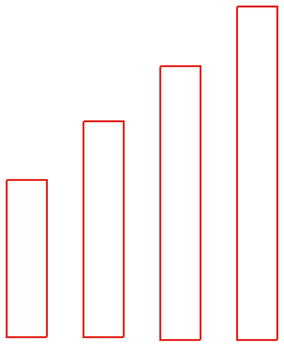
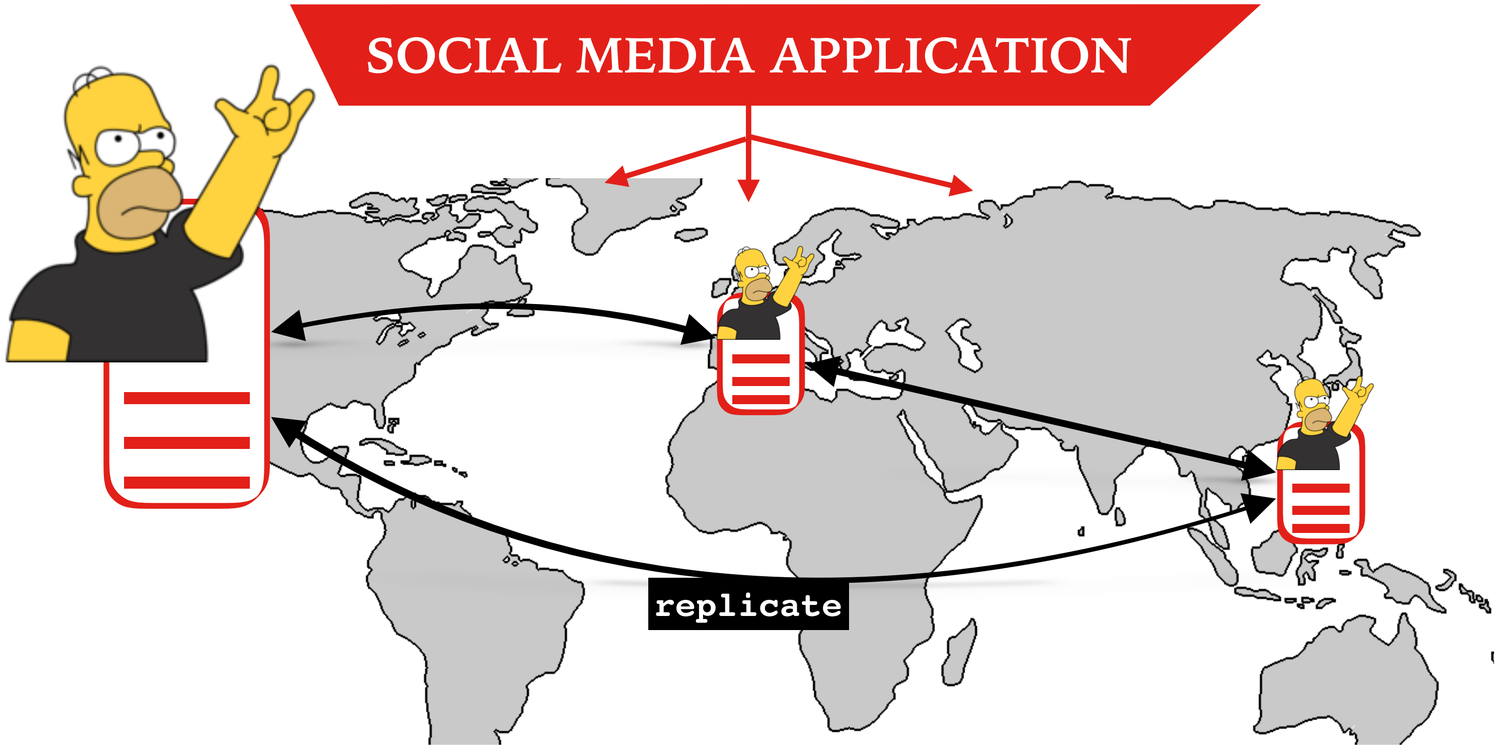


