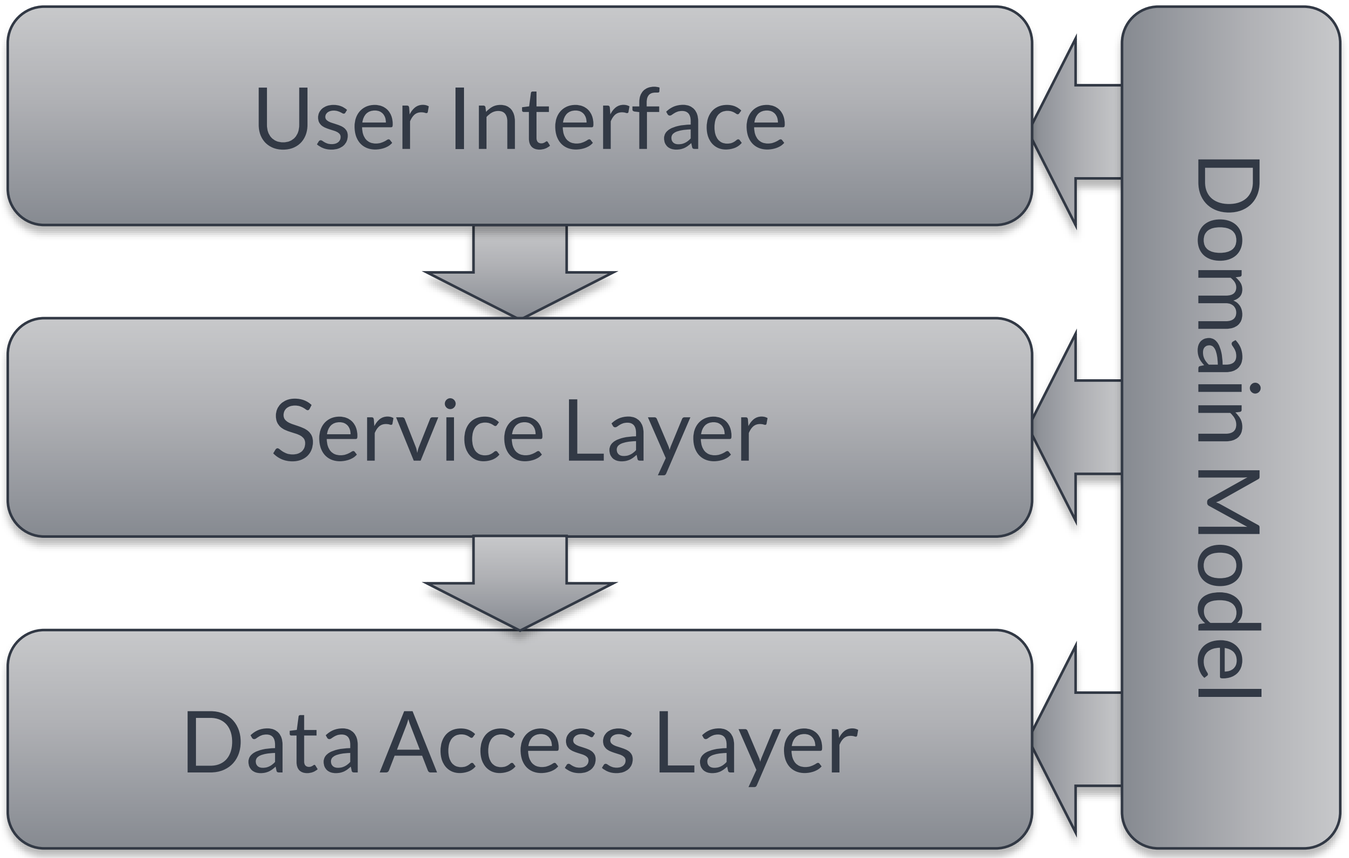
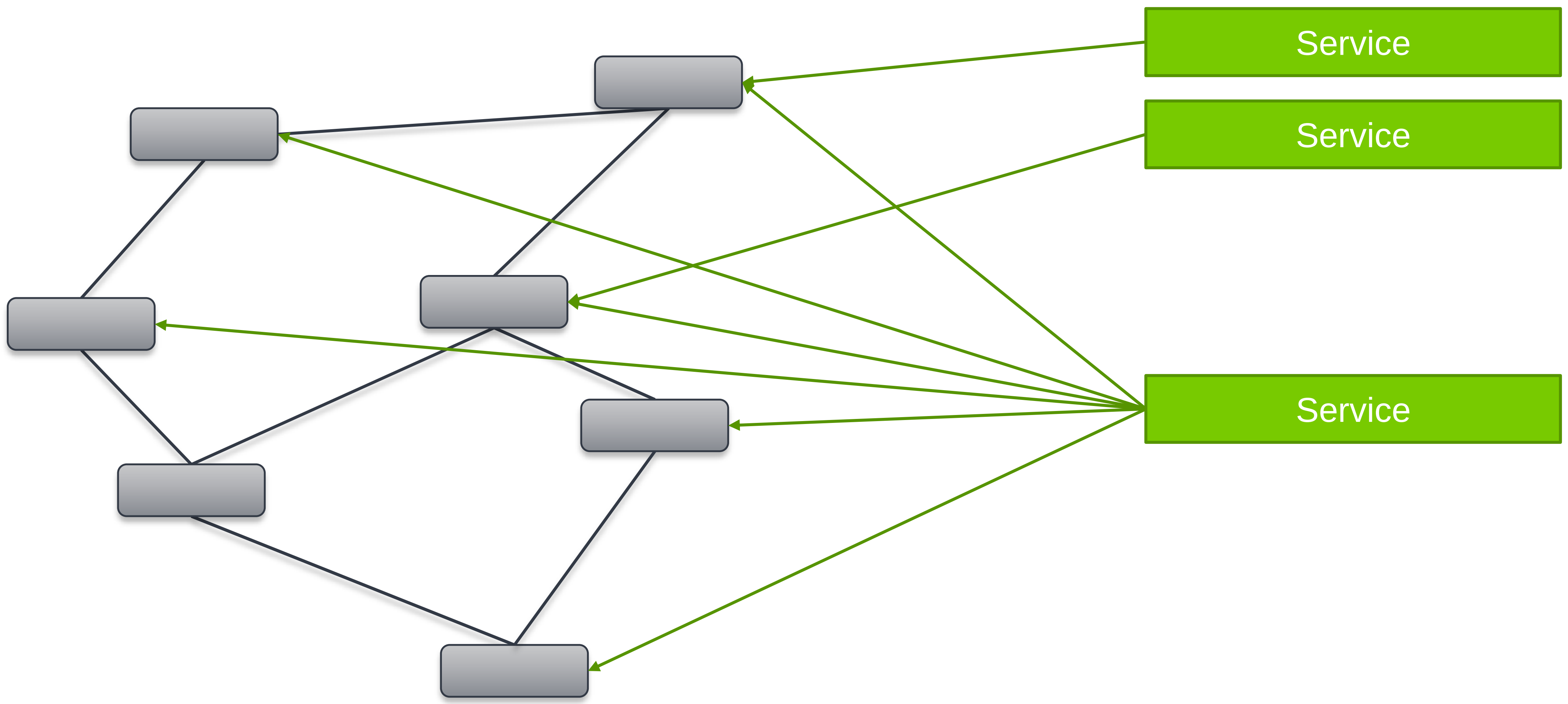
Performance aspects of Axon-based CQRS/ES systems

