Forging a Functional Enterprise

How thinking functionally transforms line-of-business applications

Scott Havens
Director of Engineering
WalmartLabs and Jet.com


Jet


jet

What we'll talk about

- Functional principles
- System design & architecture
- Benefits (DevOps and more!)

Principles of functional programming

- Immutability
- Actions, not objects
- Purity

What is designing functionally?

Immutability →

- *Message-based communication*
- *Event sourcing*

What is designing functionally?

Immutability →

- *Message-based communication*
- *Event sourcing*

Actions, not objects →

- *Verbs, not nouns*

What is designing functionally?

Immutability →

- *Message-based communication*
- *Event sourcing*

Actions, not objects →

- *Verbs, not nouns*

Purity →

- *Isolate computations from the real world*
- *Computations and data are interchangeable*
- *No dual writes*

Introducing “Panther”

Panther

Inventory tracking

Reservation management

Panther functional goals

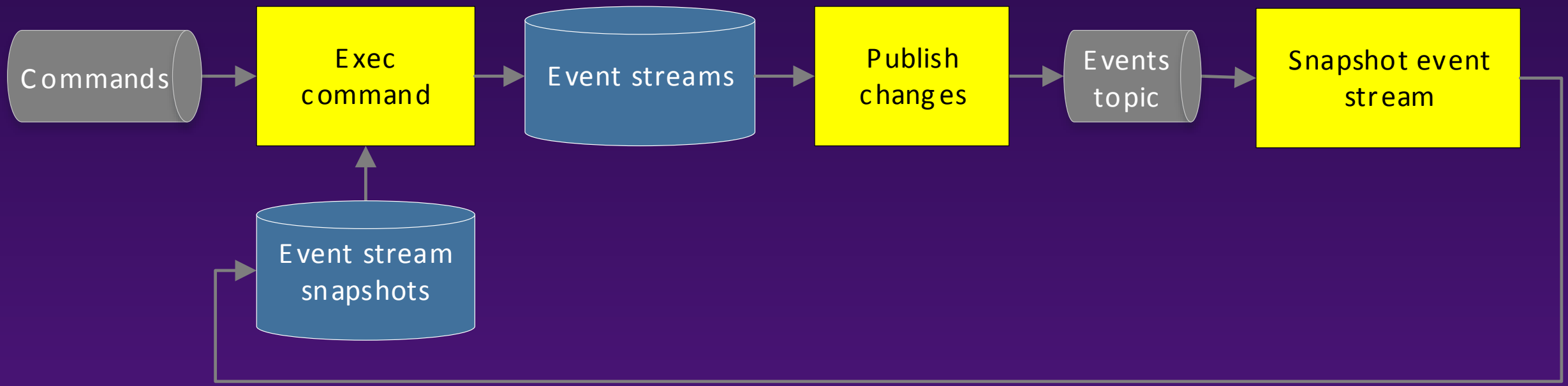
- *Maximize availability*
- *Minimize reject rates*

- *Improve customer experience*
- *Enhance insights*
- *Unify inventory mgmt*

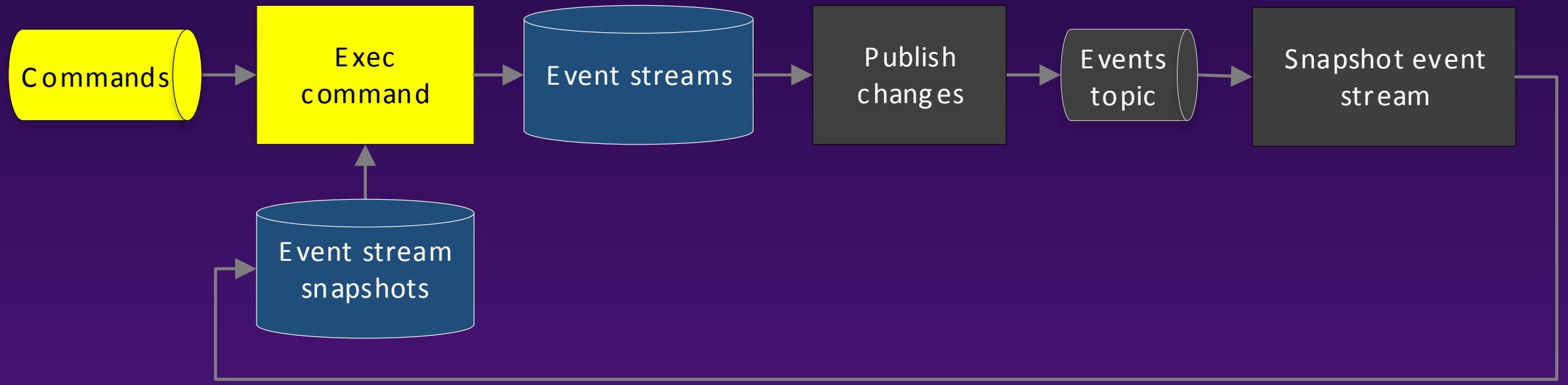
Panther nonfunctional goals

- *High availability*
- *Georedundancy*
- *SLA-backed performance*

Core flow


Commands
Exec command
Event streams
Publish changes
Events topic
Snapshot event stream
Event stream snapshots

Core flow – executing commands, producing events



1. **Ingest message**
2. **Deserialize strongly typed command**
3. Retrieve current state from database
4. Execute command
5. Commit output event

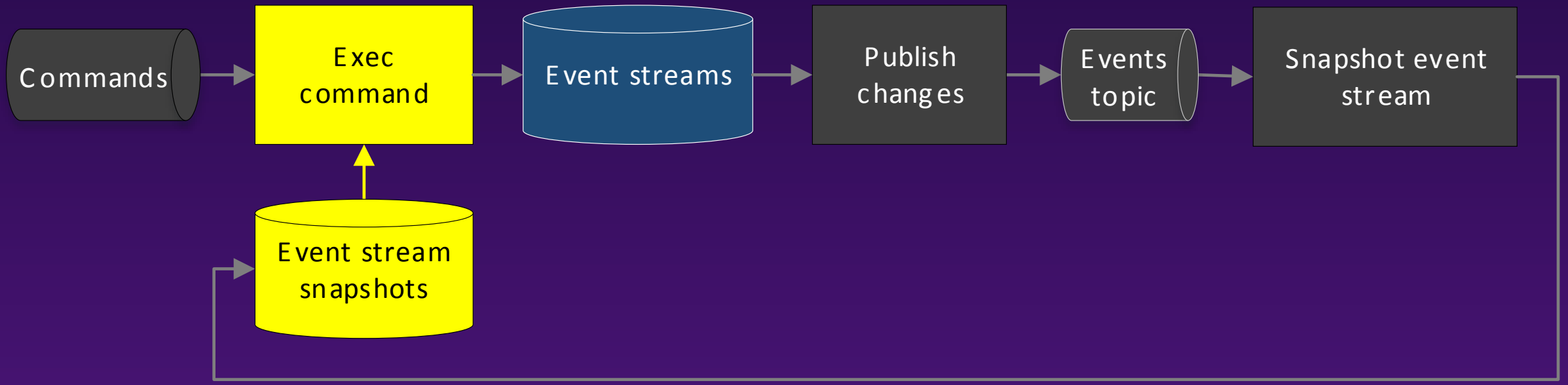
Commands

```
UpdateInventory

ReserveInventory

CancelReservation

ShipOrder

ArchiveItem
```

Strongly typed commands

```
type ReserveInventory
{
  id : CommandId
  itemId : ItemId
  warehouseId : WarehouseId
  quantity : Quantity
  orderId : CustomerOrderId
  at : DateTime
}
```

Core flow – executing commands, producing events



1. Ingest message
2. Deserialize strongly typed command
3. **Retrieve current state from database**
4. **Execute command**
5. Commit output event