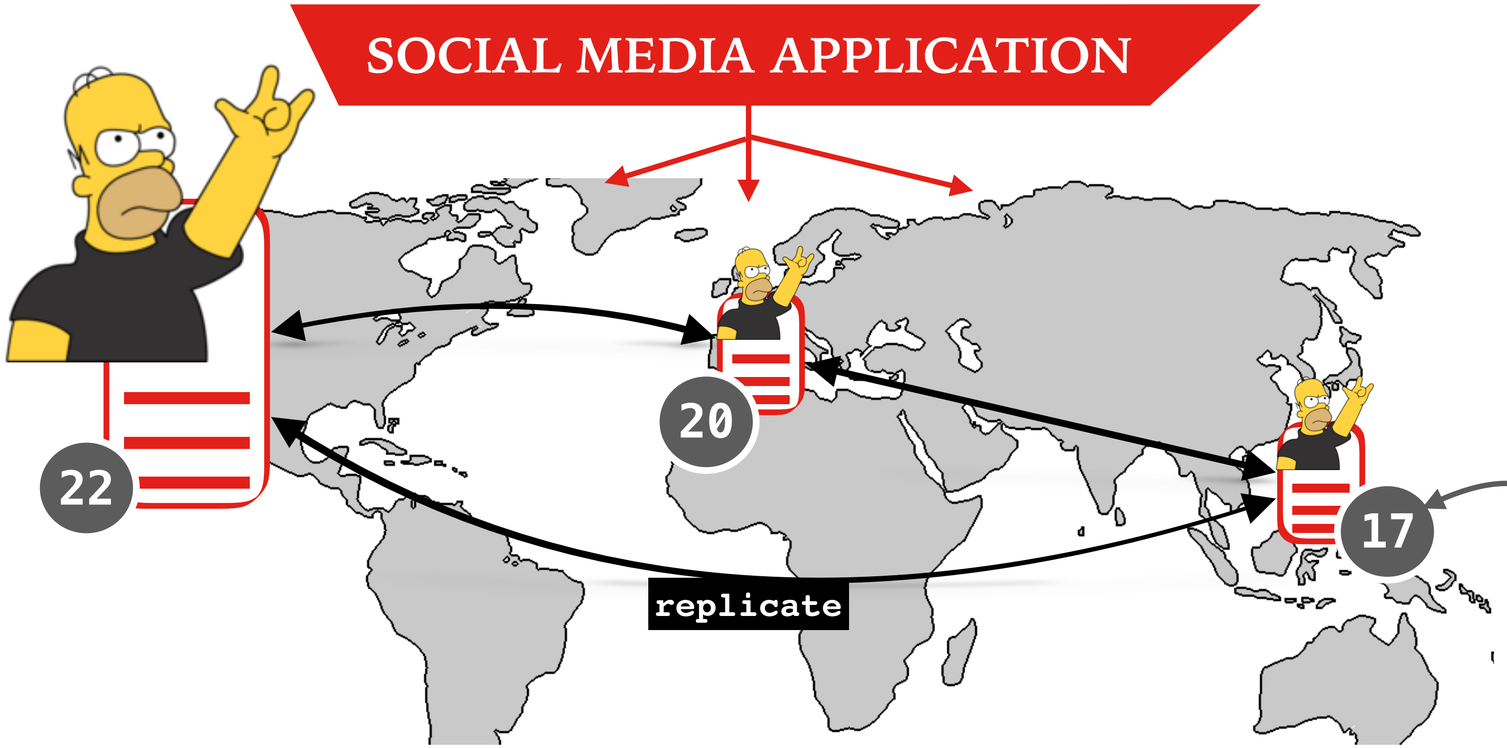
Incremental Consistency Guarantees

For Replicated Objects