Allard Buijze
Founder & CTO, AxonIQ
Creator of AxonFramework

✉ allard@axoniq.io
@allardbz

Layered architecture




Service
Service
Service

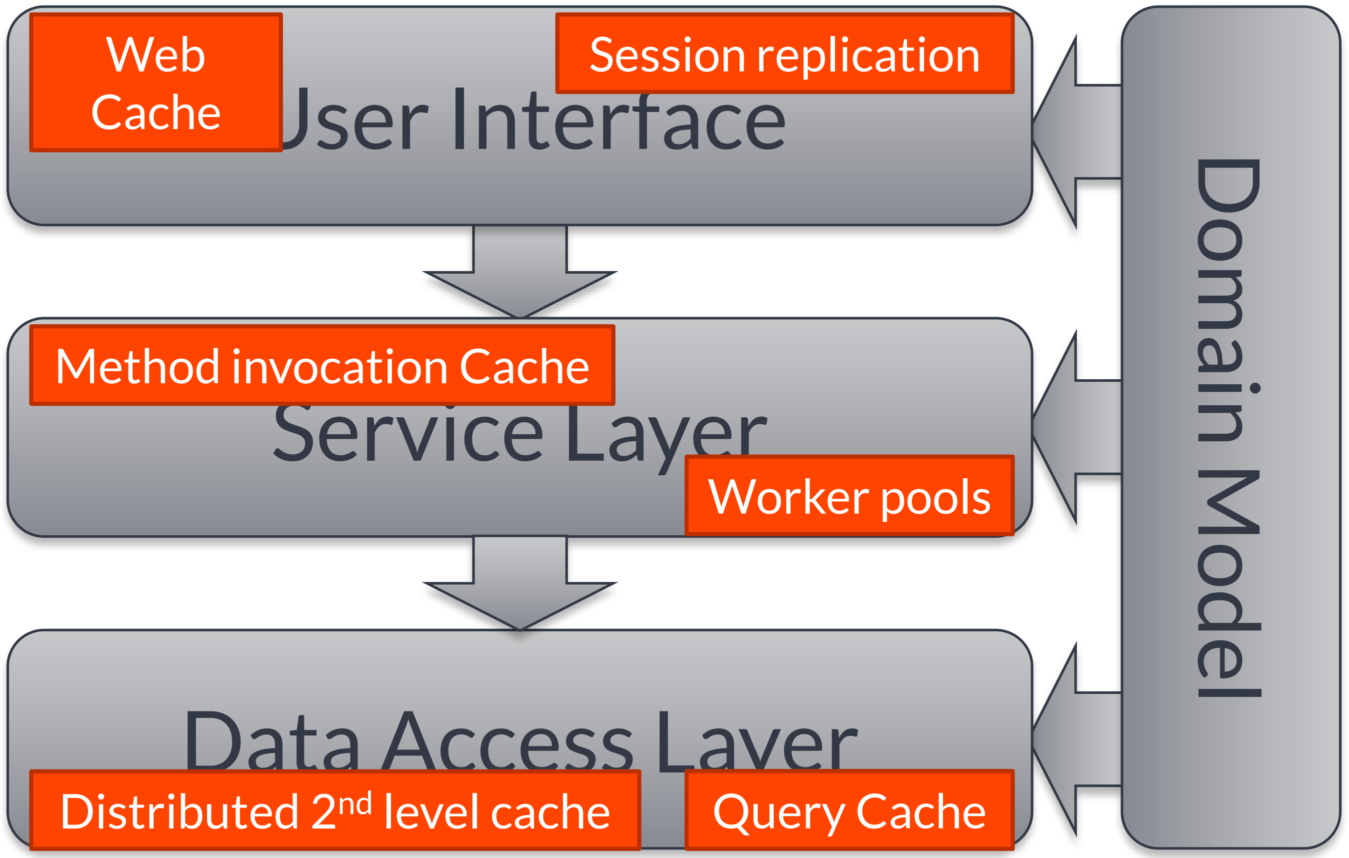
‘Normal’ SQL QUERY

```
CREATE ALGORITHM = UNDEFINED DEFINER = `dbat`@`%` SQL SECURITY DEFINERVIEW `BUSINESS_PROCESS_VIEW_DISABLED` AS SELECT `tpo`.`NAME` AS `TRADING_PARTNER_NAME`, `bt`.`NAME` AS `TRADING_PARTNER_BOARDING_TYPE`, `rto`.`NAME` AS `TRADING_PARTNER_ROOT_NAME`,
`rteid`.`VALUE` AS `TRADING_PARTNER_ROOT_EID`, `tpot`.`ORG_SUB_TYPE_NAME` AS `TRADING_PARTNER_CUSTOMER_TYPE`, `tpo`.`STATUS` AS `TRADING_PARTNER_STATUS`, `tpot`.`NAME` AS `TRADING_PARTNER_TYPE`, `teid`.`VALUE` AS `TRADING_PARTNER_EID`, `tduns`.`VALUE` AS
`TRADING_PARTNER_DUNS`, `tgln`.`VALUE` AS `TRADING_PARTNER_GLN`, `bp`.`BUSINESS_PROCESS` AS `BUSINESS_PROCESS`, (CASE CONCAT((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`), (`myo`.`ORGANIZATION_ID` = `rel`.`PARTY2_ID`), `bp`.`PARTY1_DIRECTION`) WHEN '10TO'
THEN 'TO' WHEN '10FROM' THEN 'FROM' WHEN '11TO' THEN 'TO' WHEN '01TO' THEN 'FROM' WHEN '01FROM' THEN 'TO' END) AS `DOCUMENT_DIRECTION`, `bp`.`STATUS` AS `STATUS`, `idt`.`DOC_TYPE` AS `INBOUND_TYPE`, `idt`.`DOC_SUB_TYPE` AS `INBOUND_SUB_TYPE`,
`idt`.`DOC_VERSION` AS `INBOUND_VERSION`, `iodt`.`DOC_DIRECTION` AS `INBOUND_DIRECTION`, `odt`.`DOC_TYPE` AS `OUTBOUND_TYPE`, `odt`.`DOC_SUB_TYPE` AS `OUTBOUND_SUB_TYPE`, `odt`.`DOC_VERSION` AS `OUTBOUND_VERSION`, `oodt`.`DOC_DIRECTION` AS
`OUTBOUND_DIRECTION`, `bp`.`GO_LIVE_DT` AS `GO_LIVE_DATE`, `bp`.`BUSINESS_PROCESS_ID` AS `ID`, `myo`.`ORGANIZATION_ID` AS `ORG_ID`, `myo`.`NAME` AS `ORG_NAME`, `meid`.`VALUE` AS `ORG_EID`, `ro`.`ORGANIZATION_ID` AS `ROOT_ORG_ID`, `ro`.`NAME` AS
`ROOT_ORG_NAME`, `reid`.`VALUE` AS `ROOT_ORG_EID`, `idt`.`DOC_TYPE_ID` AS `INBOUND_DOC_TYPE_ID`, `odt`.`DOC_TYPE_ID` AS `OUTBOUND_DOC_TYPE_ID`, `rel`.`RELATIONSHIP_ID` AS `RELATIONSHIP_ID`, `oop`.`ISA05` AS `OB_ENV_SNDR_QUAL`, `oop`.`ISA06` AS
`OB_ENV_SNDR_ID`, `oop`.`ISA07` AS `OB_ENV_RCVR_QUAL`, `oop`.`ISA08` AS `OB_ENV_RCVR_ID`, `oop`.`GS02` AS `OB_IN_SNDR_ID`, `oop`.`GS03` AS `OB_IN_RCVR_ID`, `iop`.`ISA05` AS `IB_ENV_SNDR_QUAL`, `iop`.`ISA06` AS `IB_ENV_SNDR_ID`, `iop`.`ISA07` AS
`IB_ENV_RCVR_QUAL`, `iop`.`ISA08` AS `IB_ENV_RCVR_ID`, `iop`.`GS02` AS `IB_IN_SNDR_ID`, `iop`.`GS03` AS `IB_IN_RCVR_ID`, `bpop`.`IB_OVR_ENV_RCVR_QUAL` AS `IB_OVR_ENV_RCVR_QUAL`, `bpop`.`IB_OVR_ENV_RCVR_ID` AS `IB_OVR_ENV_RCVR_ID`,
`bpop`.`IB_OVR_ENV_SNDR_QUAL` AS `IB_OVR_ENV_SNDR_QUAL`, `bpop`.`IB_OVR_ENV_SNDR_ID` AS `IB_OVR_ENV_SNDR_ID`, `bpop`.`IB_OVR_INNER_ENV_RCVR_ID` AS `IB_OVR_INNER_RCVR_ID`, `bpop`.`IB_OVR_INNER_ENV_SNDR_ID` AS `IB_OVR_INNER_SNDR_ID`,
`bpop`.`OB_OVR_ENV_RCVR_QUAL` AS `OB_OVR_ENV_RCVR_QUAL`, `bpop`.`OB_OVR_ENV_RCVR_ID` AS `OB_OVR_ENV_RCVR_ID`, `bpop`.`OB_OVR_ENV_SNDR_QUAL` AS `OB_OVR_ENV_SNDR_QUAL`, `bpop`.`OB_OVR_ENV_SNDR_ID` AS `OB_OVR_ENV_SNDR_ID`, `bpop`.`OB_OVR_INNER_ENV_RCVR_ID`
AS `OB_OVR_INNER_RCVR_ID`, `bpop`.`OB_OVR_INNER_ENV [illegible] AS `IB_TRX_ENV_RCVR_Q [illegible] `IB_TRX_ENV_SNDR_QUAL` AS `IB_TRX_ENV_SNDR_QUAL`,
`bpop`.`IB_TRX_ENV_SNDR_ID` AS `IB_TRX_ENV_SNDR [illegible] `.`IB_TRX_INNER_EN [illegible] `OB_TRX_ENV_RCVR_QUAL`,
`bpop`.`OB_TRX_ENV_RCVR_ID` AS `OB_TRX_ENV_RCVR [illegible] B_TRX_ENV_SNDR_ID [illegible] _INNER_RCVR_ID`,
`bpop`.`OB_TRX_INNER_ENV_SNDR_ID` AS `OB_TRX_INN [illegible] `rel`.`XSBL_ID` [illegible] LATIONSHIP` `rel` ON ((`rel`.`RELATIONSHIP_ID`
= `bp`.`RELATIONSHIP_ID`))) LEFT JOIN `ORGANIZAT [illegible] .`ORGANIZATION_ID [illegible] _ID` = `ro`.`ORGANIZATION_ID`))) LEFT JOIN
`ORGANIZATION` `tpo` ON (((`tpo`.`ORGANIZATION_I [illegible] )))) LEFT JOIN `OR [illegible] ANIZATION_ID`))) LEFT JOIN `BOARDING_TYPE` `bt`
ON ((`bta`.`BOARDING_TYPE_ID` = `bt`.`BOARDING_T [illegible] `ORGANIZATION_ID`) [illegible] = `rto`.`ORGANIZATION_ID`) AND
(`rteid`.`PRIMARY_FLAG` = 1) AND (`rteid`.`ORG_I [illegible] TYPE_ID` FROM `DIM [illegible] )))) LEFT JOIN `ORG_TYPE_ASSOC` `tpoa` ON
((`tpoa`.`ORGANIZATION_ID` = `tpo`.`ORGANIZATION [illegible] oa`.`ORG_TYPE_ID`) [illegible] myo`.`ORGANIZATION_ID`) AND
(`meid`.`PRIMARY_FLAG` = 1) AND (`meid`.`ORG_IDE [illegible] PE_ID` FROM `DIM_C [illegible] )) LEFT JOIN `ORG_IDENTIFIER` `teid` ON
(((`teid`.`ORGANIZATION_ID` = `tpo`.`ORGANIZATIO [illegible] _ID` = (SELECT `DI [illegible] ER_TYPE` WHERE (`DIM_ORG_IDENTIFIER_TYPE`.`NAME`
= 'EID')))))) LEFT JOIN `ORG_IDENTIFIER` `tduns` [illegible] ns`.`PRIMARY_FLAG` = [illegible] ER_TYPE`.`ORG_IDENTIFIER_TYPE_ID` FROM
`DIM_ORG_IDENTIFIER_TYPE` WHERE (`DIM_ORG_IDENTIFIER_TYPE`.`NAME` = 'DUNS')))))) LEFT JOIN `ORG_IDENTIFIER` `tgln` ON (((`tgln`.`ORGANIZATION_ID` = `tpo`.`ORGANIZATION_ID`) AND (`tgln`.`PRIMARY_FLAG` = 1) AND (`tgln`.`ORG_IDENTIFIER_TYPE_ID` = (SELECT
`DIM_ORG_IDENTIFIER_TYPE`.`ORG_IDENTIFIER_TYPE_ID` FROM `DIM_ORG_IDENTIFIER_TYPE` WHERE (`DIM_ORG_IDENTIFIER_TYPE`.`NAME` = 'GLN')))))) LEFT JOIN `ORG_IDENTIFIER` `reid` ON (((`reid`.`ORGANIZATION_ID` = `ro`.`ORGANIZATION_ID`) AND (`reid`.`PRIMARY_FLAG` =
1) AND (`reid`.`ORG_IDENTIFIER_TYPE_ID` = (SELECT `DIM_ORG_IDENTIFIER_TYPE`.`ORG_IDENTIFIER_TYPE_ID` FROM `DIM_ORG_IDENTIFIER_TYPE` WHERE (`DIM_ORG_IDENTIFIER_TYPE`.`NAME` = 'EID')))))) LEFT JOIN `ORG_DOC_TYPE` `iodt` ON ((((`iodt`.`ORG_DOC_TYPE_ID` =
`bp`.`IB_DOC_TYPE_ID`) AND (((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'FROM')) OR ((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY2_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'TO')))) OR ((`iodt`.`ORG_DOC_TYPE_ID` =
`bp`.`OB_DOC_TYPE_ID`) AND (((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'TO')) OR ((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY2_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'FROM'))))))) LEFT JOIN `DIM_DOC_TYPE` `idt` ON
((`idt`.`DOC_TYPE_ID` = `iodt`.`DOC_TYPE_ID`))) LEFT JOIN `ORG_DOC_TYPE` `oodt` ON ((((`oodt`.`ORG_DOC_TYPE_ID` = `bp`.`IB_DOC_TYPE_ID`) AND (((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'TO')) OR ((`myo`.`ORGANIZATION_ID`
= `rel`.`PARTY2_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'FROM')))) OR ((`oodt`.`ORG_DOC_TYPE_ID` = `bp`.`OB_DOC_TYPE_ID`) AND (((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'FROM')) OR ((`myo`.`ORGANIZATION_ID` =
`rel`.`PARTY2_ID`) AND (`bp`.`PARTY1_DIRECTION` = 'TO'))))))) LEFT JOIN `DIM_DOC_TYPE` `odt` ON ((`odt`.`DOC_TYPE_ID` = `oodt`.`DOC_TYPE_ID`))) LEFT JOIN `BUSINESS_PROCESS_OVERRIDE_PARAM` `bpop` ON ((`bpop`.`BUSINESS_PROCESS_OVERRIDE_PARAM_ID` =
`bp`.`BUSINESS_PROCESS_ID`))) LEFT JOIN `ORG_DOC_TYPE_PARAMETER_BUSINESS_PROCESS_VIEW` `iop` ON ((`iop`.`ID` = `bp`.`IB_DOC_TYPE_ID`))) LEFT JOIN `ORG_DOC_TYPE_PARAMETER_BUSINESS_PROCESS_VIEW` `oop` ON ((`oop`.`ID` = `bp`.`OB_DOC_TYPE_ID`))) WHERE
((((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`) AND (`tpo`.`ORGANIZATION_ID` = `rel`.`PARTY2_ID`)) OR ((`myo`.`ORGANIZATION_ID` = `rel`.`PARTY2_ID`) AND (`tpo`.`ORGANIZATION_ID` = `rel`.`PARTY1_ID`))) AND ISNULL(`bp`.`ORIGIN_BUSINESS_PROCESS_ID`))
```

22 JOINS

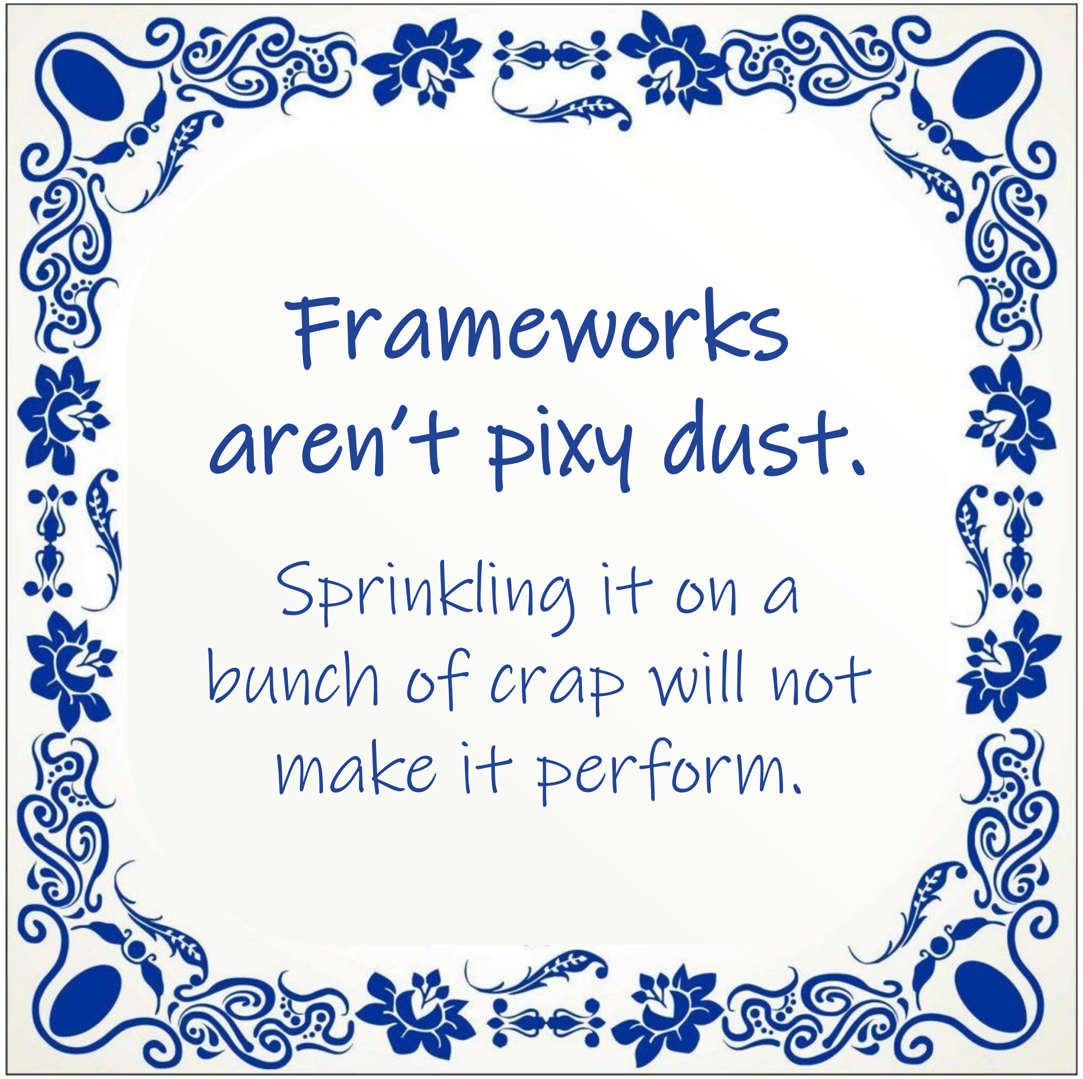
6 SUBQUERIES

Layered architecture



352 KTH
MODIFIEDPLANET.COM

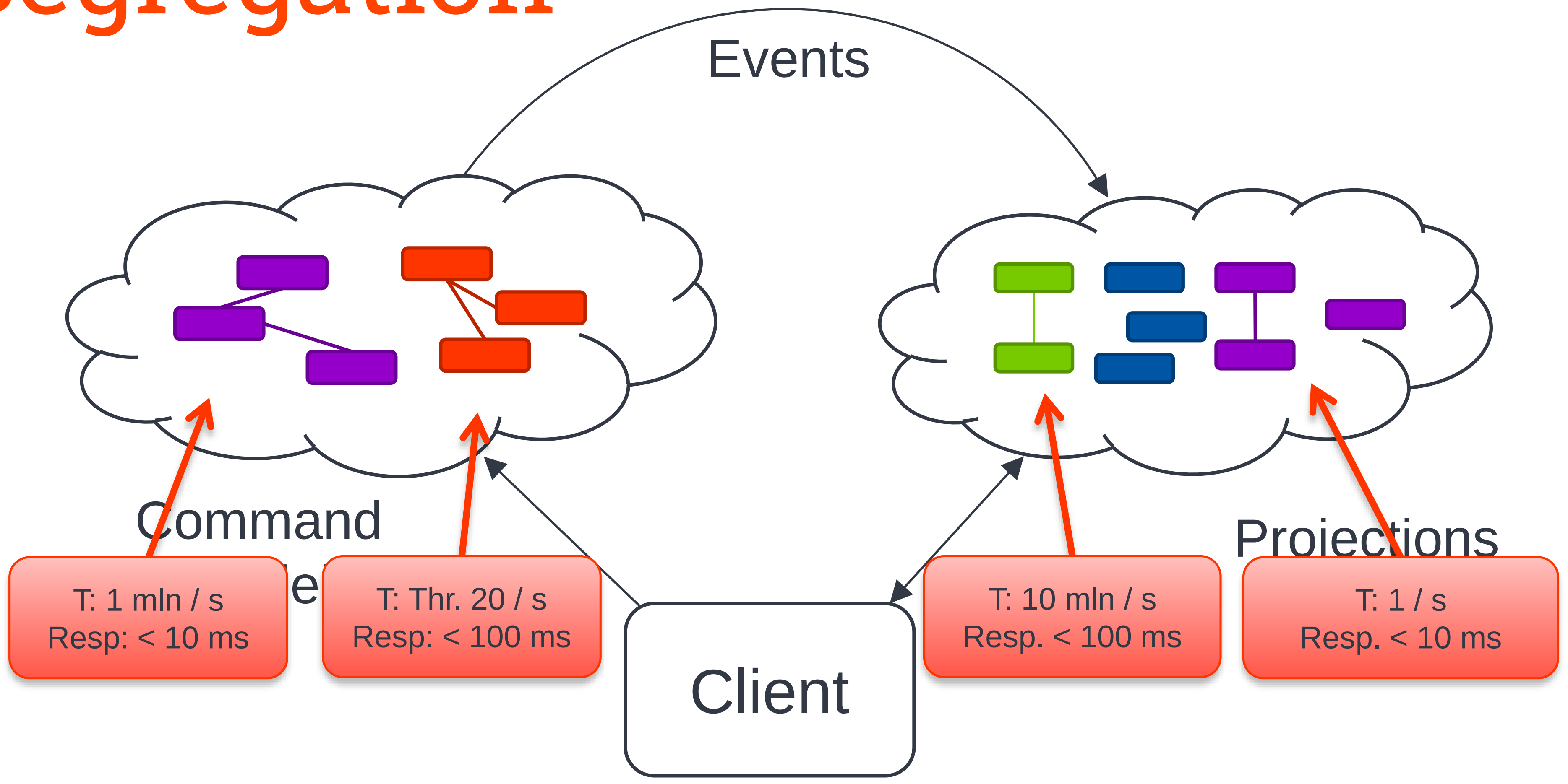
Source: http://www.sabisabi.com/images/DungBeetle-on-dung.JPG



Frameworks aren't pixy dust.

