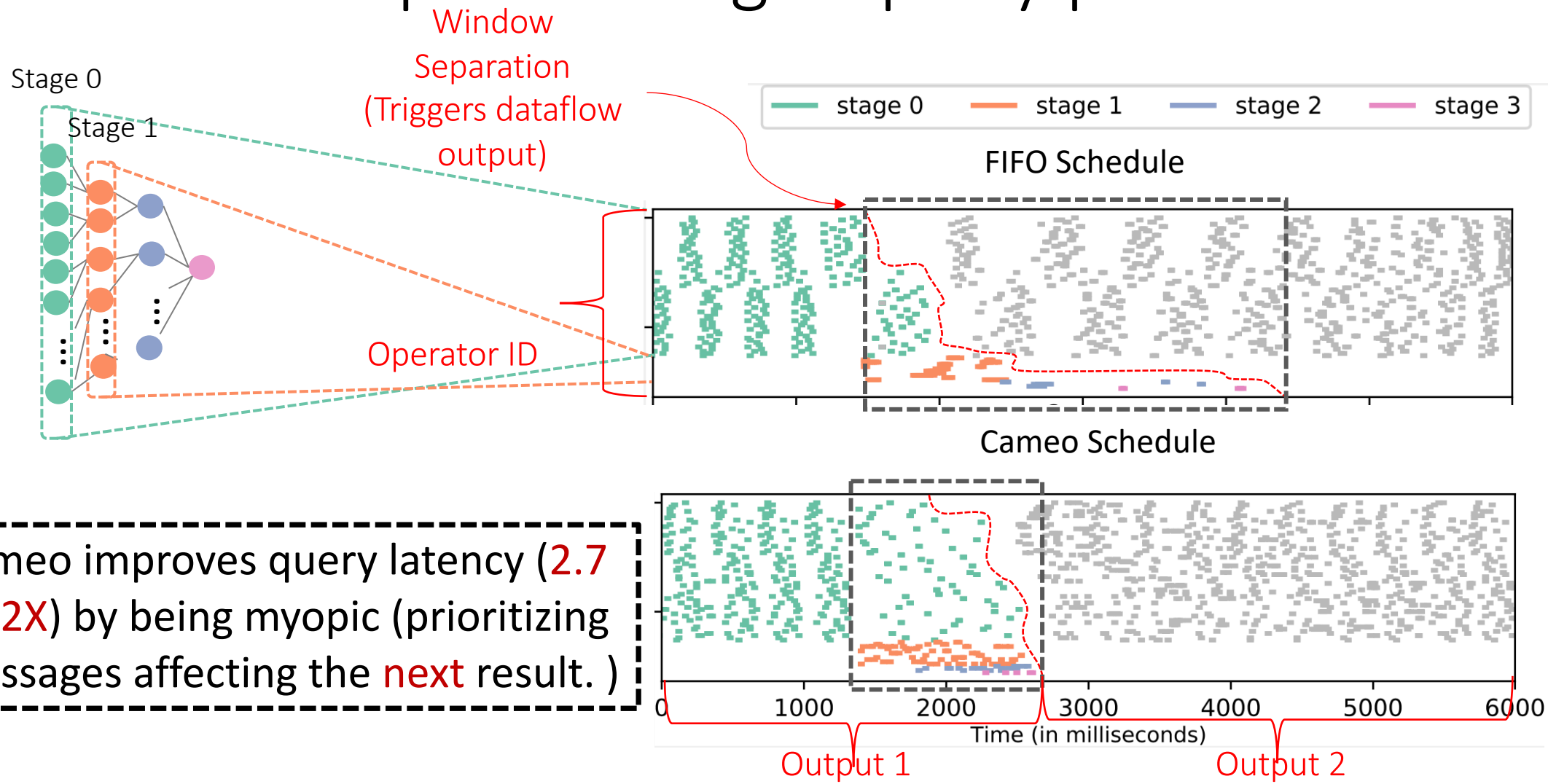
Cameo improves query latency (2.7 - 3.2X) by being myopic (prioritizing messages affecting the next result.)