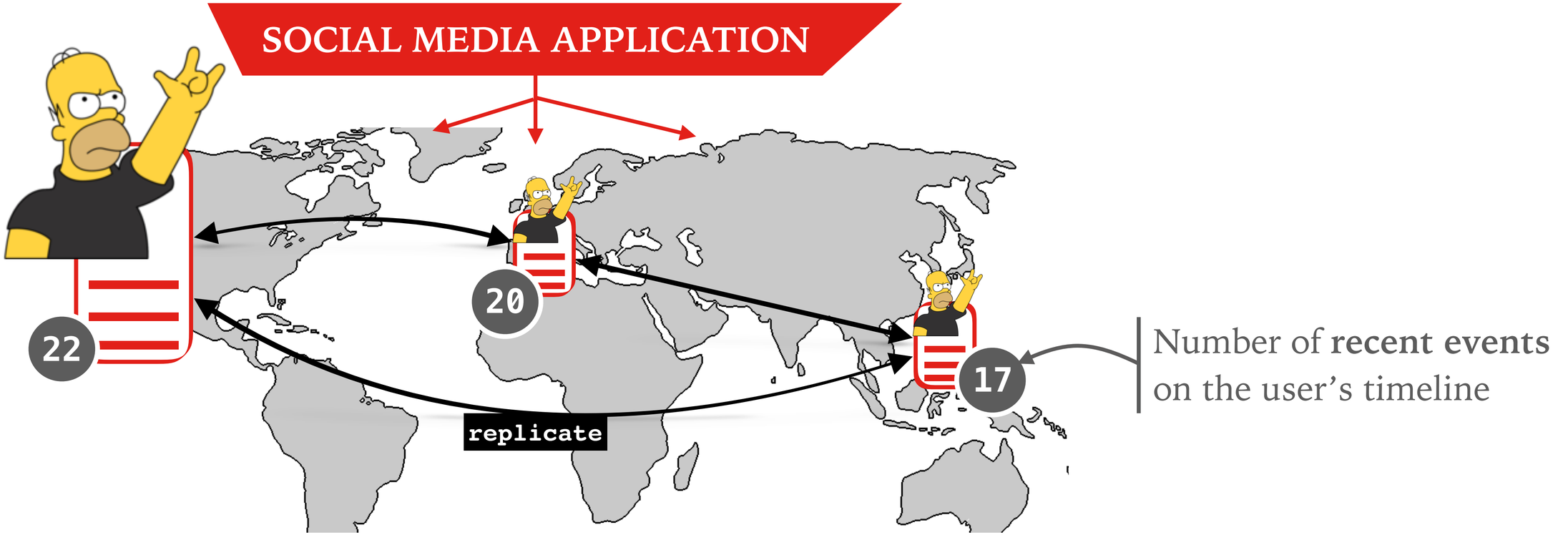
Rachid Guerraoui, Matej Pavlovic, Dragos-Adrian Seredinschi