Sprinkling it on a bunch of crap will not make it perform.

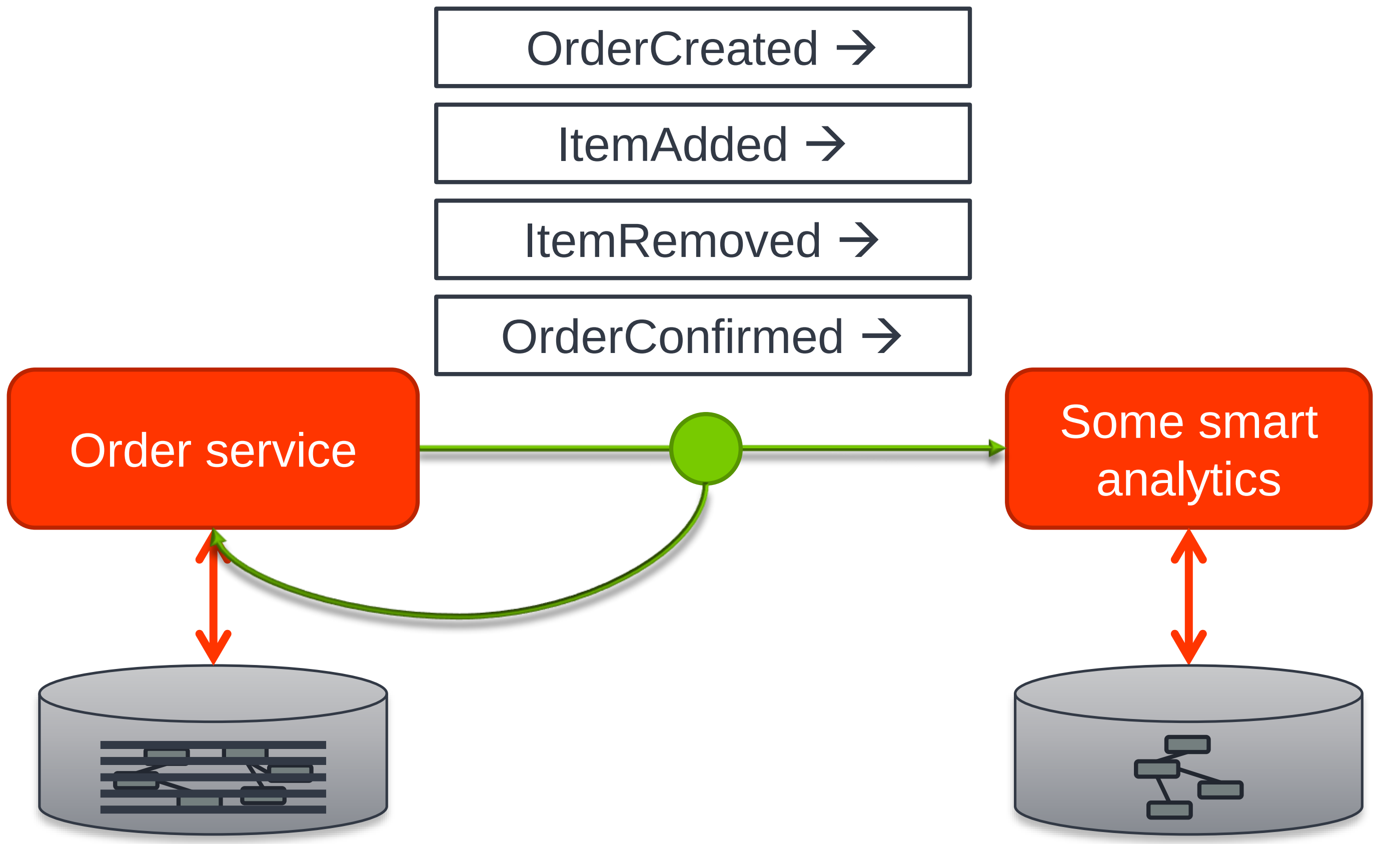
Command Query Responsibility Segregation



Events retain value

Event Sourcing is an Architectural pattern in which Events are considered the “source of truth”, based on which components (re)build their internal state.

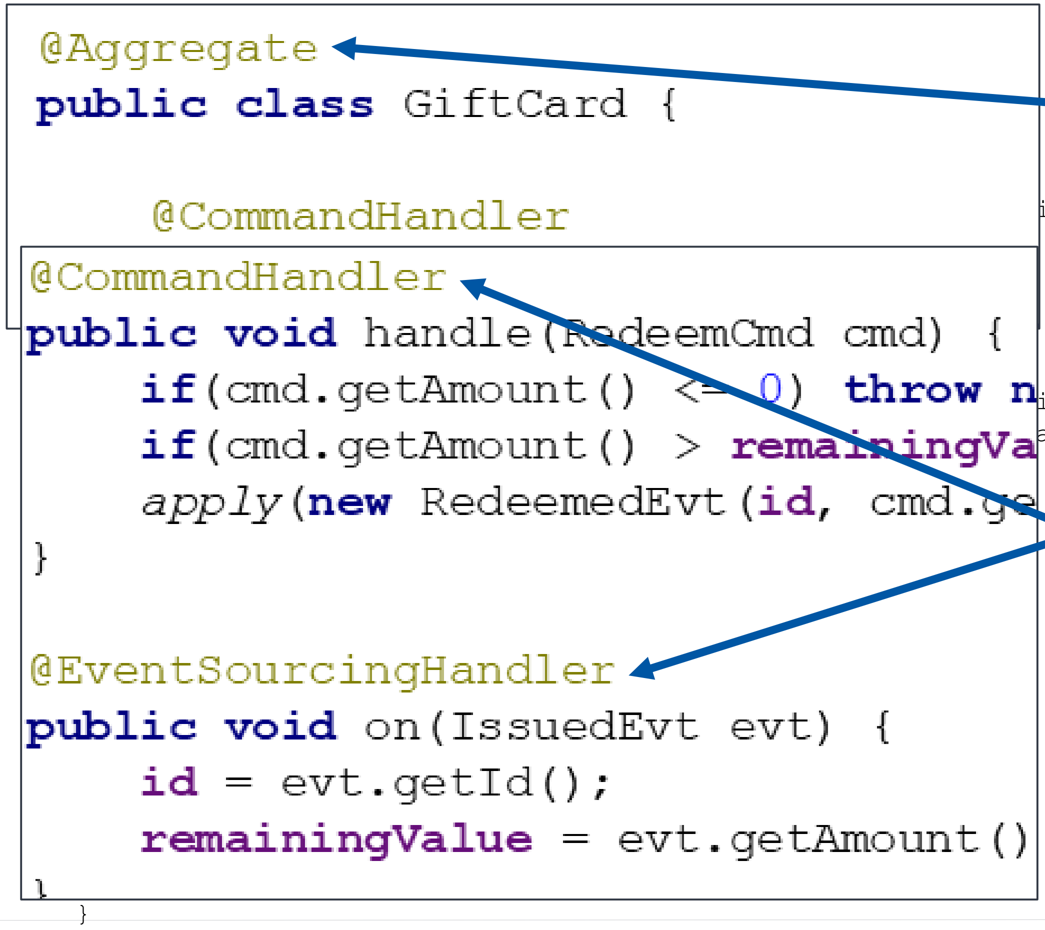
Event Sourcing



Event Store

An Event Store stores the published events to be retrieved both by consumers as well as the publishing component itself.

Event sourcing, made easy



```
@Aggregate
public class GiftCard {

    @CommandHandler
```

```
ion("amount <= 0");
```

```
@CommandHandler
public void handle(RedeemCmd cmd) {
    if(cmd.getAmount() <= 0) throw n
    if(cmd.getAmount() > remainingVa
    apply(new RedeemedEvt(id, cmd.ge
}

@EventSourcingHandler
public void on(IssuedEvt evt) {
    id = evt.getId();
    remainingValue = evt.getAmount()
}
    }
```

```
ion("amount <= 0");
ateException("amount > remaining value");
```

Some annotation to discover the handling class

Some annotations to have correct methods triggered…

Framework does the rest…

Event handling, made easy

```
@Component
public class CardSummaryProjection {

    private final EntityManager entityManager;
    private final QueryUpdateEmitter queryUpdateEmitter;

    @EventHandler
    public void on(IssuedEvt event) {
        entityManager.persist(new Car                                 vent.getAmount(),
    }

    @EventHandler
    public void on(RedeemedEvt event)
        CardSummary summary = entityMa                                lass, event.getId(
        summary.setRemainingValue(summary.getRemainingValue() - event.getAmount(
    }
}
```

Some annotation to discover the handling class

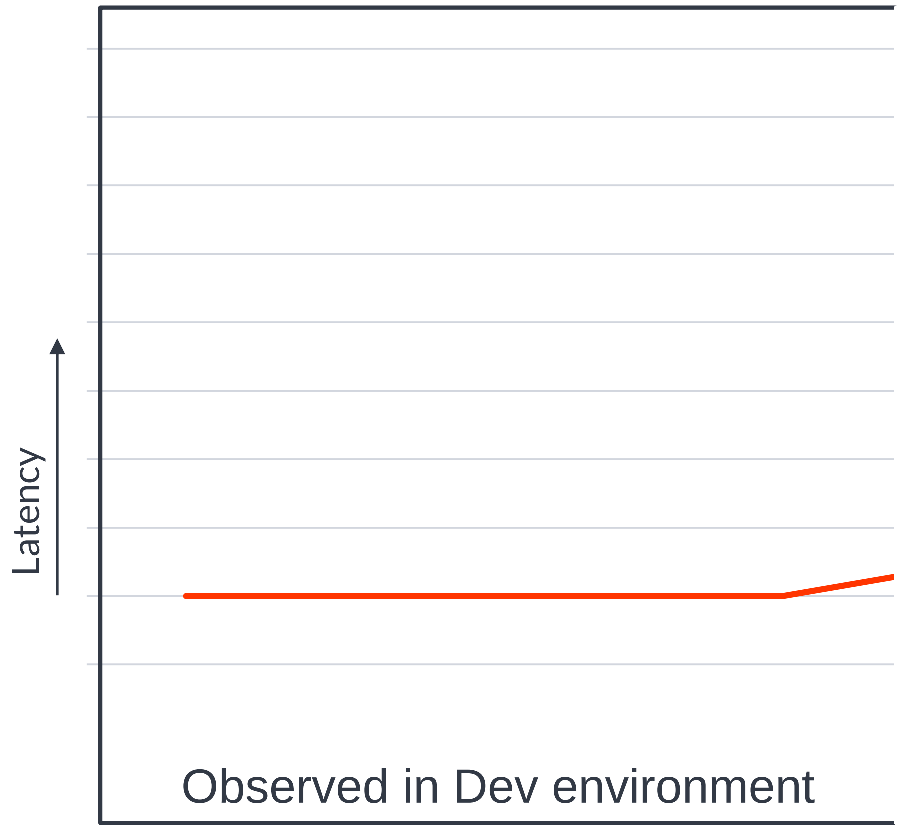
Some annotations to have correct methods triggered…