Events represent the changed state

Command
`ReserveInventory`

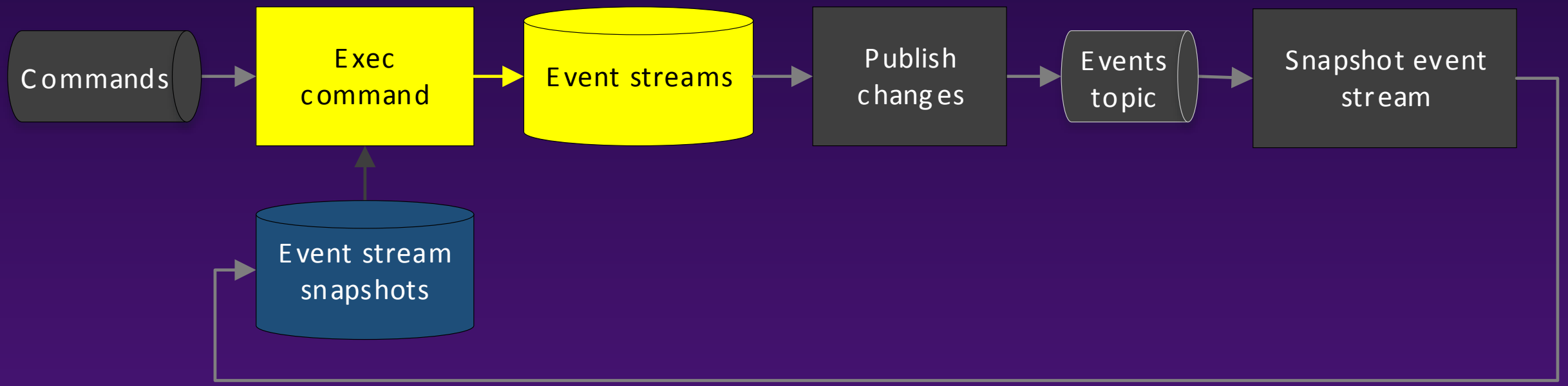
Event
`InventoryReserved`

Validation failures still produce events

Command
`ReserveInventory`

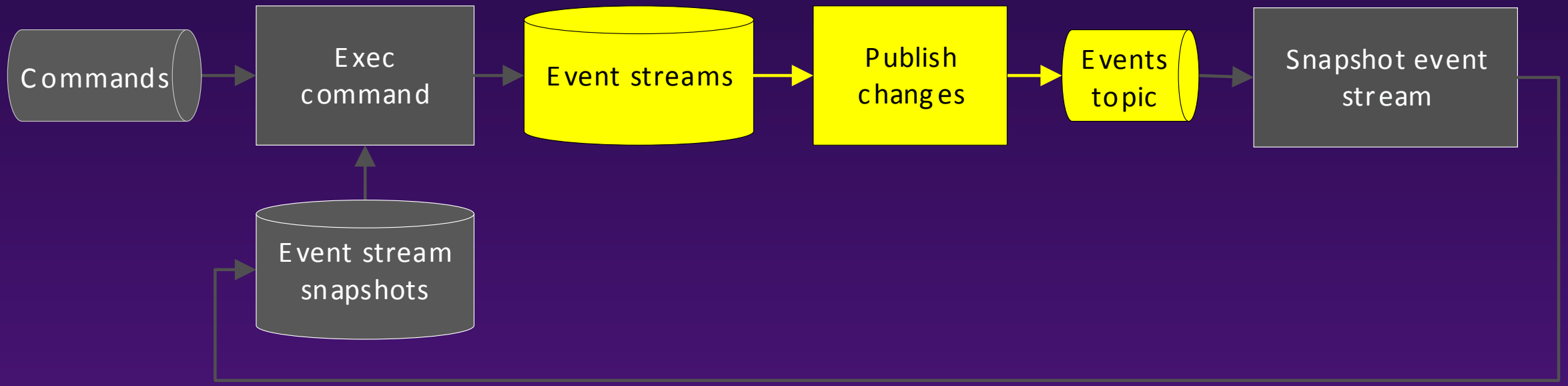
Validation Failure Event
`ReserveInventoryFailed`

Core flow – executing commands, producing events

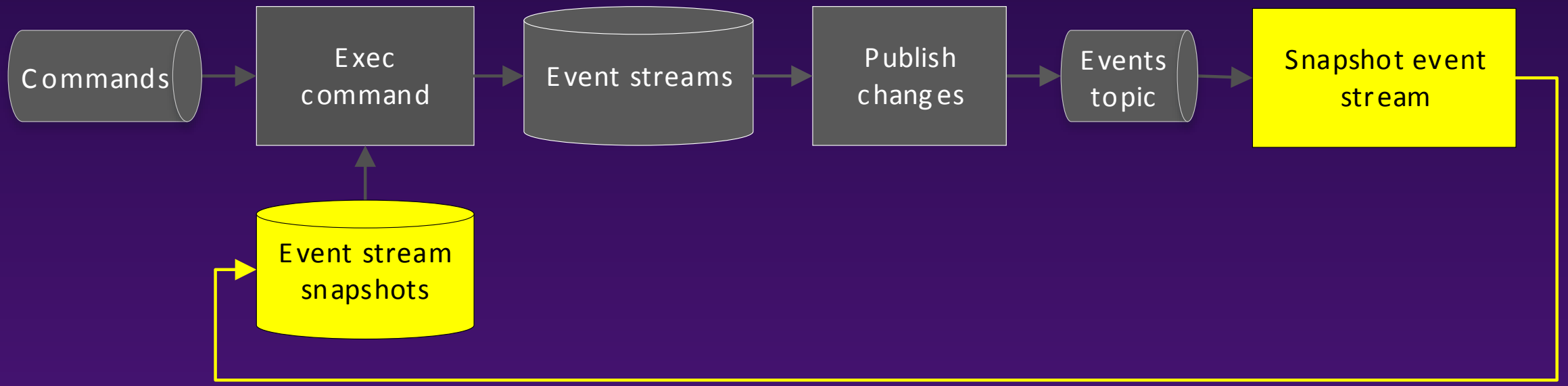


1. Ingest message
2. Deserialize strongly typed command
3. Retrieve current state from database
4. Execute command
5. **Commit output event**

Core flow – publishing changes via Change Data Capture

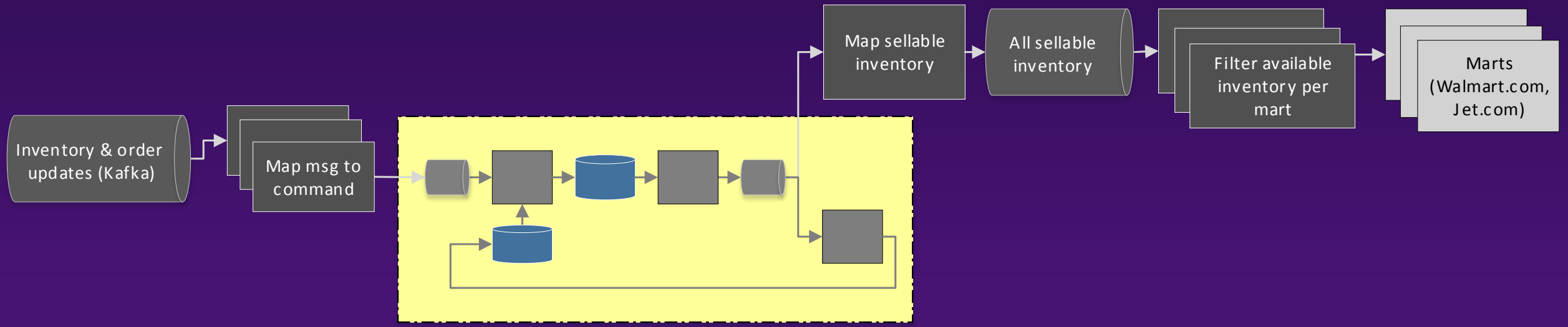


Core flow – building state from events



1. Receive batch of events for stream
2. Get latest snapshot for stream
3. Apply sequence of events to snapshot
4. Save up-to-date snapshot