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE


SOCIAL MEDIA APPLICATION
replicate



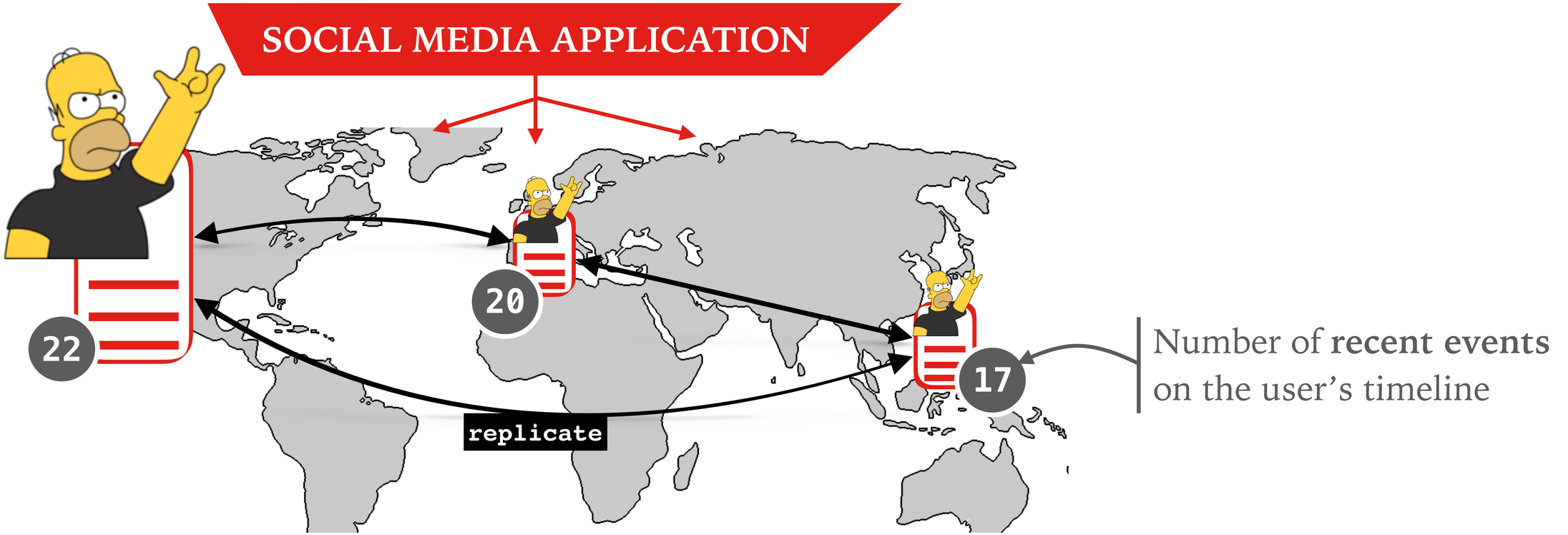
Number of **recent events** on the user's timeline



Strong Consistency

- Returns the **correct** data 22
- Latency: ~200 ms
- Can become **unavailable**

[CAP], [PACELC]