Framework does the rest…

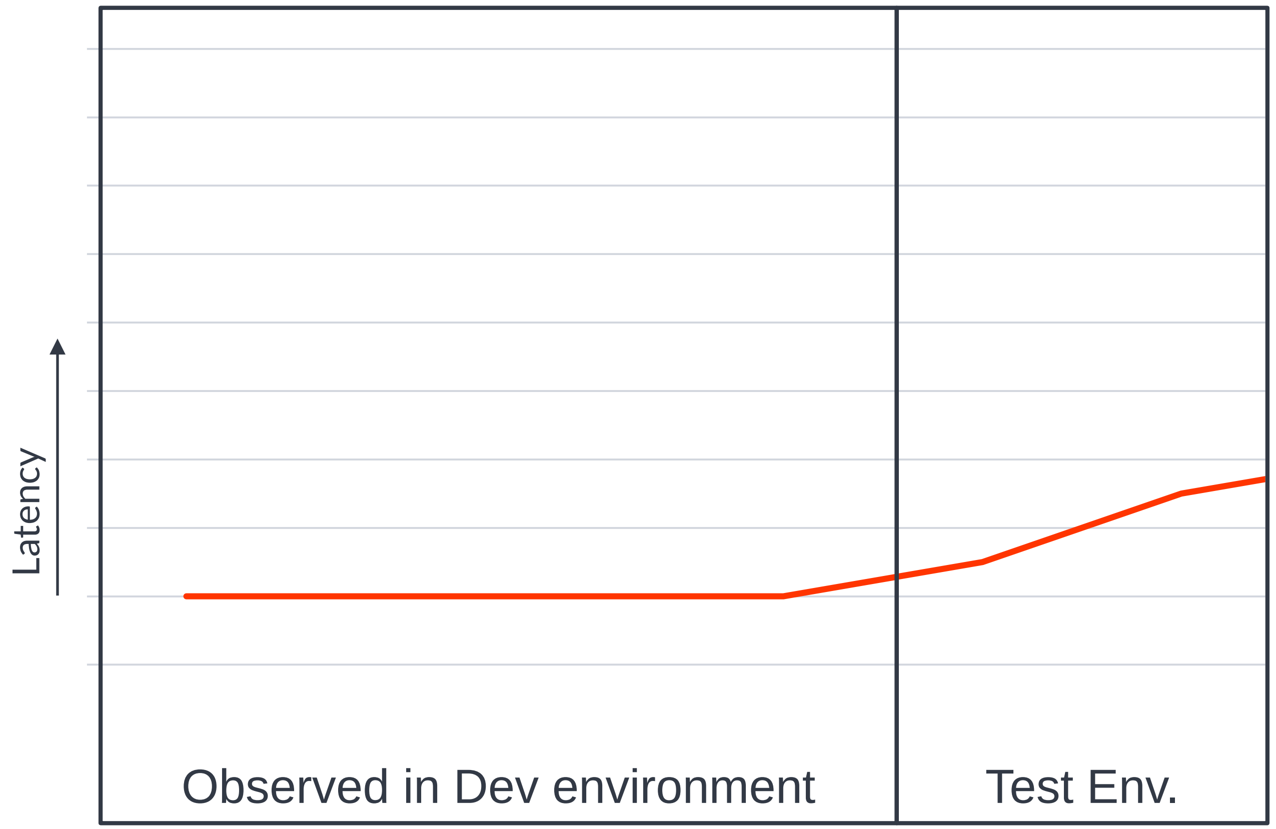
AxonIQ

@allardbz

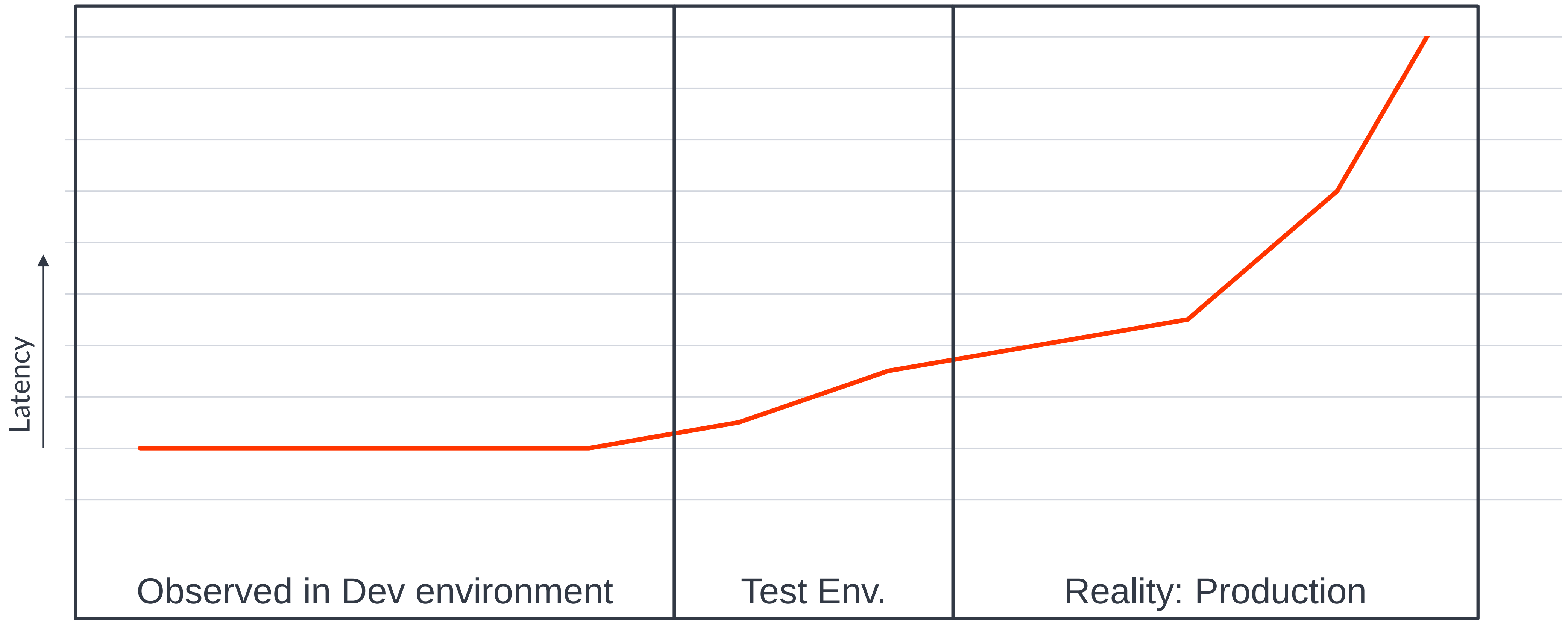
In reality...



In reality, numbers are...

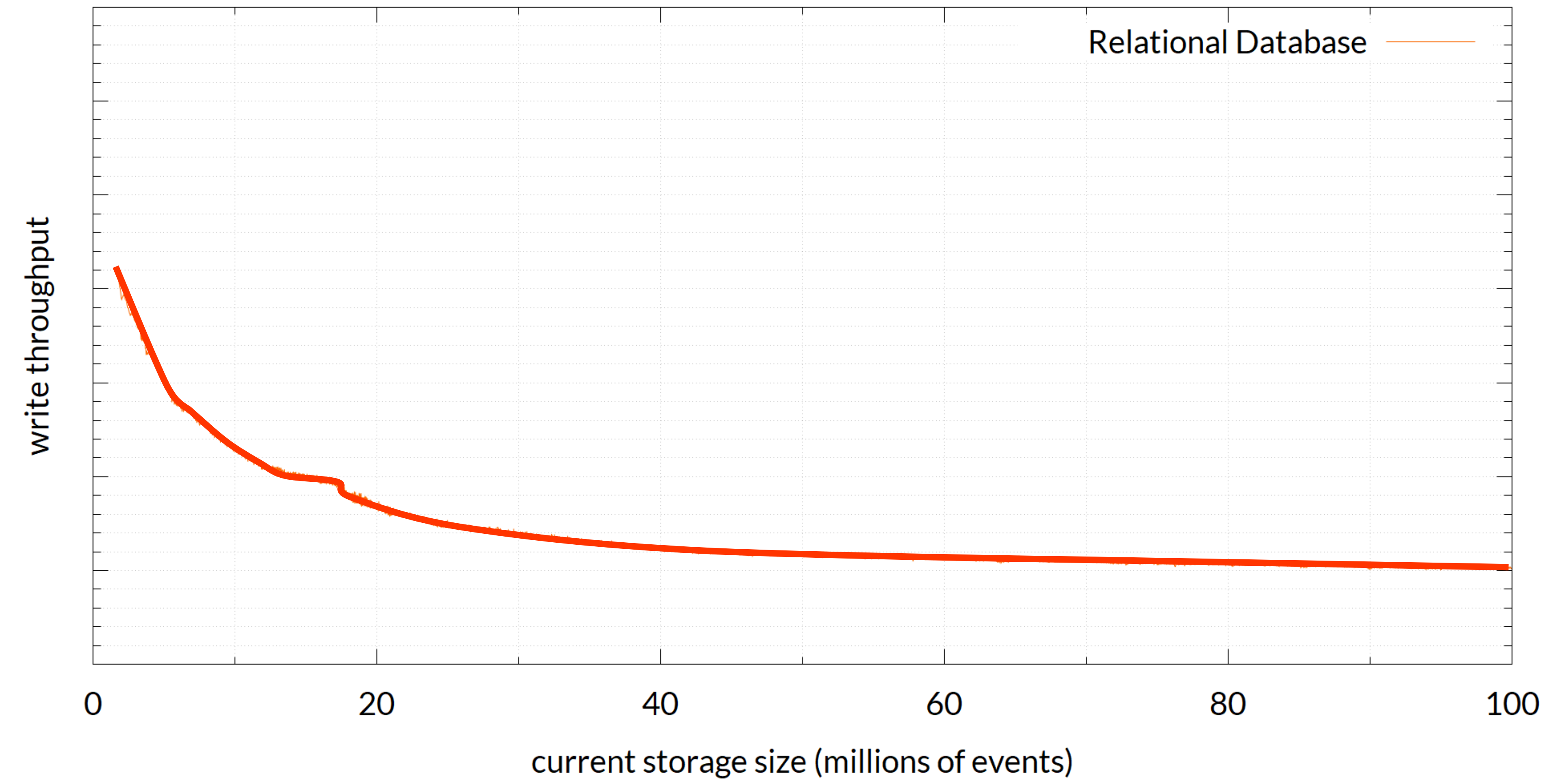


In reality, numbers are worse...



The effort to add data increases with the amount of data stored

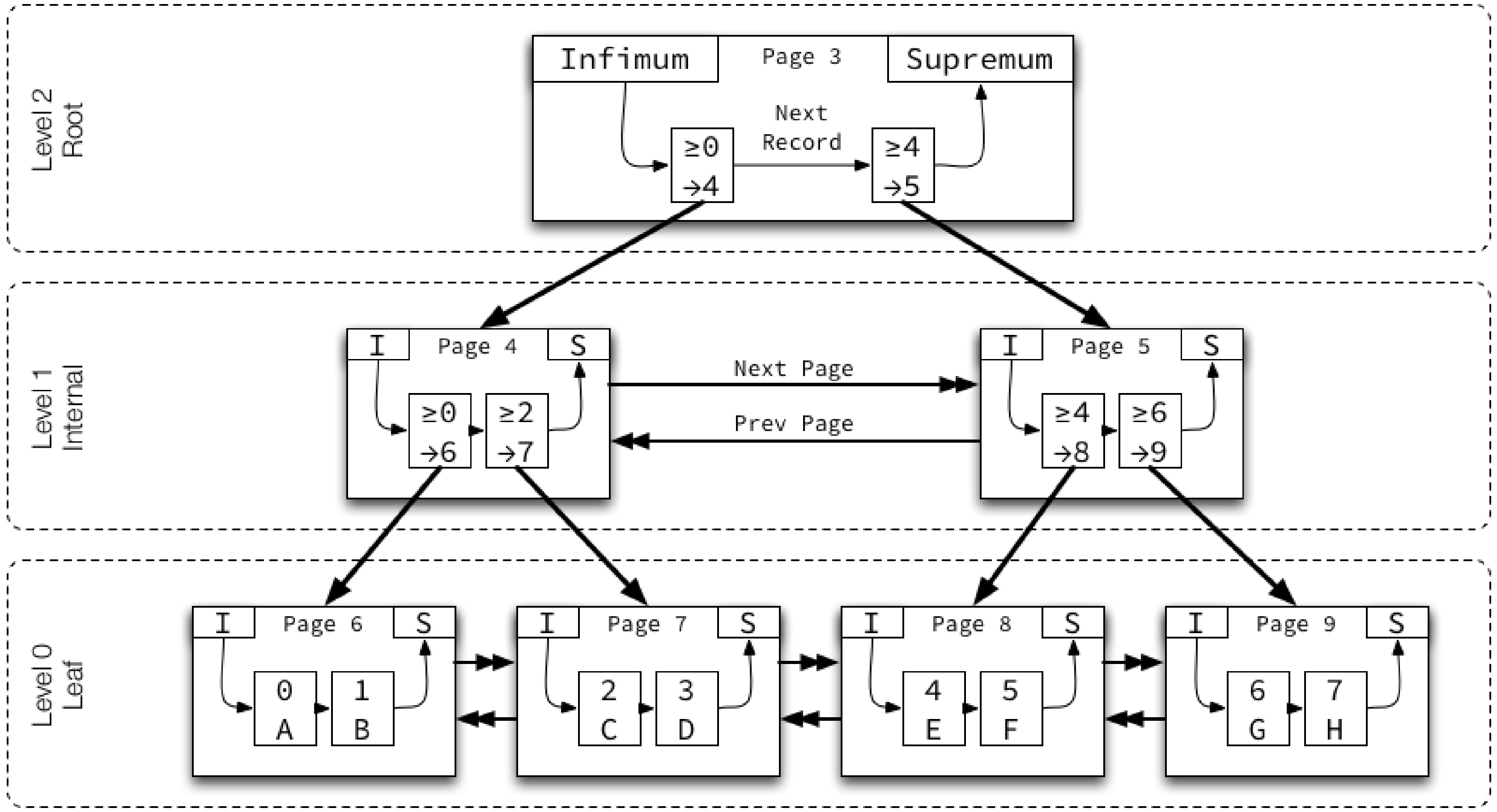
Problem #1



AxonIQ

@allardbz

B-Tree Index



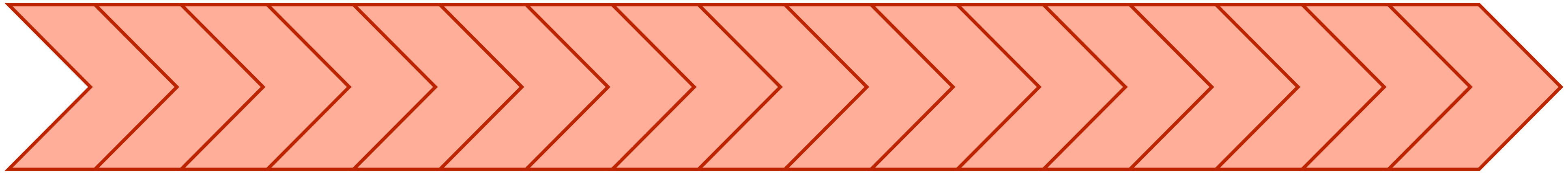
Source: blog.jcole.us

AxonIQ

@allardbz

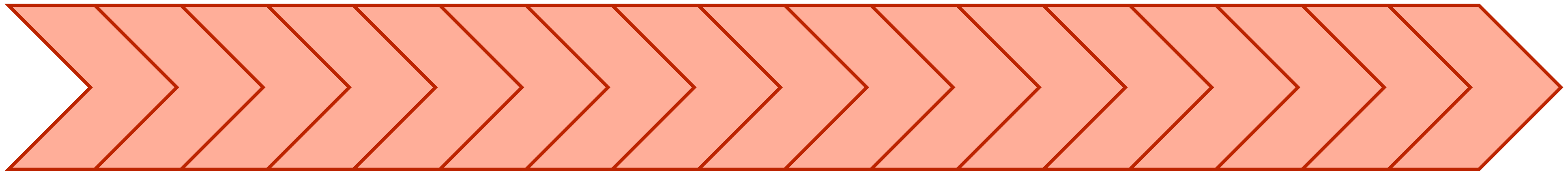
Event Store operations

- Append
- Validate ‘sequence’