Panther architecture



Benefits

Resiliency

Resiliency

Out of order messages

Resiliency

Out of order messages

Downstream consumers lose data

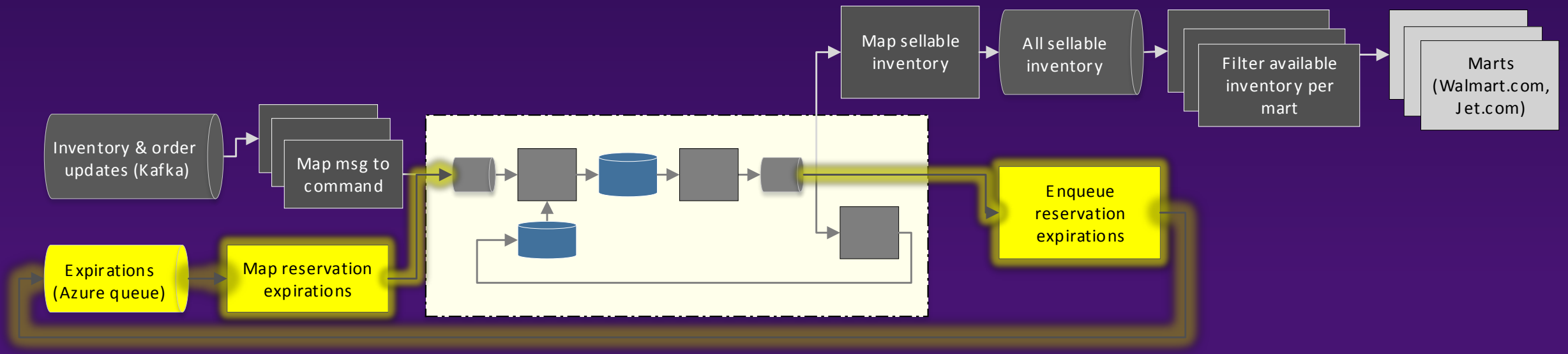
Resiliency

Out of order messages

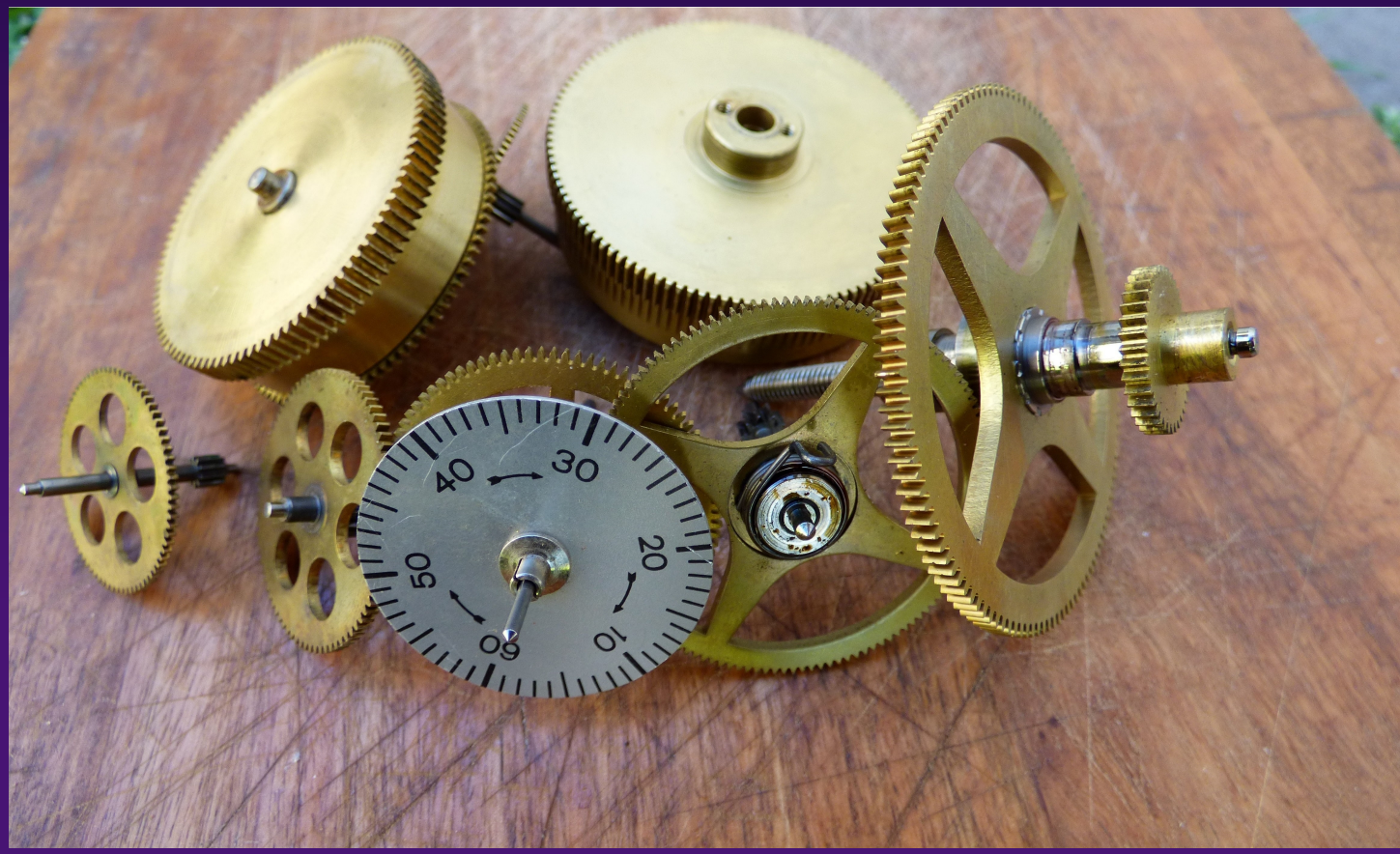
Downstream consumers lose data

Random restarts

Easy to add features



Time machine



Scaling

Testability

Operational costs

Operational costs

What affects item availability?

Operational costs

What affects item availability?