Cameo improves single-query performance



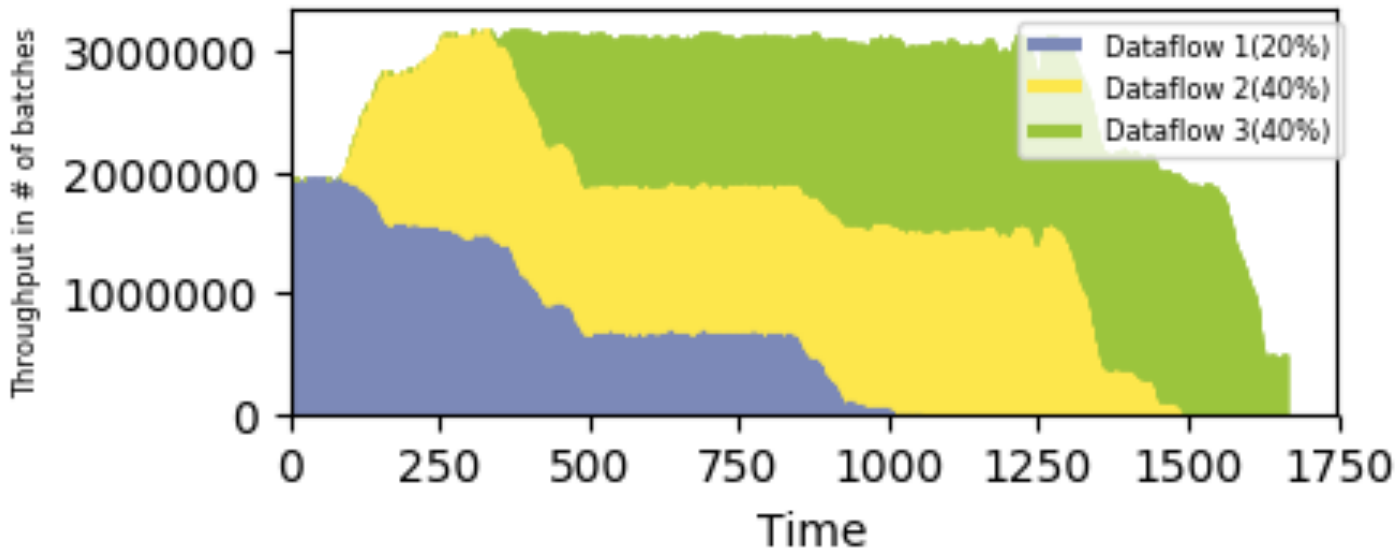
Cameo improves query latency (2.7 - 3.2X) by being myopic (prioritizing messages affecting the next result.)

Cameo improves single-query performance



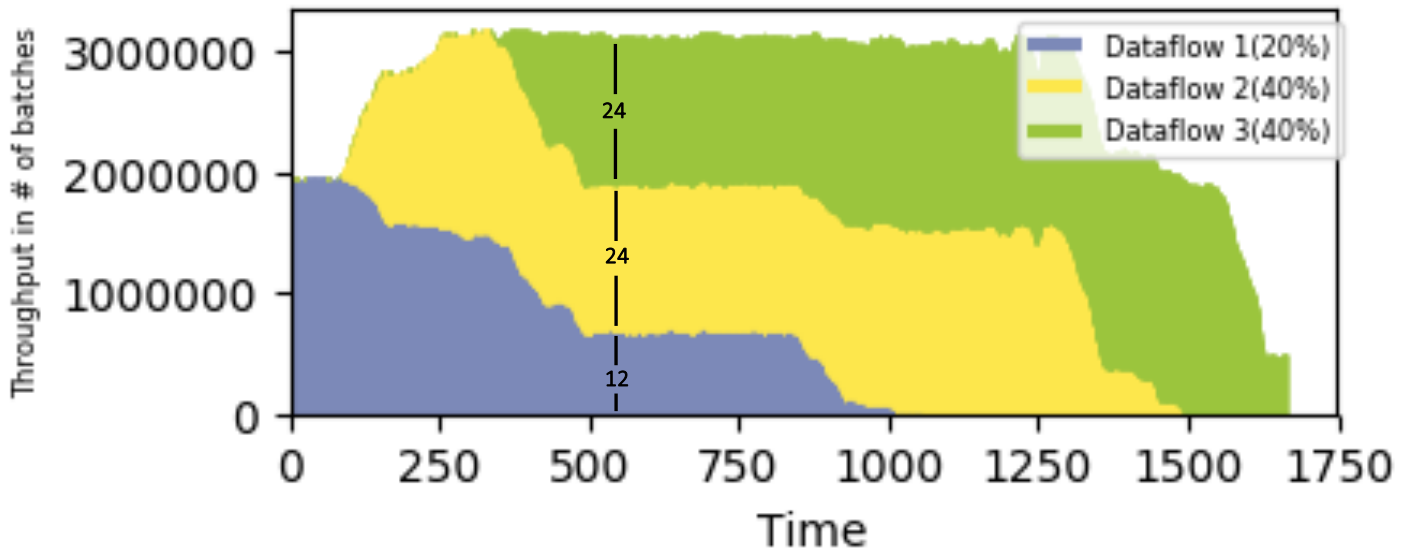
Cameo improves query latency (2.7 - 3.2X) by being myopic (prioritizing messages affecting the next result.)