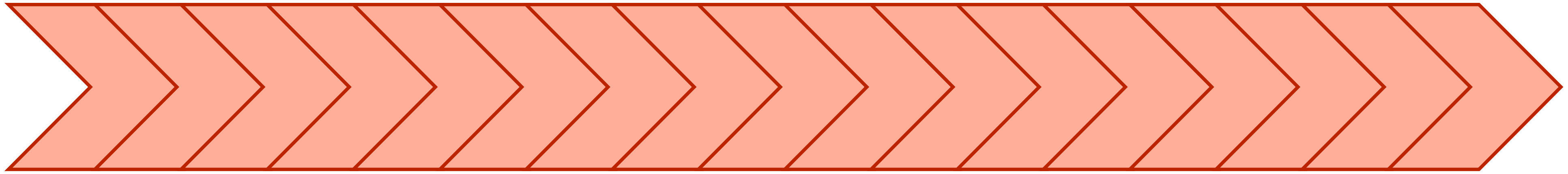
Event Store operations

- Read aggregate's events

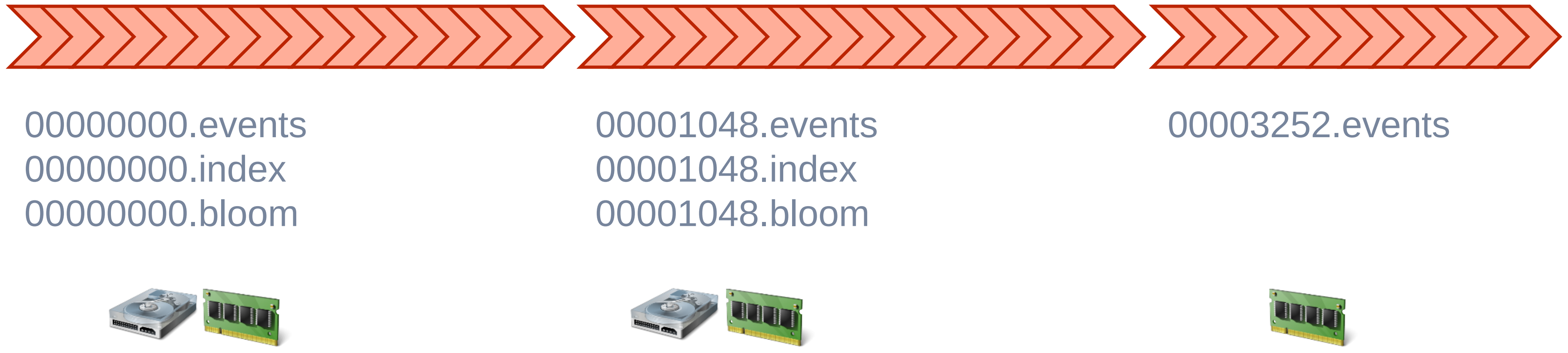


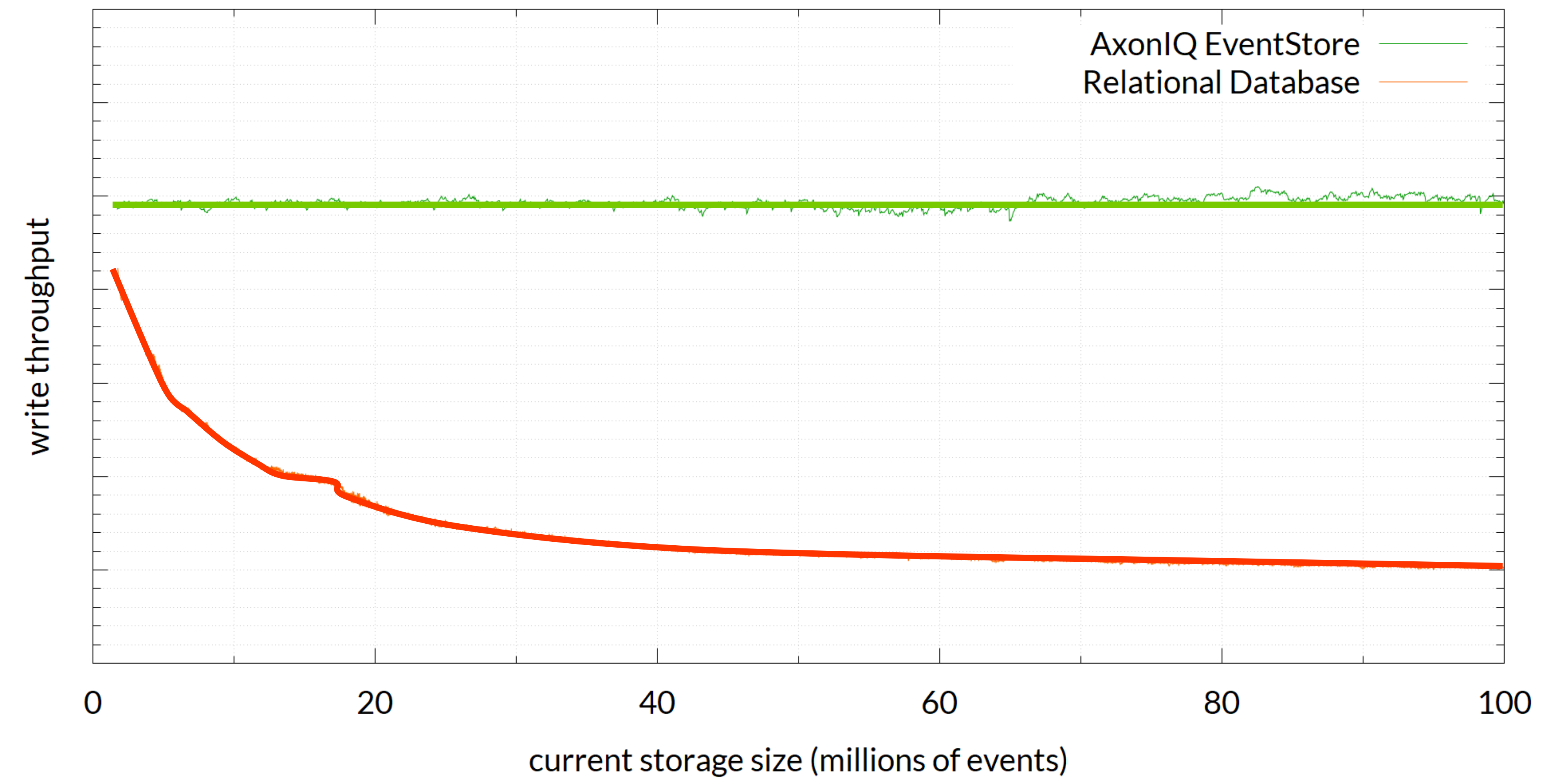
Event Store operations

- Full sequential read



Solution – Partitioning

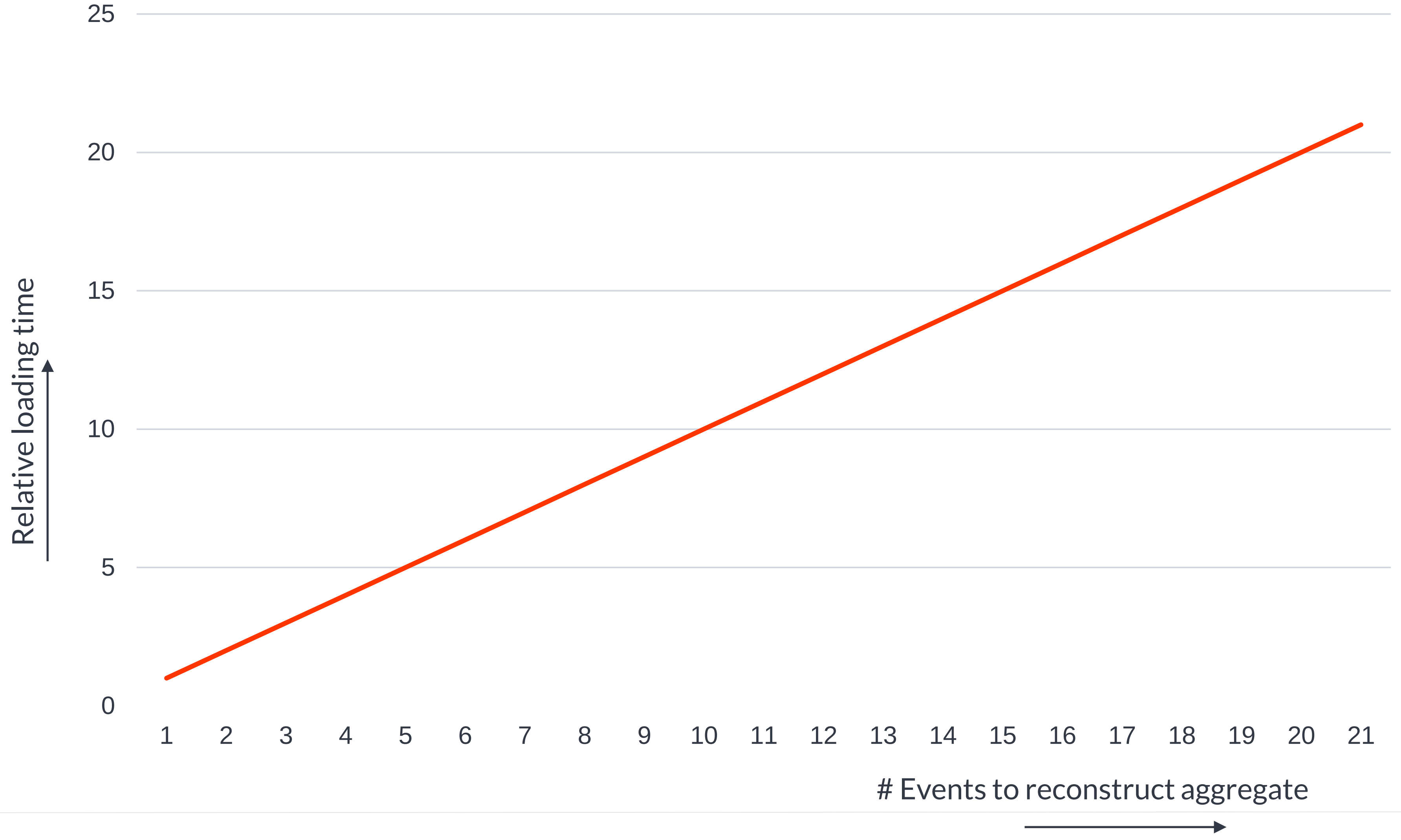



AxonIQ EventStore
Relational Database
write throughput
0
20
40
60
80
100
current storage size (millions of events)