- Warehouse inventory
- Reservations & orders

Operational costs

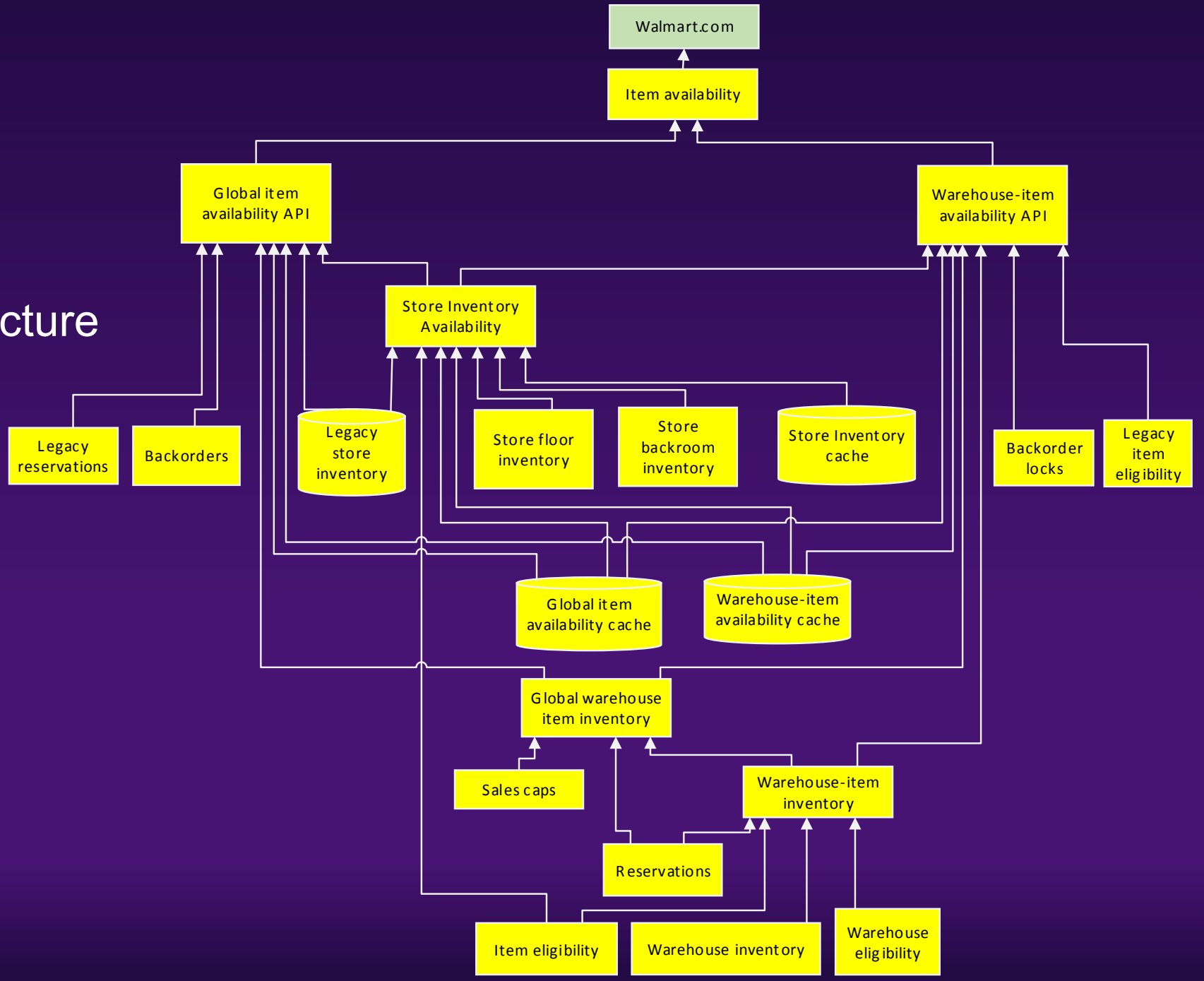
What affects item availability?

- Warehouse inventory
- Reservations & orders
- Store floor inventory
- Store backroom inventory
- Item eligibility
- Warehouse eligibility
- Sales caps
- Backorders
- Legacy systems