Strong Consistency

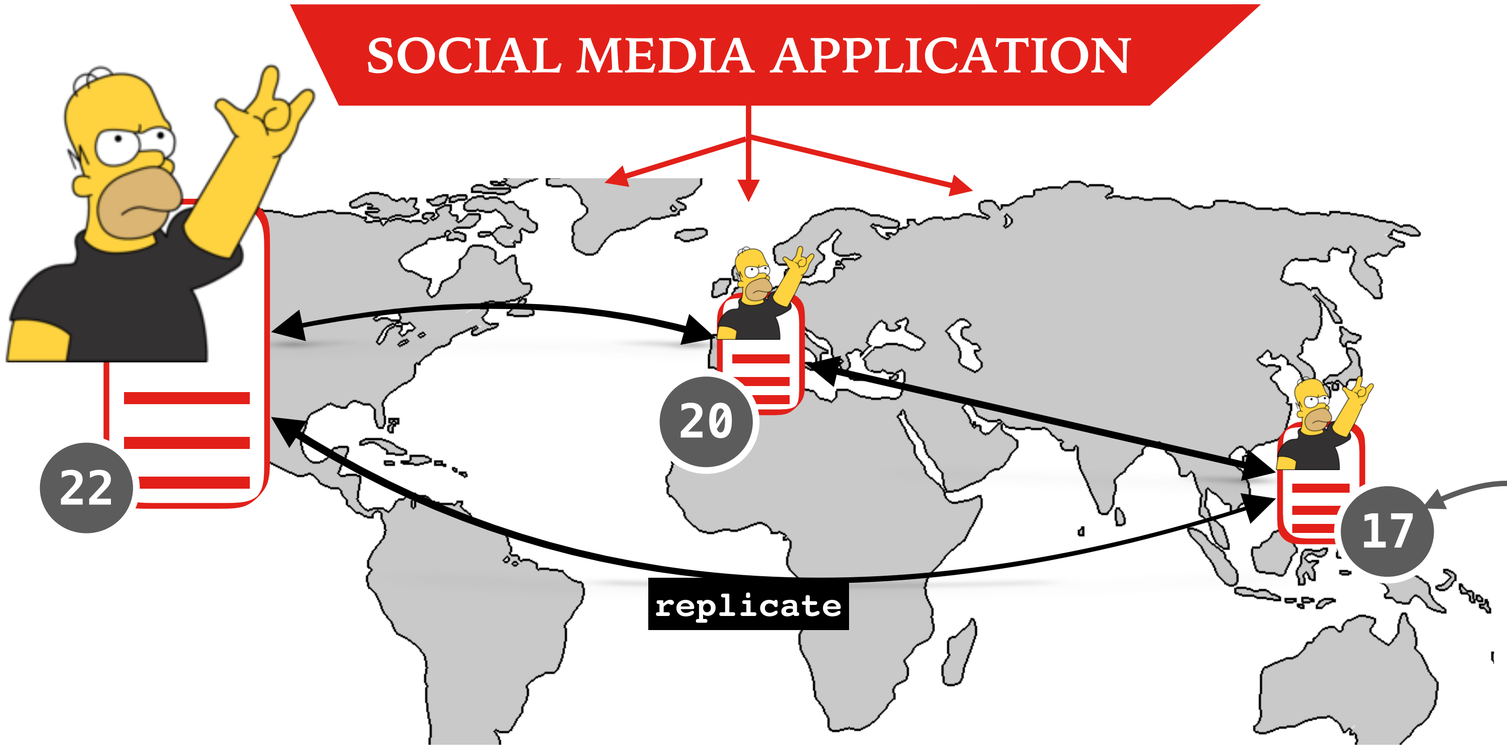
- Returns the **correct** data 22
- Latency: ~200 ms
- Can become **unavailable**

[CAP], [PACELC]