The effort to read an aggregate increases as it’s being used

Problem #2

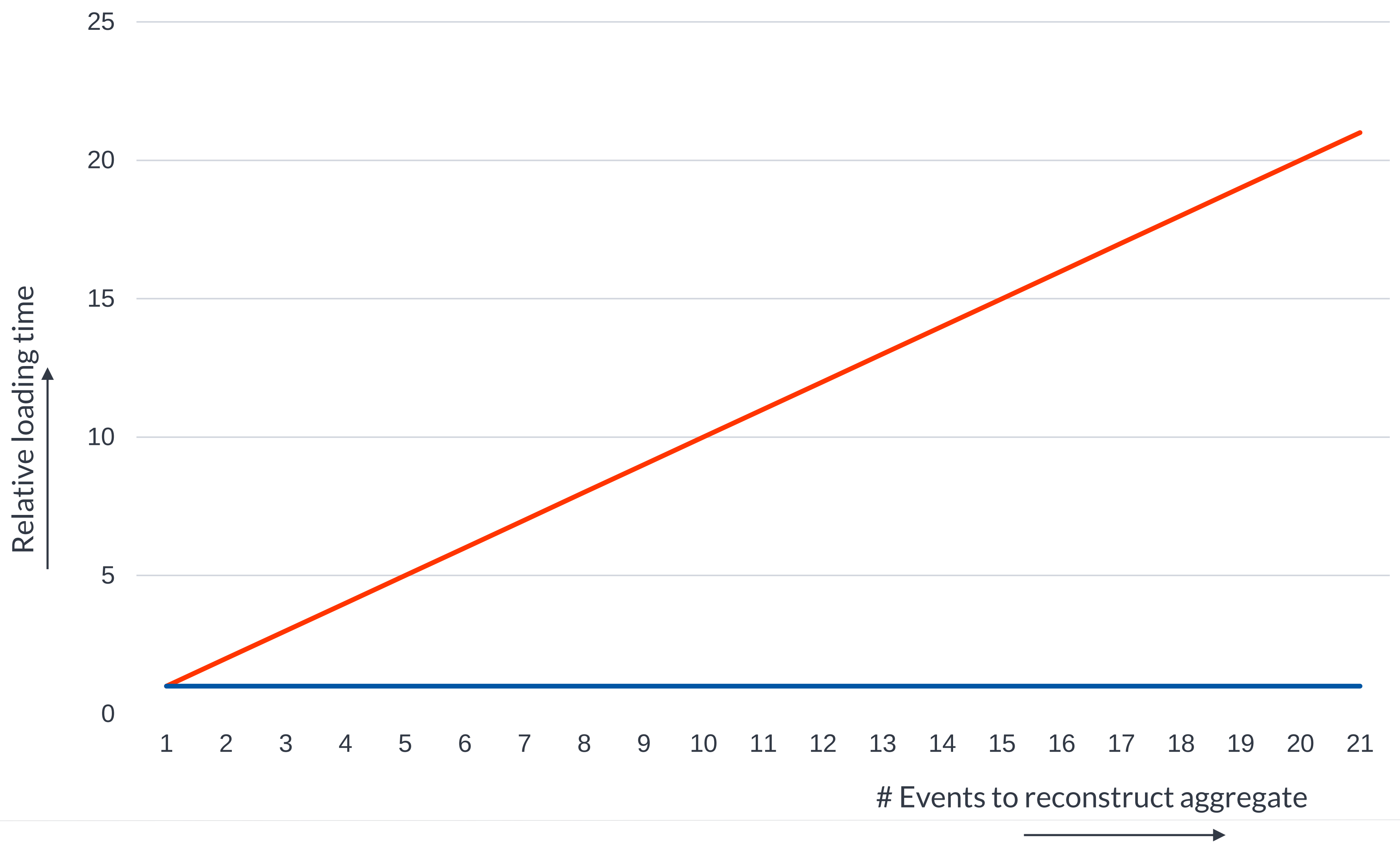
Loading time



Solution – Caching

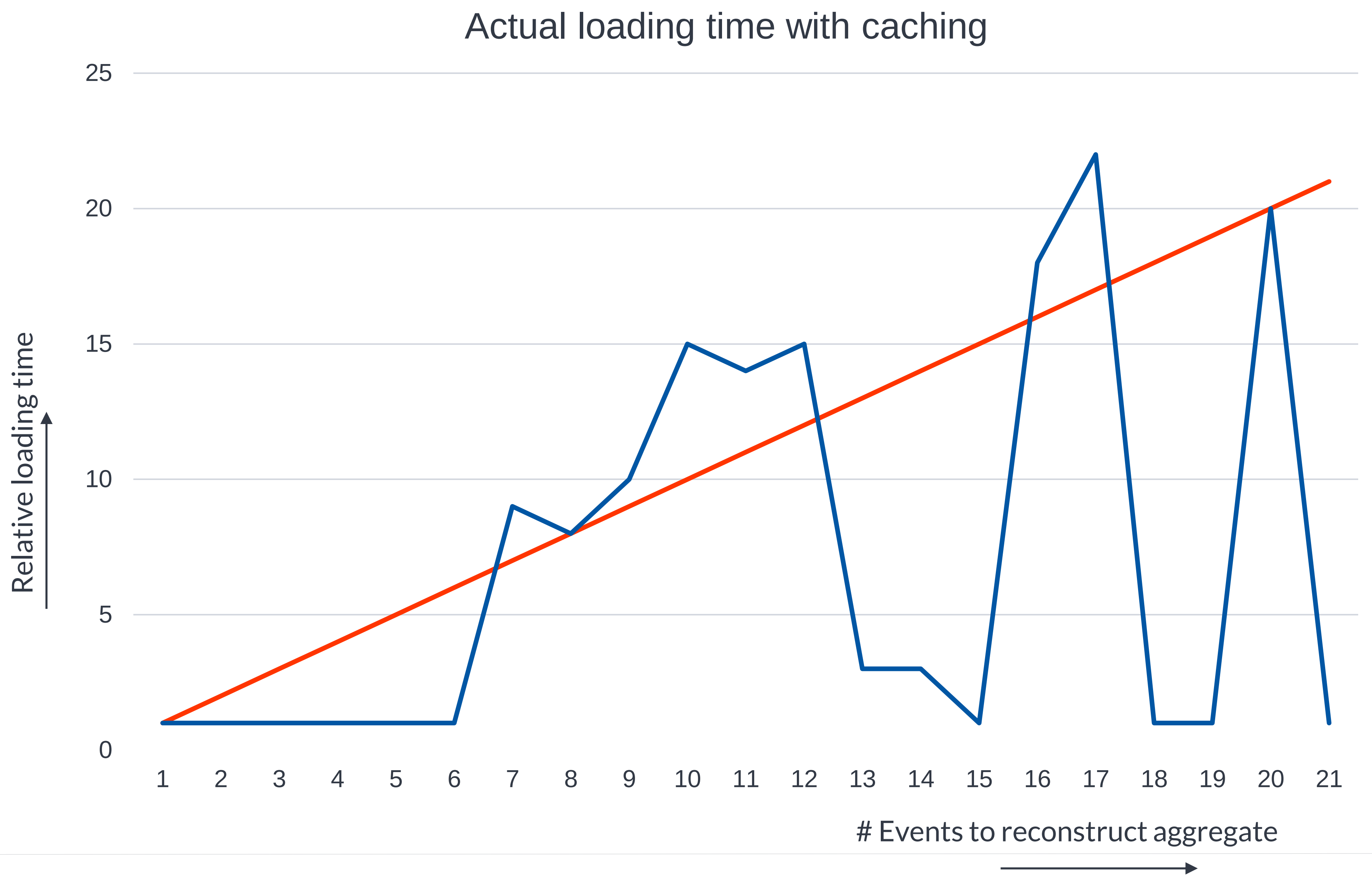
- (Aggressively) caching Aggregates prevents the need to load them

Expected loading time with caching

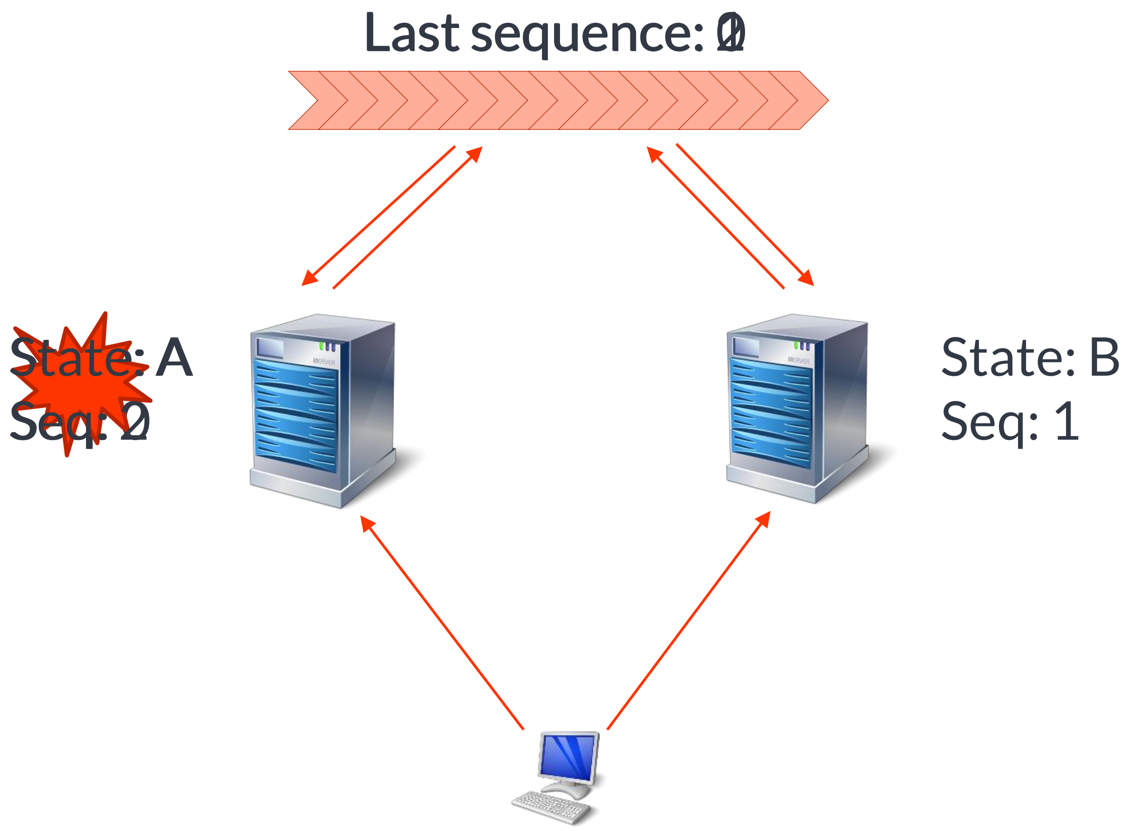


AxonIQ

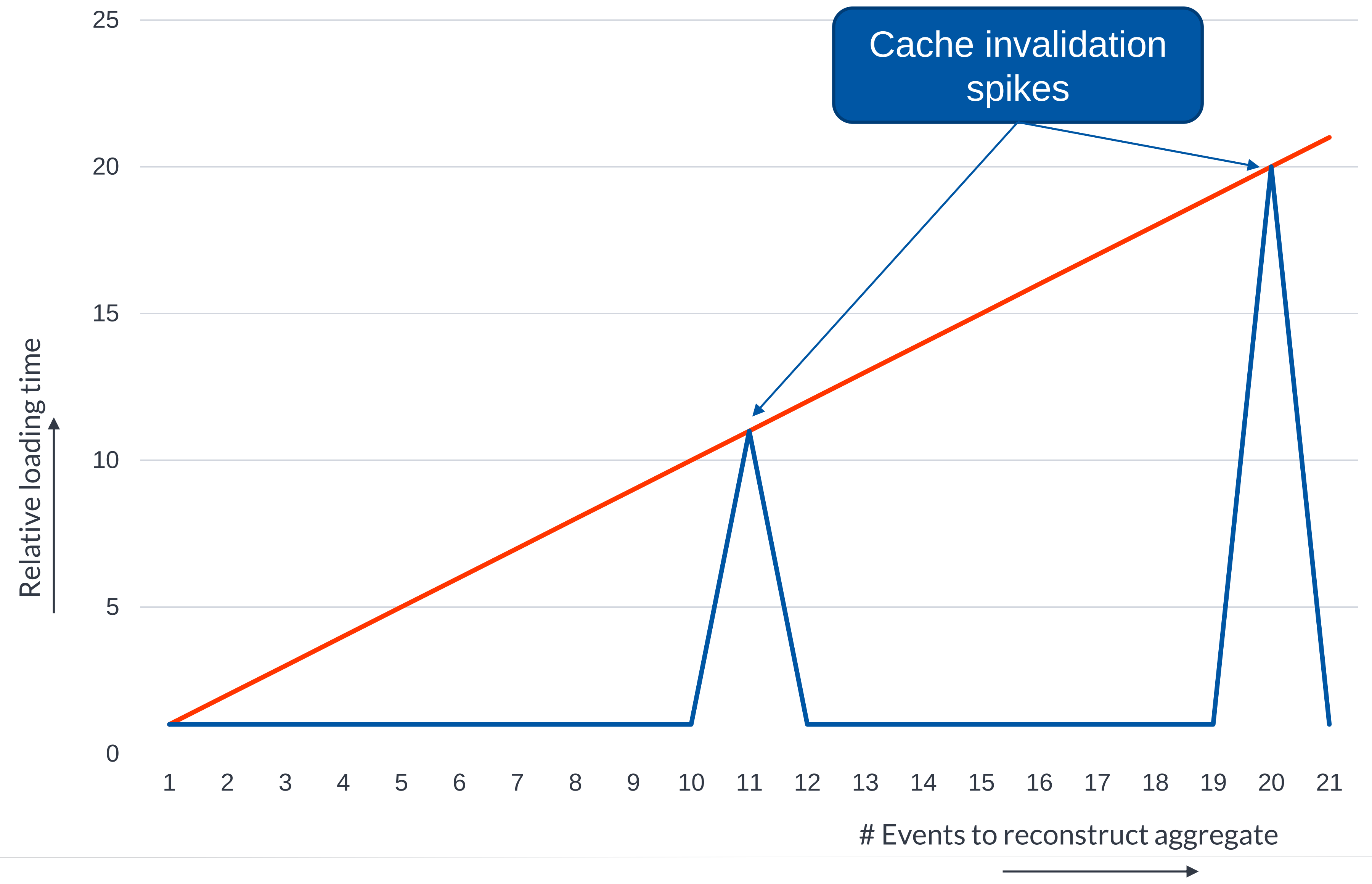
@allardbz


Actual loading time with caching
Relative loading time
# Events to reconstruct aggregate
25
20
15
10
5
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21