SLA: **99.98%** uptime @ **300ms**

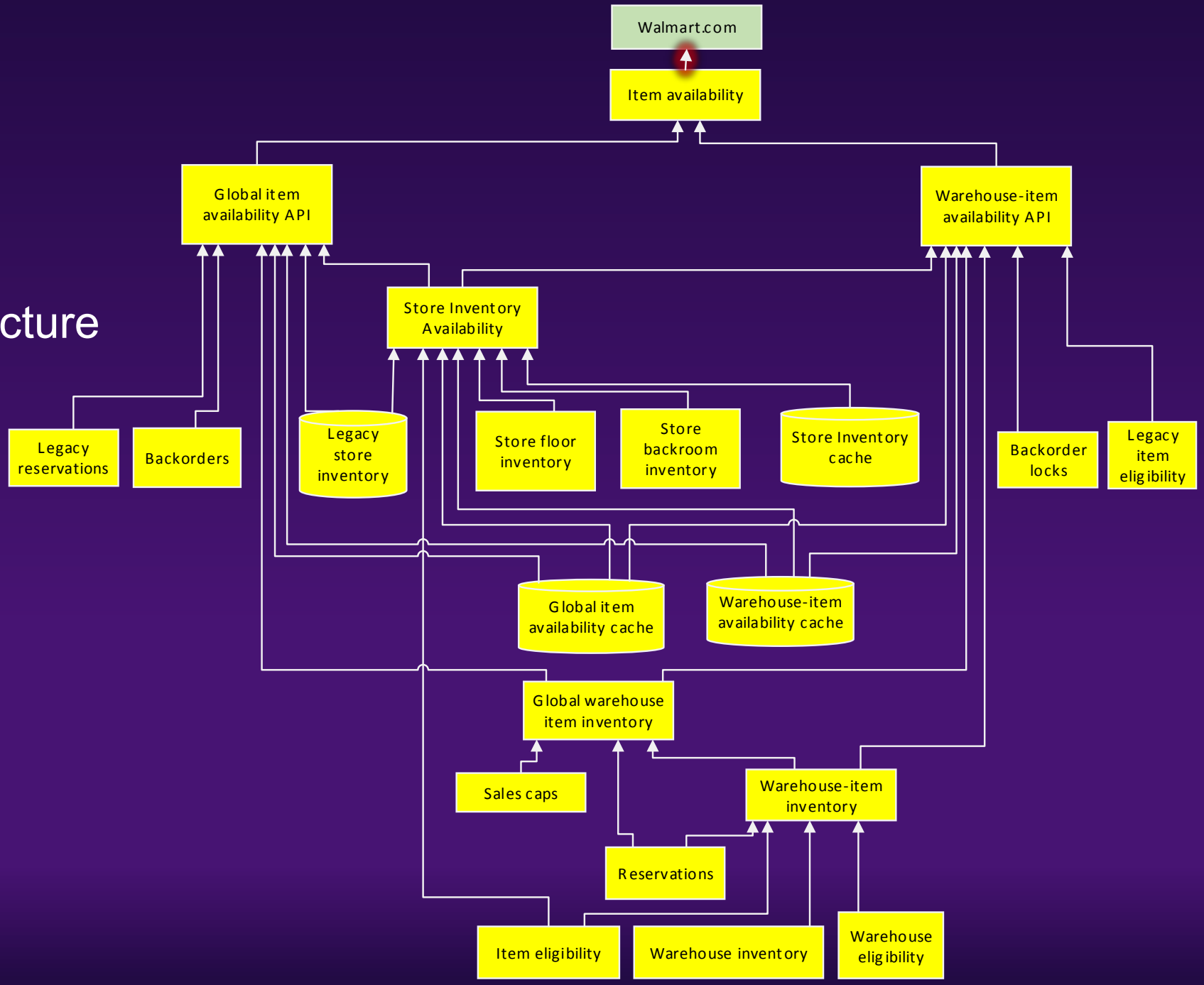
Operational costs

Item availability via
Service-oriented architecture



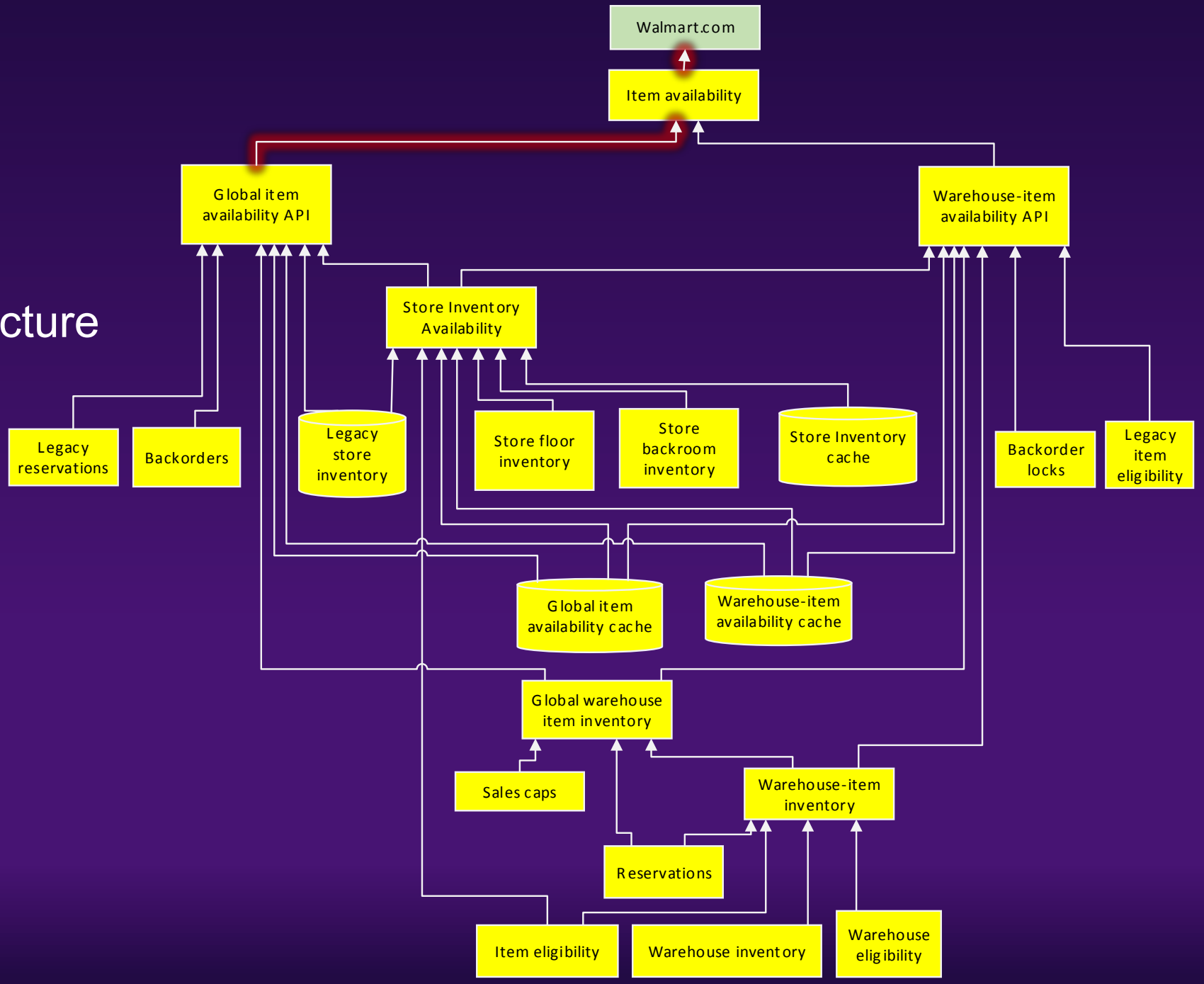
Operational costs

Item availability via
Service-oriented architecture



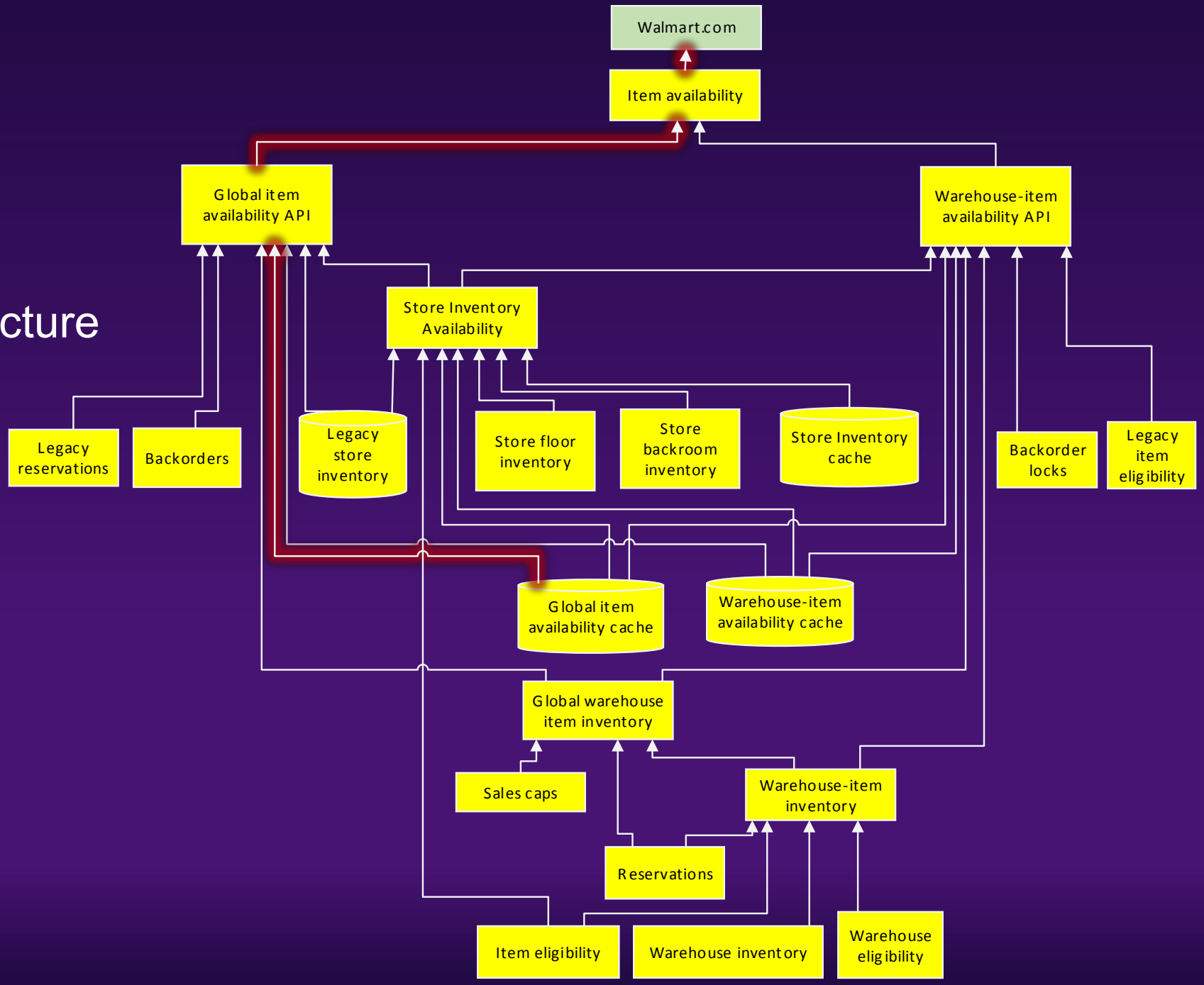
Operational costs

Item availability via
Service-oriented architecture



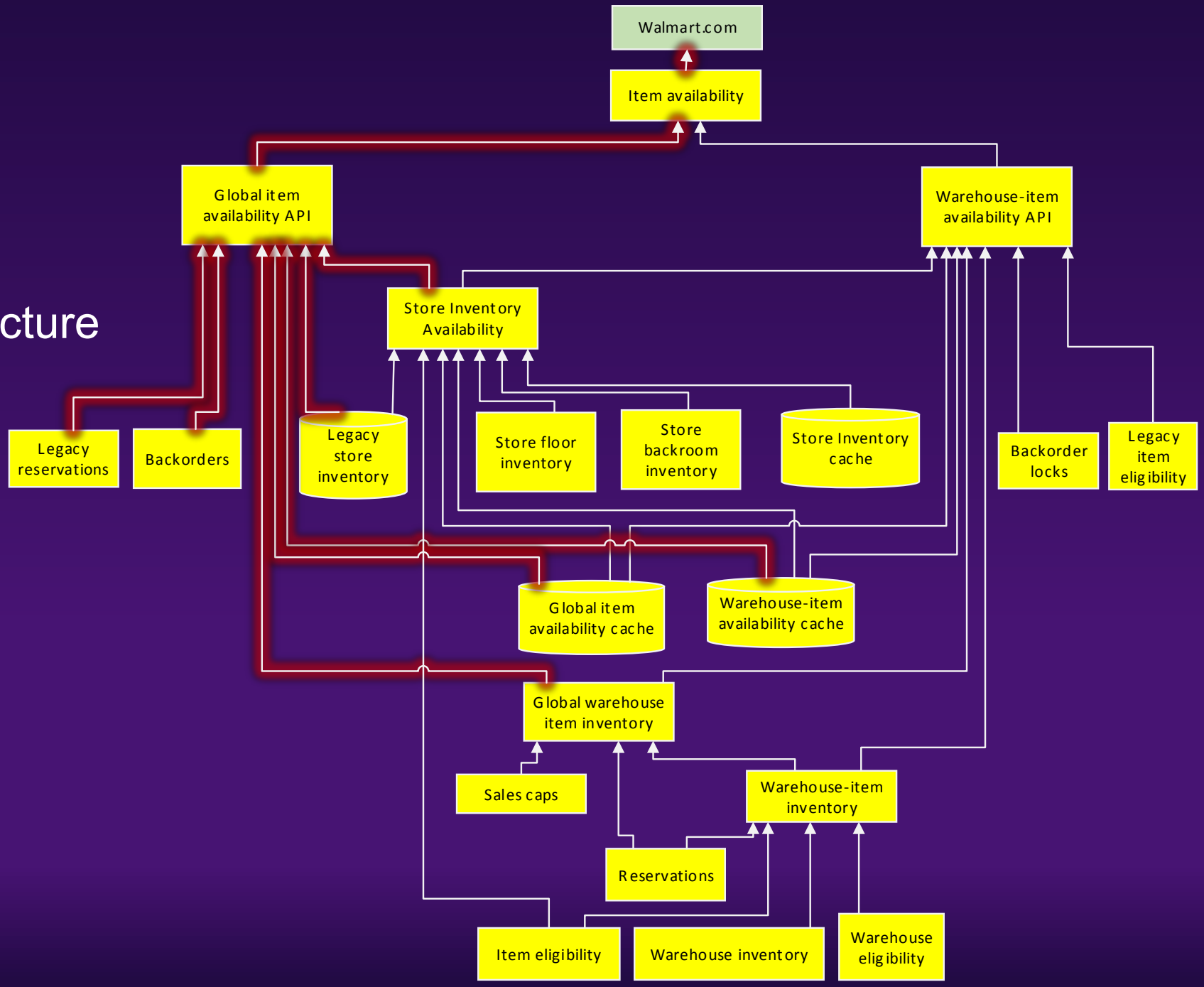
Operational costs

Item availability via
Service-oriented architecture



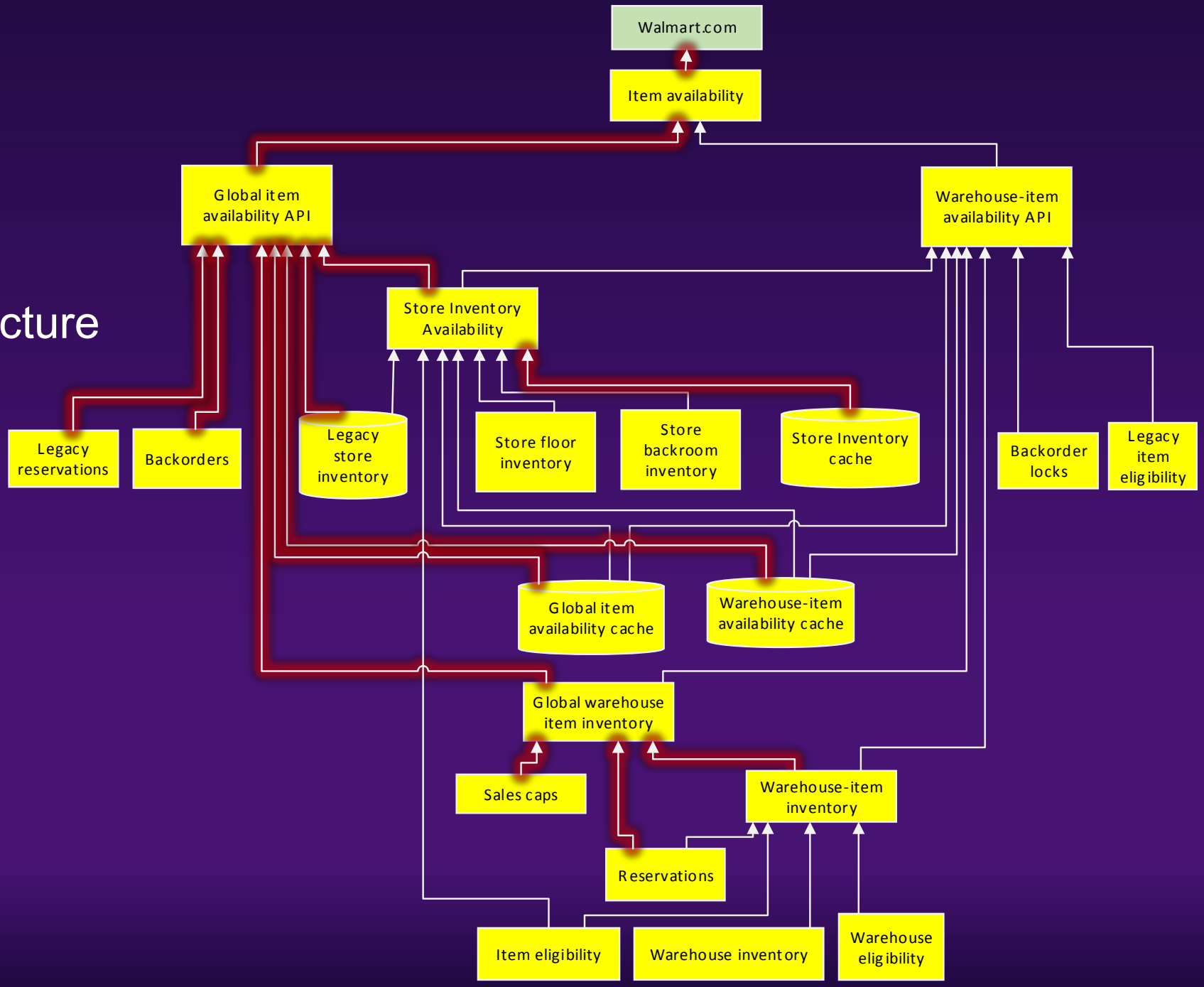
Operational costs

Item availability via
Service-oriented architecture



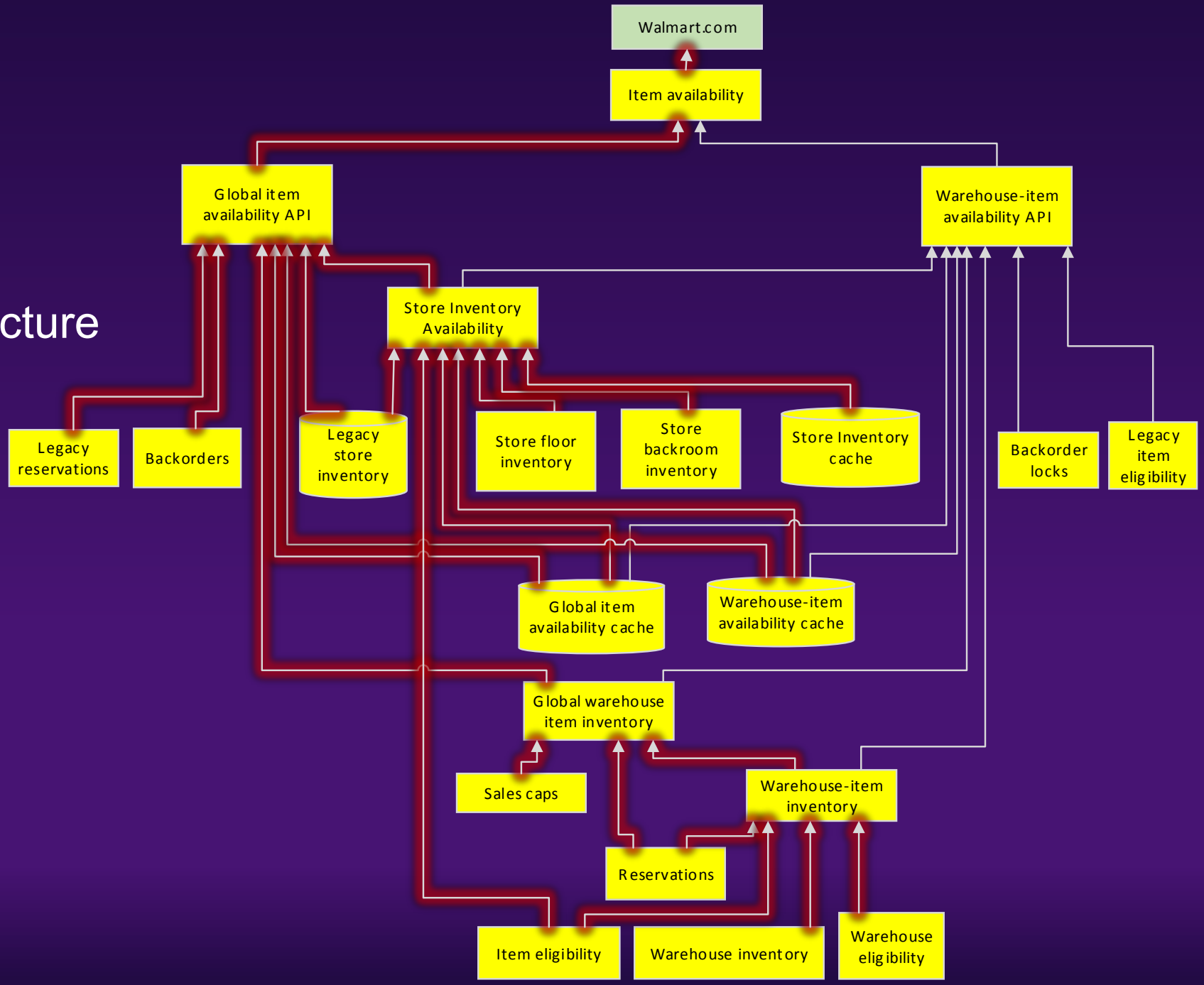
Operational costs

Item availability via
Service-oriented architecture