Cameo supports plugged-in strategies



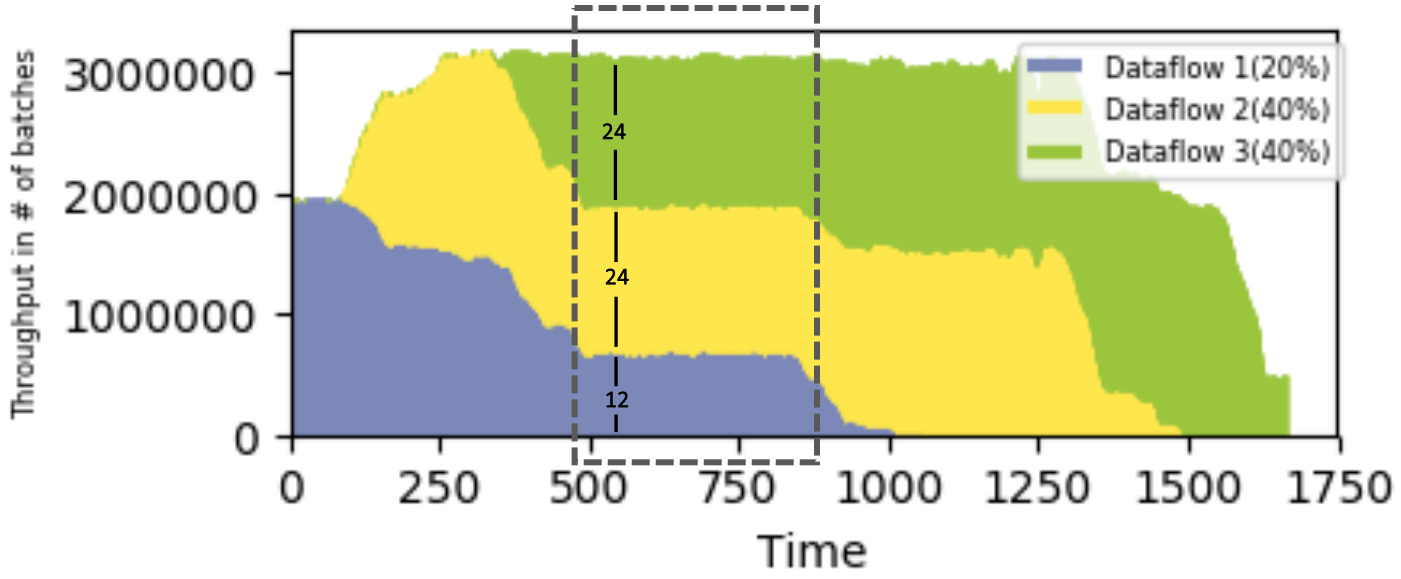
Proportional fair sharing. Three identical dataflows.
Dataflows start 300 seconds apart

Cameo supports plugged-in strategies



Proportional fair sharing. Three identical dataflows.
Dataflows start 300 seconds apart

Cameo supports plugged-in strategies



Proportional fair sharing. Three identical dataflows.
Dataflows start 300 seconds apart

Before we end…

More in the paper…

- Cameo improves query performance:
  - Cameo reduces query latency by 2.7X for single query execution, 4.6X in multi tenant scenarios
  - Cameo weathers transient spikes by reducing tail latency by 21X – 30X
- Cameo supports general scheduling and user-defined scheduling states
- Cameo infers dataflow output based on input messages

More in the paper…

- Cameo improves query performance:
  - Cameo reduces query latency by 2.7X for single query execution, 4.6X in multi tenant scenarios
  - Cameo weathers transient spikes by reducing tail latency by 21X – 30X
- Cameo supports general scheduling and user-defined scheduling states
- Cameo infers dataflow output based on input messages

Cameo proposes fine-grained scheduling scheme for serverless, event-driven stream processing services that achieves both high utilization and controllable performance

For questions: contact author <lexu1@Illinois.edu>