Caching & distribution



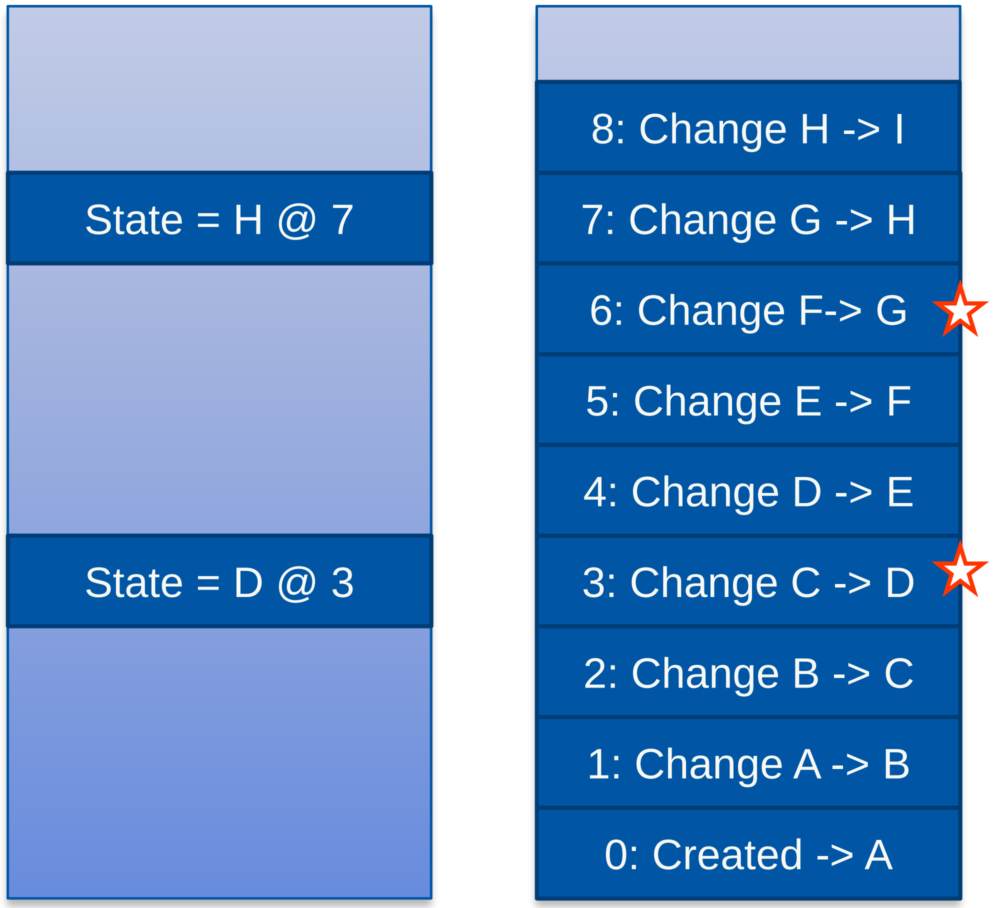
Consistent hashing

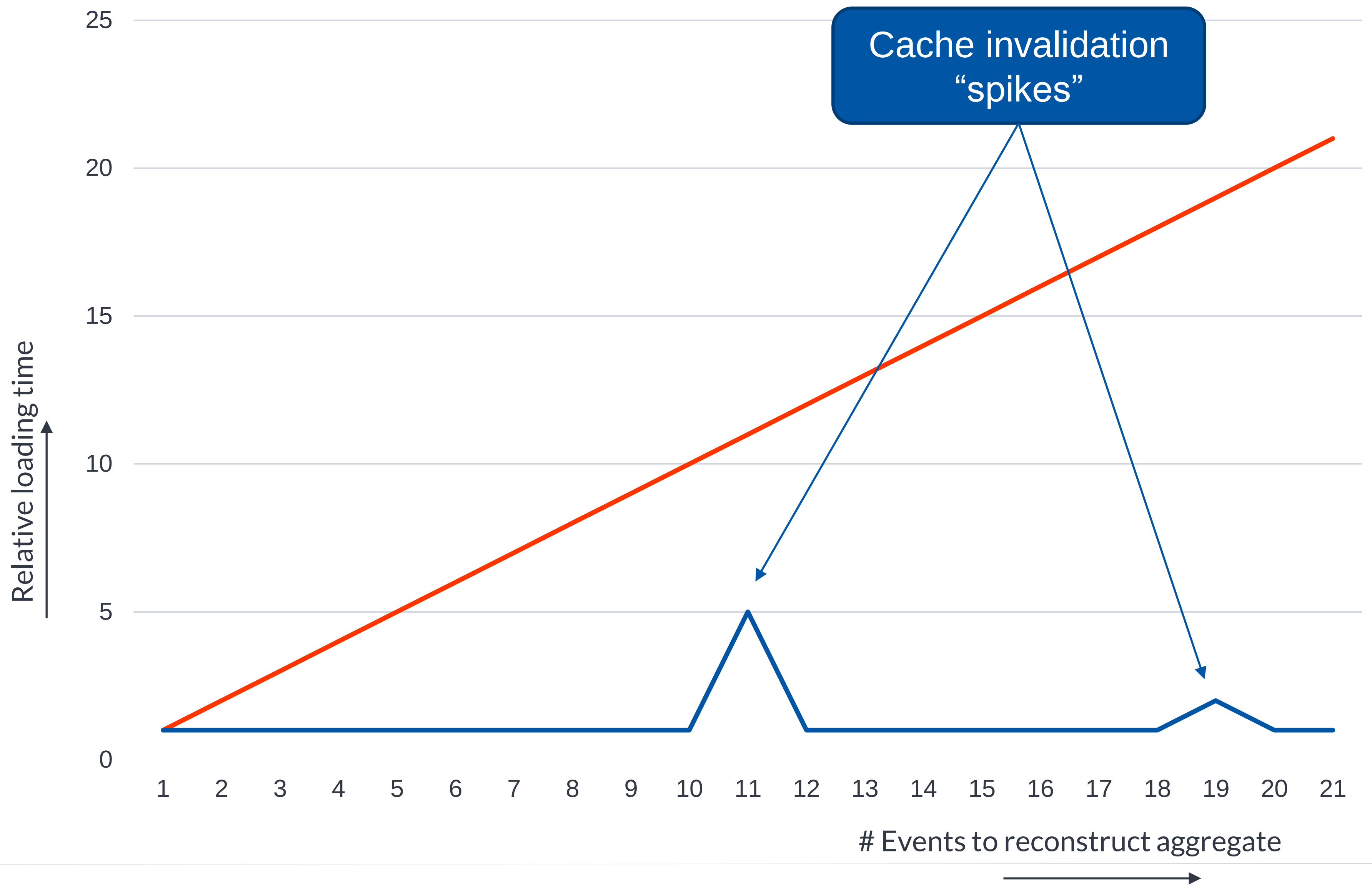


AxonIQ

@allardbz

Snapshotting

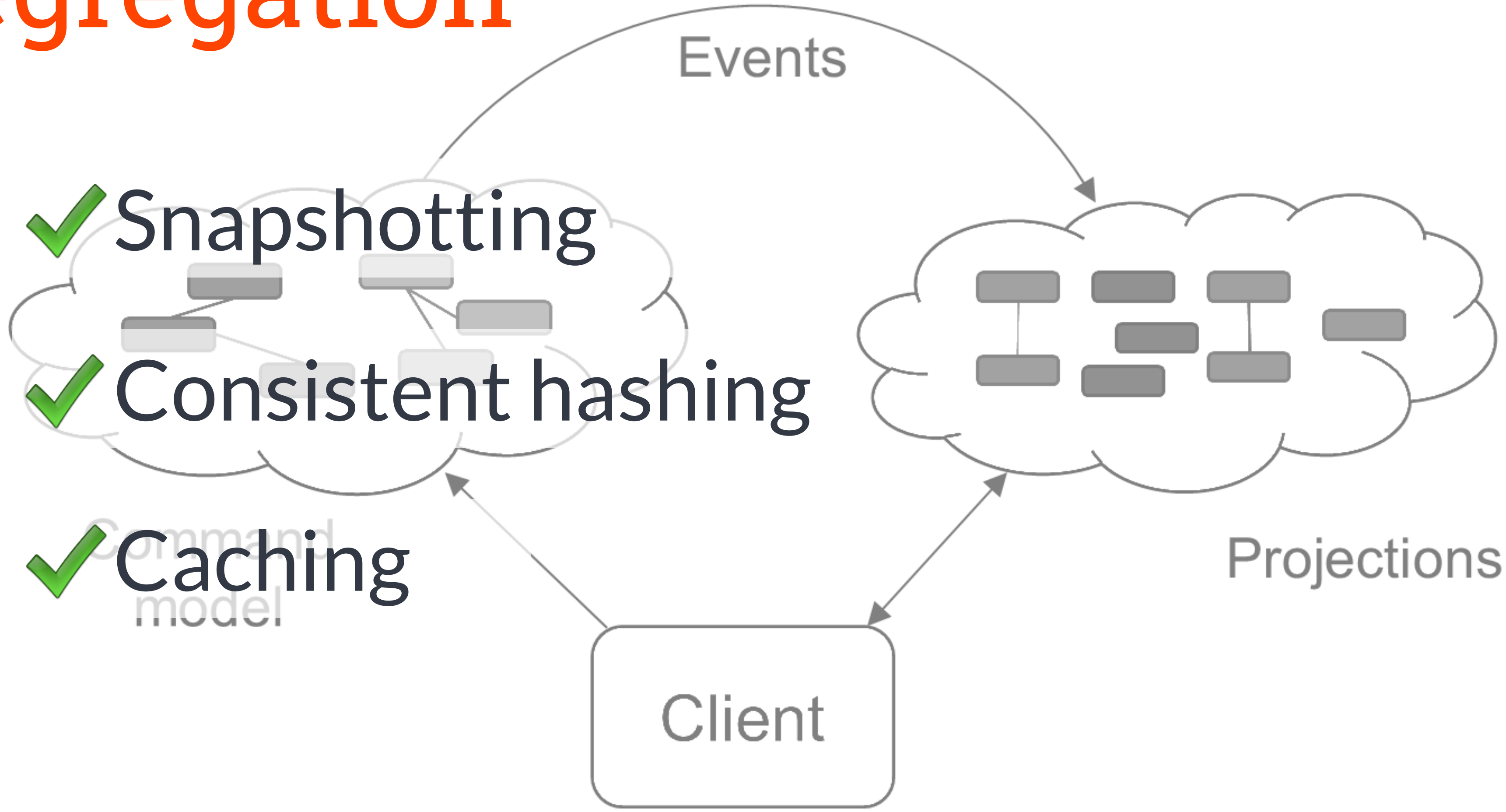




AxonIQ

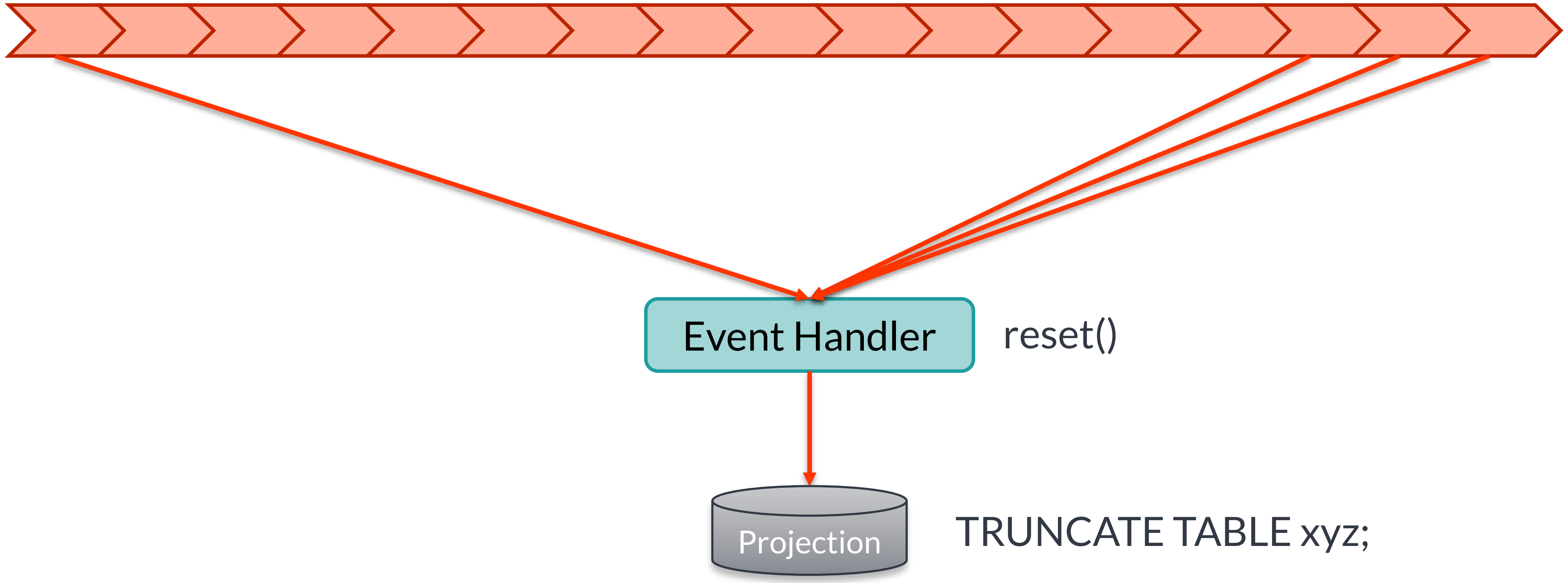
@allardbz

Command Query Responsibility Segregation



- ✓ Snapshotting
- ✓ Consistent hashing
- ✓ Caching

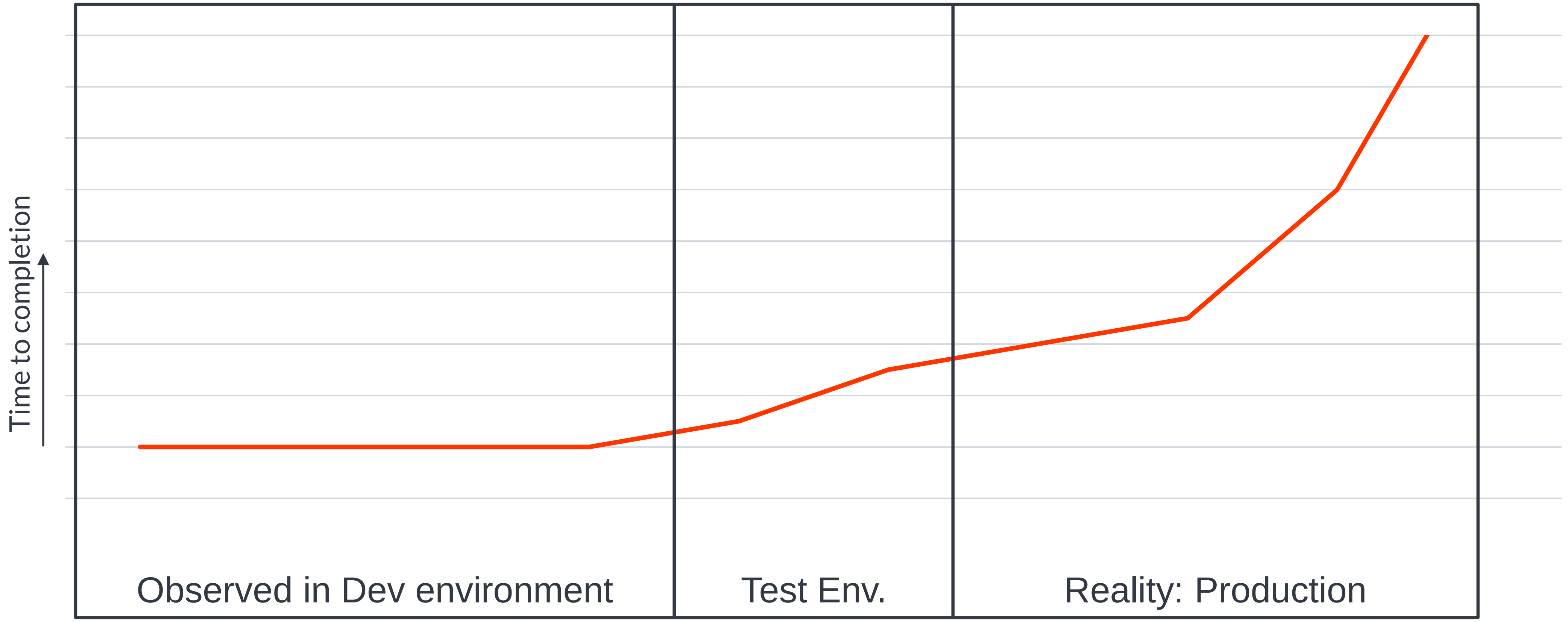
Replays



Replay duration increases as stored data accumulates

Problem #3

Time to replay...



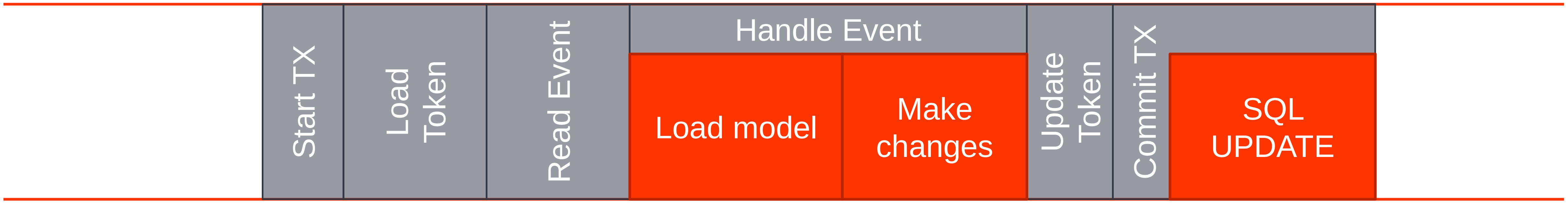
Typical event handler

```
@EventHandler
public void on(RedeemedEvt event) {
    CardSummary summary = entityManager.find(CardSummary.class,
                                             event.getId());
    summary.setRemainingValue(summary.getRemainingValue()
                              - event.getAmount());
}
```

- 250 events per second
- Replaying
  - 1k events: 4 seconds
  - 1M events: 66 minutes
  - 10M events: 11 hours
  - 1B events: 46 days