Operational costs

Item availability via
Service-oriented architecture



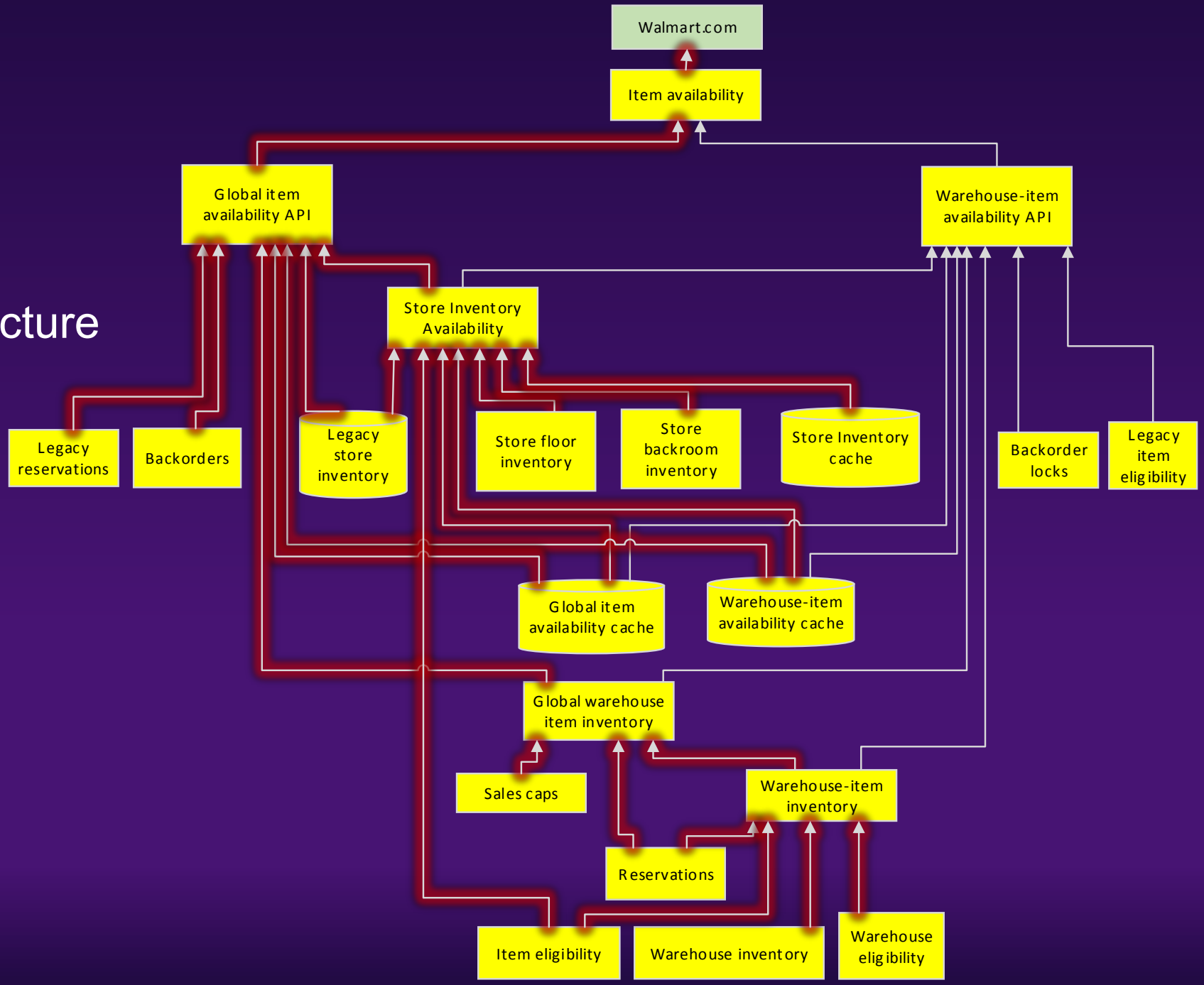
Operational costs

Item availability via
Service-oriented architecture

SLA: **99.98%** uptime **@ 300ms**

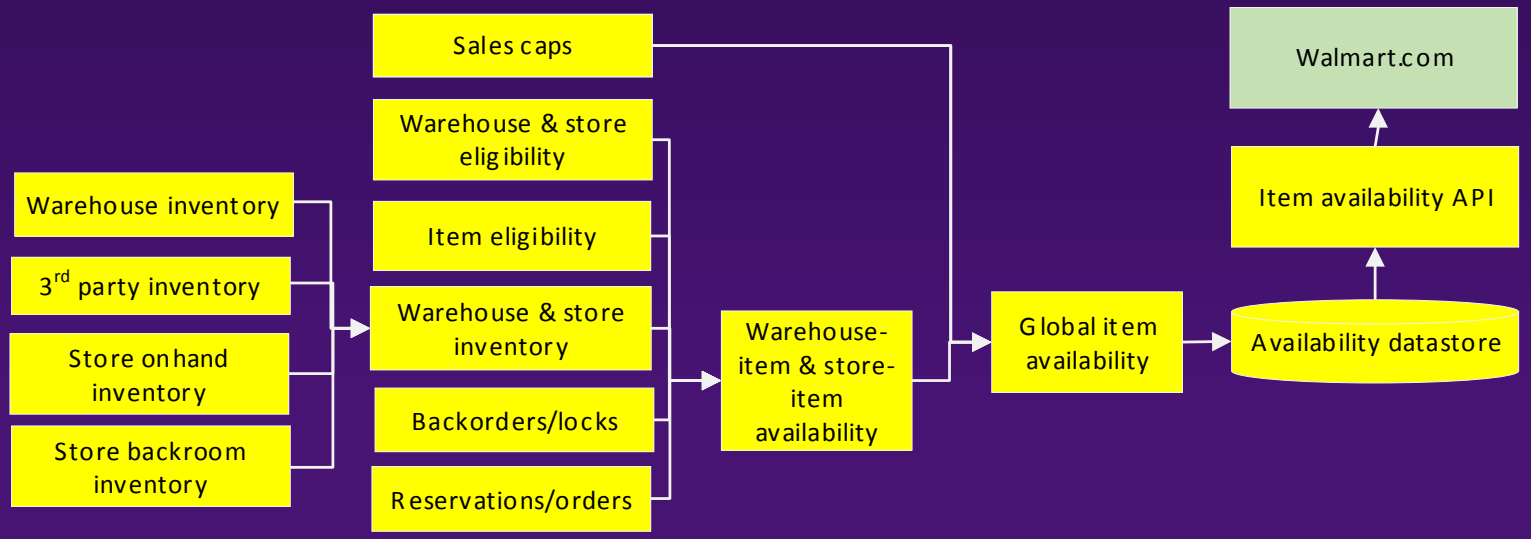
requires

23 service calls
99.999% uptime
@ 50ms mean processing SLO



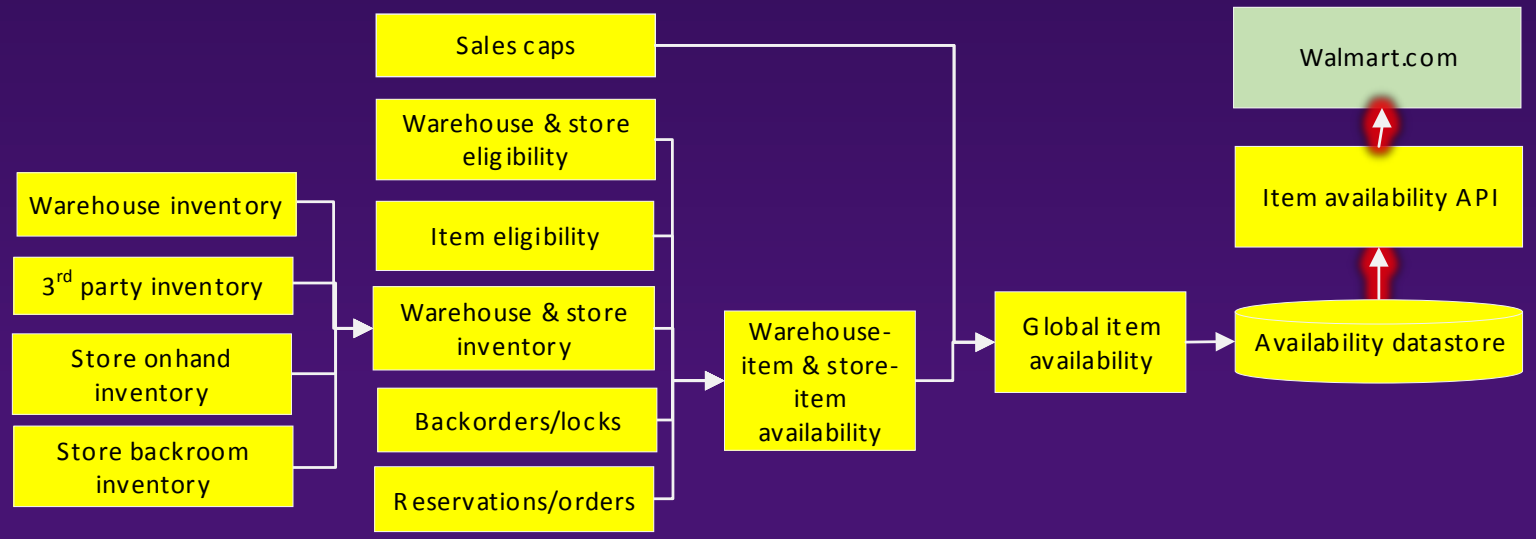
Operational costs

Item availability via
Event-driven architecture



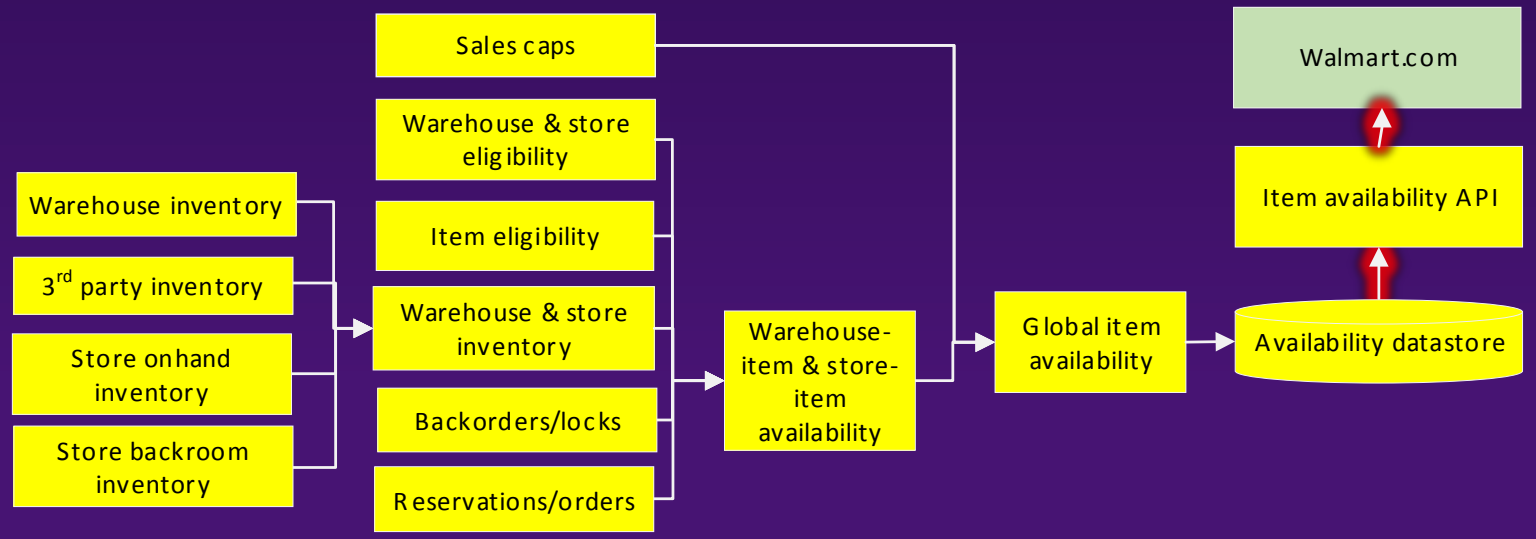
Operational costs

Item availability via
Event-driven architecture



Operational costs

Item availability via
Event-driven architecture



SLA: **99.98%** uptime **@ 300ms**

requires

2 service calls
99.99% uptime
@ 150ms mean processing SLO