Weak Consistency

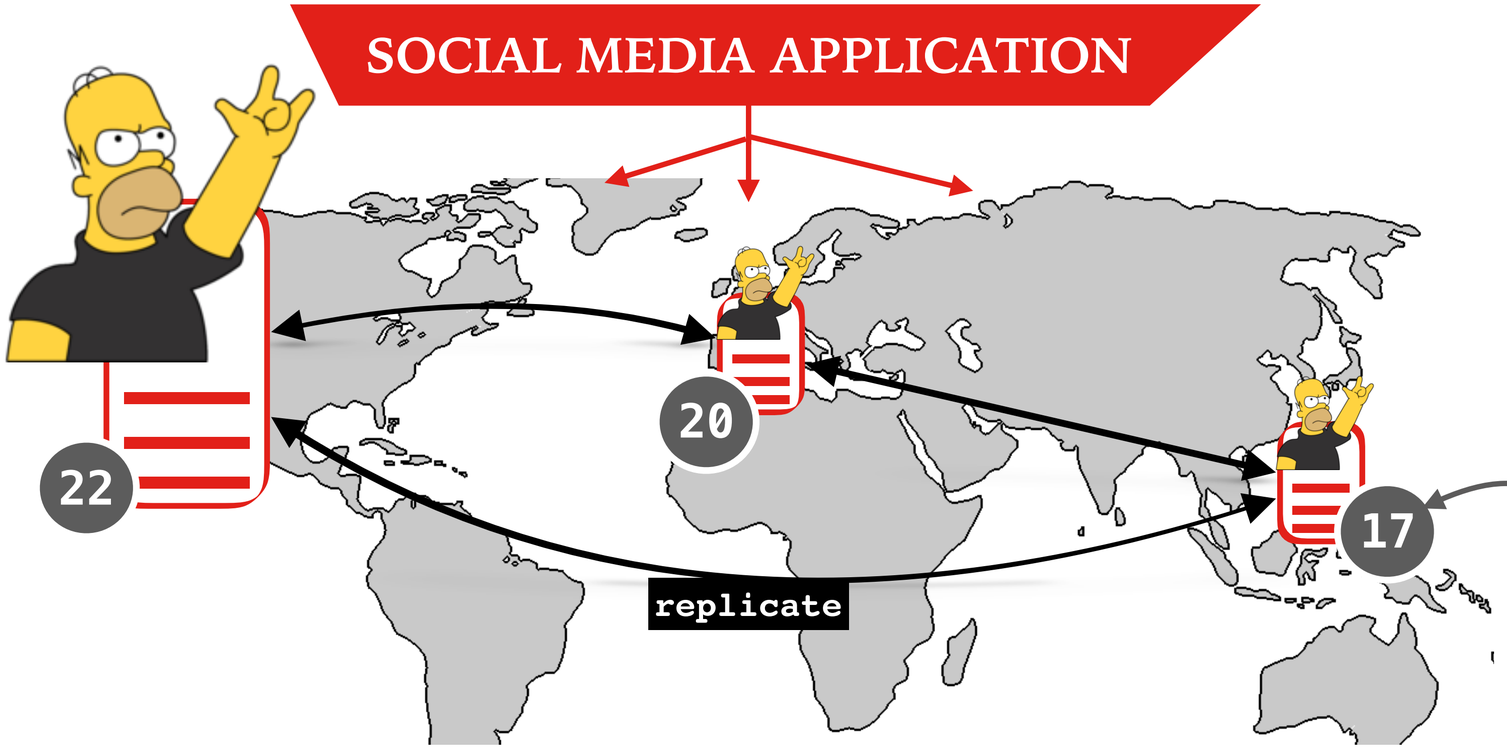
- Latency: ~100 ms
- **High availability**
- Allows inconsistencies: can return

22 *or* 20 *or* 17



Strong Consistency **Weak Consistency**

Neither model is ideal!



Strong Consistency **Weak Consistency**

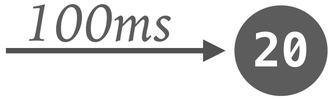
Neither model is ideal!

↓

We use both models.

Multiple models

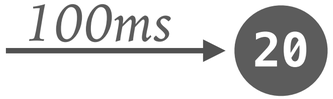
1. **Weak consistency**



2. **Strong consistency**

Multiple models

1. **Weak consistency**



2. **Strong consistency**

Increasingly many systems expose multiple consistency models:

App Engine

cassandra

Dynamo [SOSP'07]

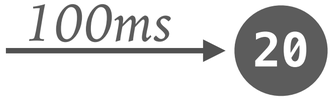
SimpleDB

riakKV

Pileus [SOSP'13]

Multiple models

1. **Weak consistency**



2. **Strong consistency**

Increasingly many systems expose multiple consistency models:

App Engine

cassandra

Dynamo [SOSP'07]

SimpleDB

riakKV

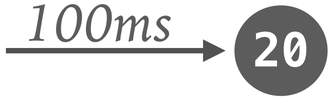
Pileus [SOSP'13]

Issues

1. *Send multiple requests?*
2. *How to leverage individual responses?*
3. *Semantics?*
4. *…*

Multiple models

1. **Weak consistency**



2. **Strong consistency**

Increasingly many systems expose multiple consistency models:

App Engine

cassandra

Dynamo [SOSP'07]

SimpleDB

riakKV

Pileus [SOSP'13]

Issues

1. *Send multiple requests?*
2. *How to leverage individual responses?*
3. *Semantics?*
4. *…*