1. Fetch entity
2. Update state
3. Persist result (implicit)

Event Processing

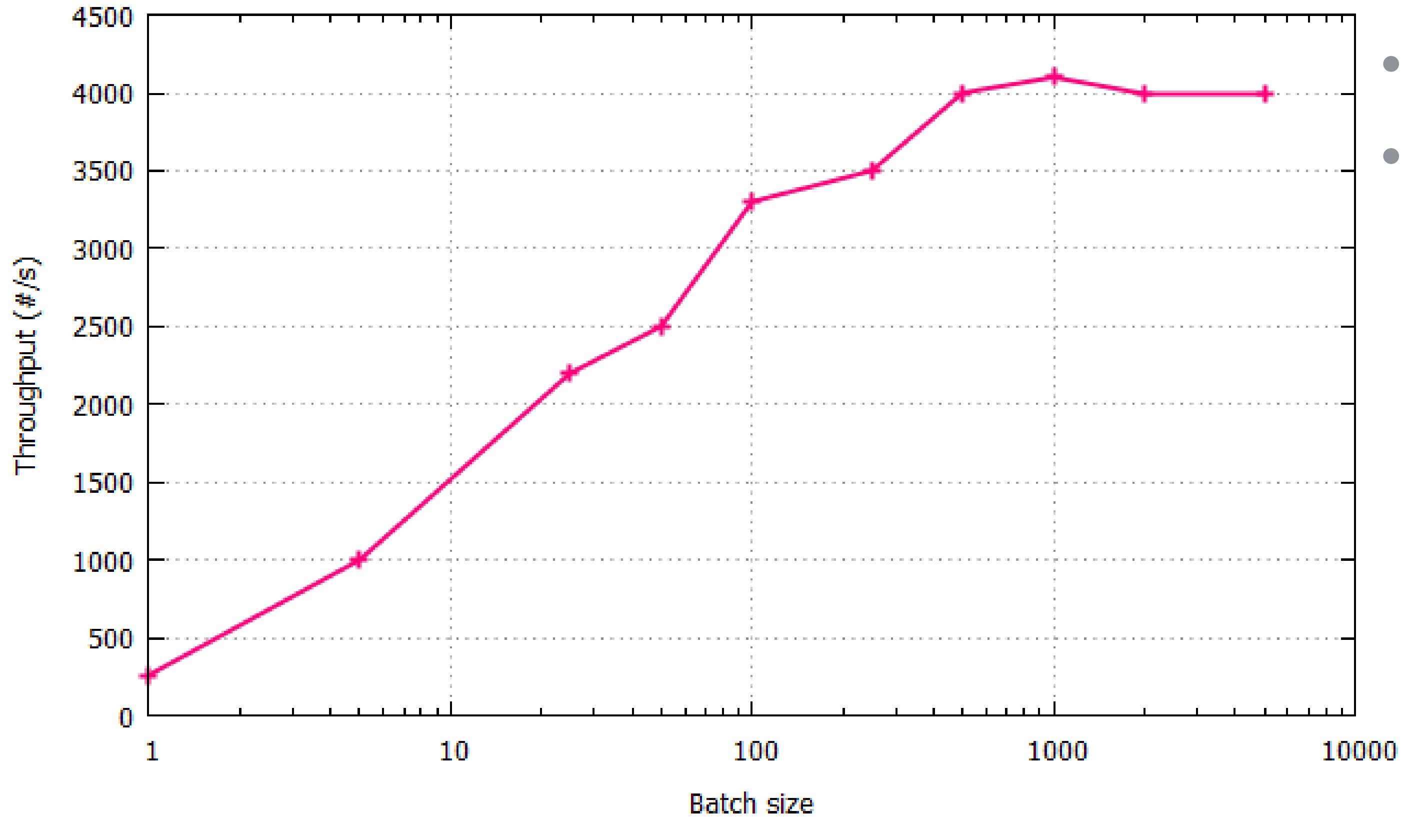


AxonIQ

@allardbz

Partial solution

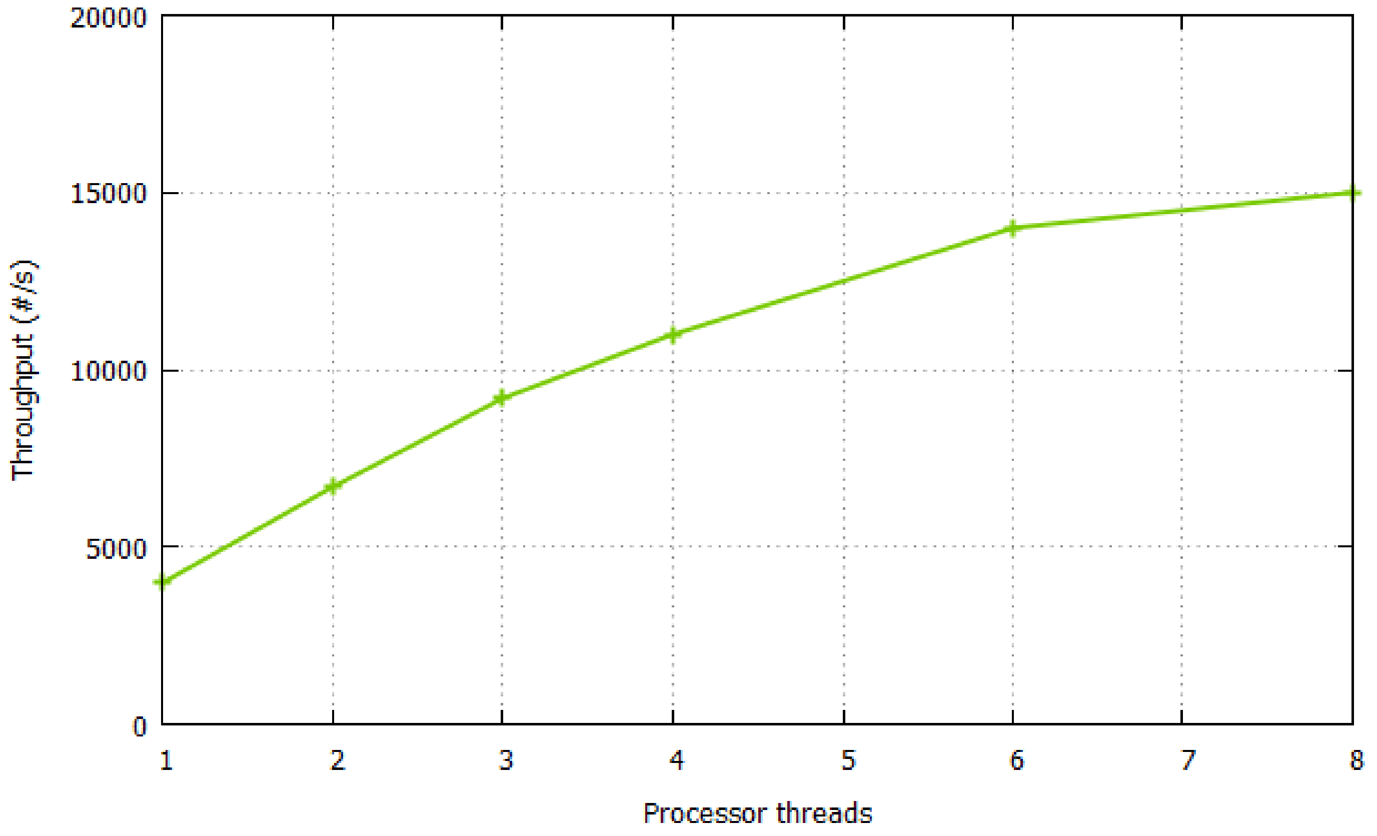
~~Solution~~ – Batch size



- 4 000 events / second
- Replaying
  - 1k events: 250 ms
  - 1M events: 4 minutes
  - 10M events: 42 minutes
  - 1B events: 3 days

Almost the solution

~~Solution~~ – Parallel processing



- 15 000 events / second
- Replaying
  - 1k events: 66 milliseconds
  - 1M events: 66 seconds
  - 10M events: 11 minutes
  - 1B events: 18 hours

AxonIQ

@allardbz

Solution – Batch optimization

- Write "INSERT" / "UPDATE" / "DELETE" statements directly
- Use `UnitOfWork` to stage instructions

- Combine statements into a single one

Combining updates

```
{
   operation = "insert"
   order_id = "a123"
   product_id = "p321"
   count = 20
}
```

```
ItemsAddedToCart {
      order_id = "a123"
      product_id = "p321"
      count = 10
}
```

Combining updates

```
{
   operation = “update”
   order_id = “a123”
   product_id = “p321”
   count = 20
}
```

```
ItemsAddedToCart {
        order_id = “a123”
        product_id = “p321”
        count = 10
}
```

Combining updates

```
{
  operation = "insert"
  order_id = "a123"
  product_id = "p321"
  count = 10
}
```

```
ItemsRemovedFromCart {
        order_id = "a123"
        product_id = "p321"
        count = 10
}
```

Batching in AxonFramework

```
BatchOperations ops =
unitOfWork.getOrComputeResource("batch", k -> {
    BatchOperations bo = new BatchOperations();
    unitOfWork.onPrepareCommit(uow -> bo.execute());
    return bo;
});
```

Optimization results

Parallel optimized batch

- 30 000 events per second
- Replaying
  - 1k events: 33 milliseconds
  - 1M events: 33 seconds
  - 10M events: 5.5 minutes
  - 1B events: 9 hours

Naive

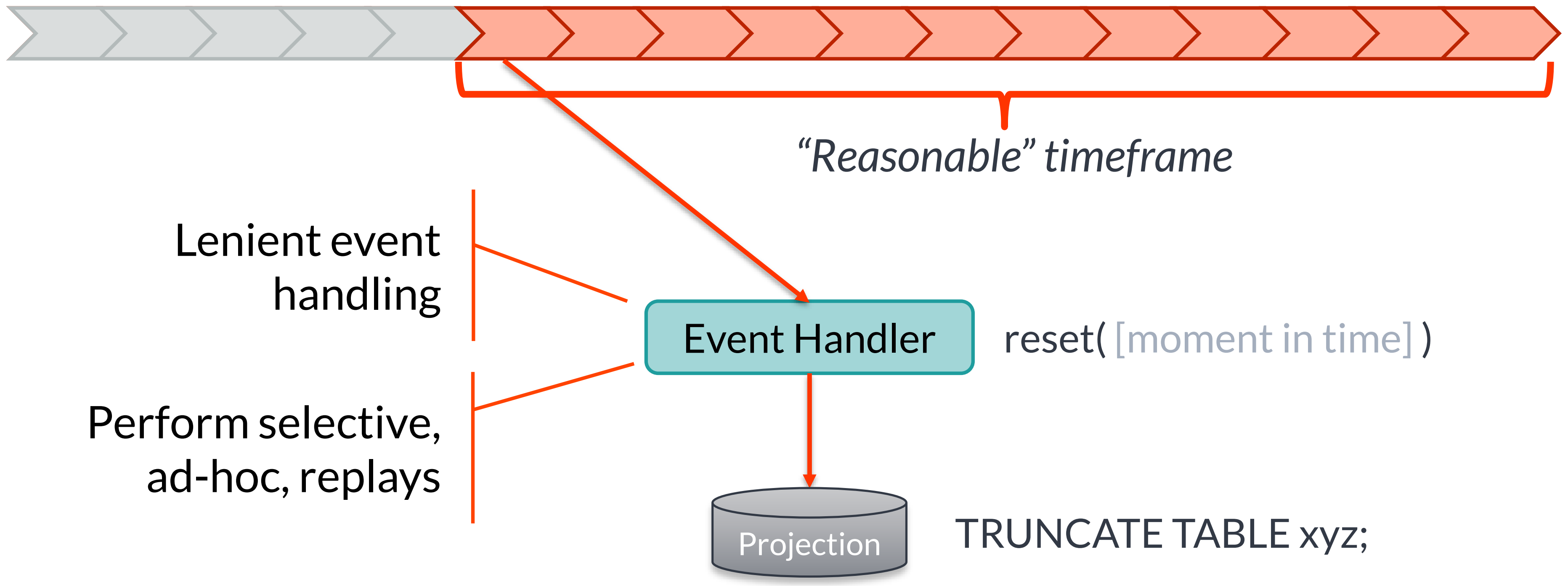
- 250 events per second
- Replaying
  - 1k events: 4 seconds
  - 1M events: 66 minutes
  - 10M events: 11 hours
  - 1B events: 46 days

But wait….

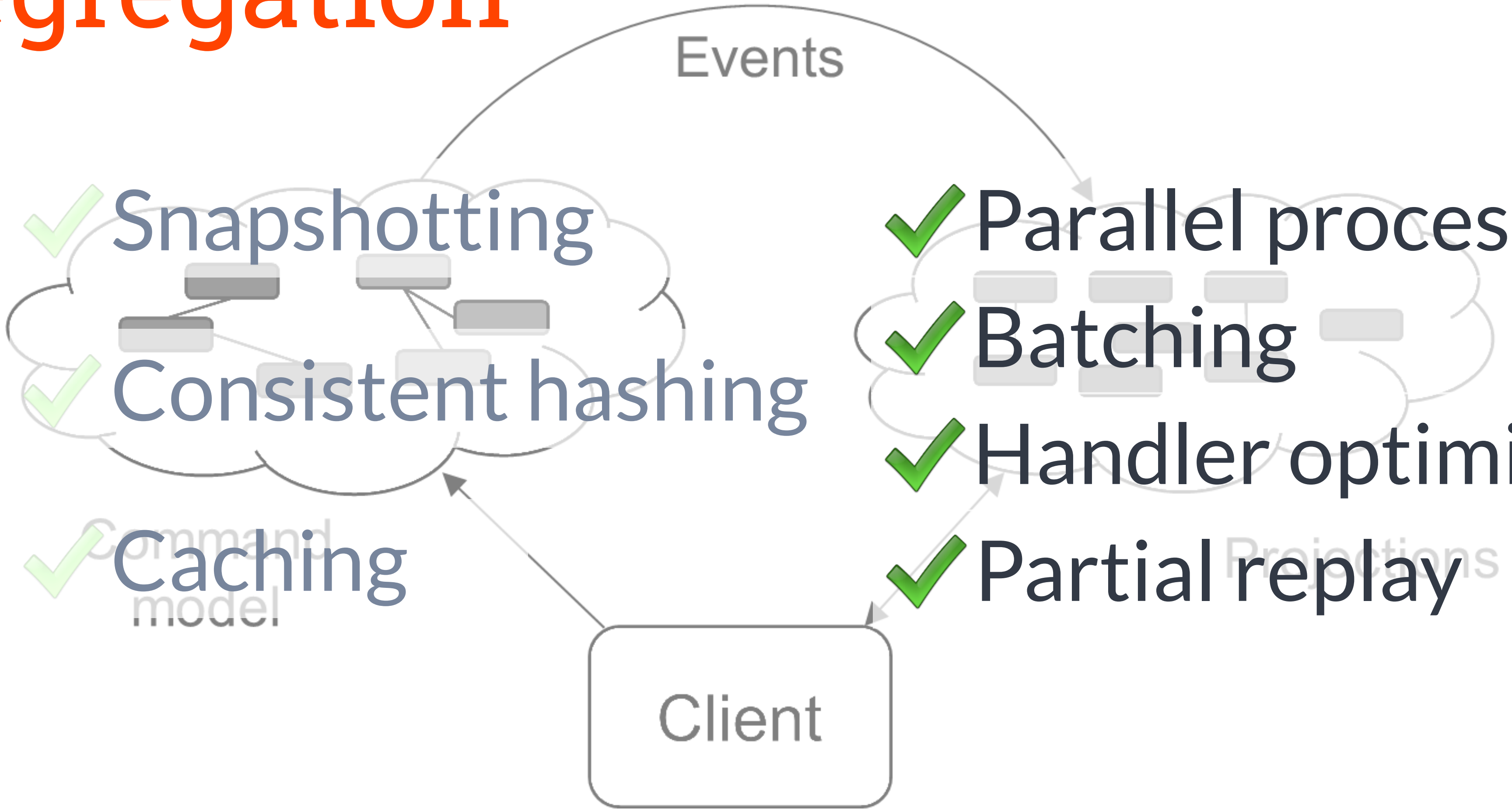
Replay duration $\overset{\text{still}}{\ }$ increases as stored data accumulates

Problem #3

Partial replays



Command Query Responsibility Segregation



- ✓ Snapshotting
- ✓ Consistent hashing
- ✓ Caching

- ✓ Parallel processing
- ✓ Batching
- ✓ Handler optimization
- ✓ Partial replay

CQRS and Event Sourcing

…

CQRS and Event Sourcing

aren’t “faster”

CQRS and Event Sourcing aren't "magic Pixy Dust"

CQRS and Event Sourcing

require tuning

(like any other technology would...)

CQRS and Event Sourcing

allow for more focused, efficient, optimization

References

- Axon
  - axoniq.io
  - github.com/axonframework
  - github.com/axoniq
  - @axonframework
  - @axon_iq

- QuickStart: axoniq.io/download