Operational costs

Service uptime	Processing latency	Operational cost
99.9%	1 min	$

Operational costs

Service uptime	Processing latency	Operational cost
99.9%	1 min	$
99.99%	150 ms	

Operational costs

Service uptime	Processing latency	Operational cost
99.9%	1 min	$
99.99%	150 ms	$$$$$$$$$$

Operational costs

Service uptime	Processing latency	Operational cost
99.9%	1 min	$
99.99%	150 ms	$$$$$$$$$$
99.999%	50 ms	

Operational costs

Service uptime	Processing latency	Operational cost
99.9%	1 min	$
99.99%	150 ms	$$$$$$$$$$
99.999%	50 ms	$$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$$$ $$$$$$$$$$$$$$$

Operational costs

Item availability via
Event-driven architecture

Item availability via
Service-oriented architecture

Operational costs

Item availability via
Event-driven architecture

$$$$$$$

Item availability via
Service-oriented architecture

Operational costs

Item availability via
Event-driven architecture

$$$$$$$

Item availability via
Service-oriented architecture


Jet

What can you do?

What can you do?

Eliminate one dual write

What can you do?

Eliminate one dual write

Switch one web service to also publish events

What can you do?

Eliminate one dual write

Switch one web service to also publish events

Try property based testing

What can you do?

Eliminate one dual write

Switch one web service to also publish events

Try property based testing

Contact me!

scott.havens@jet.com

@ScottHavens