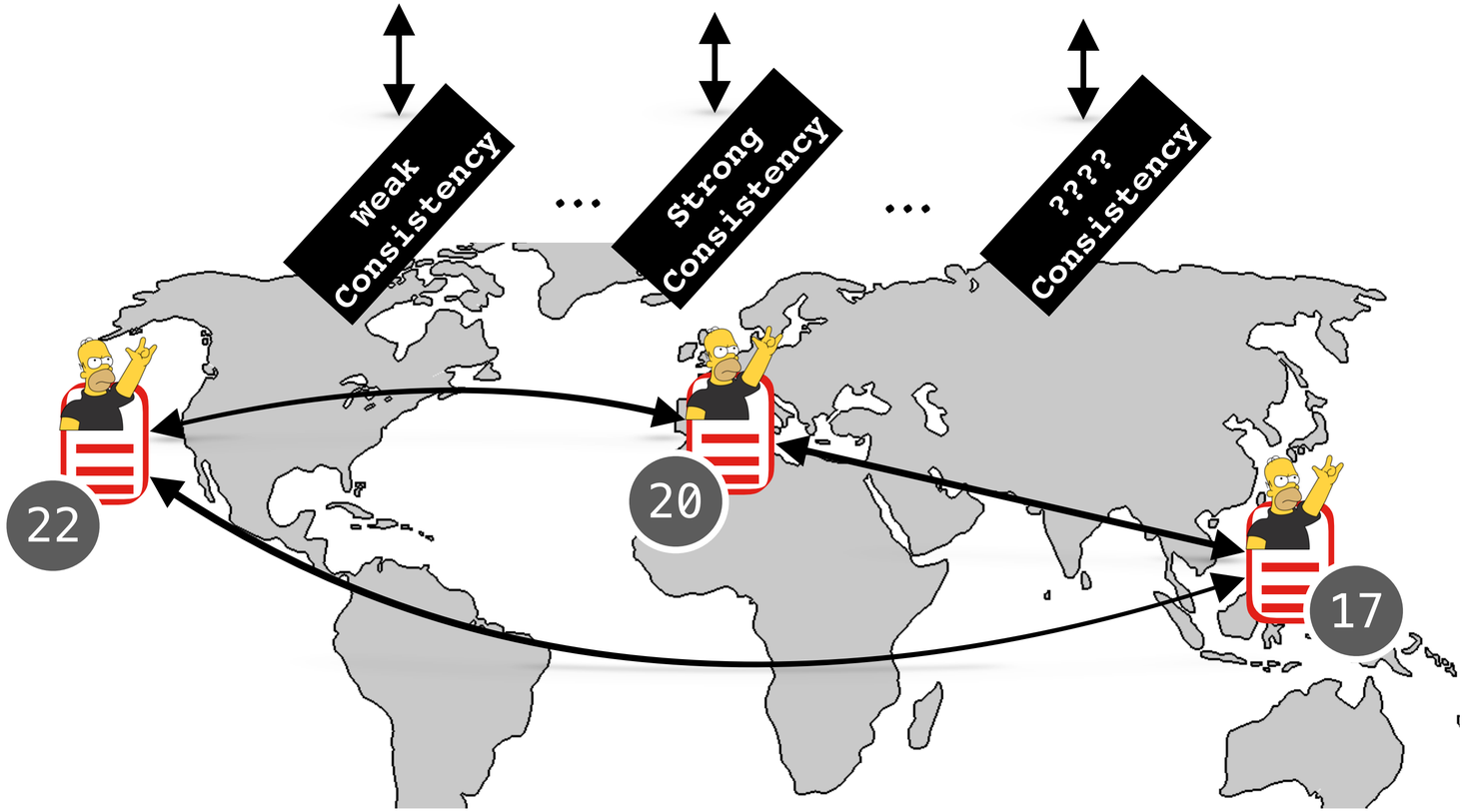
Problem

How do you program with

- ★ **inconsistencies?**
- ★ **multiple values?**

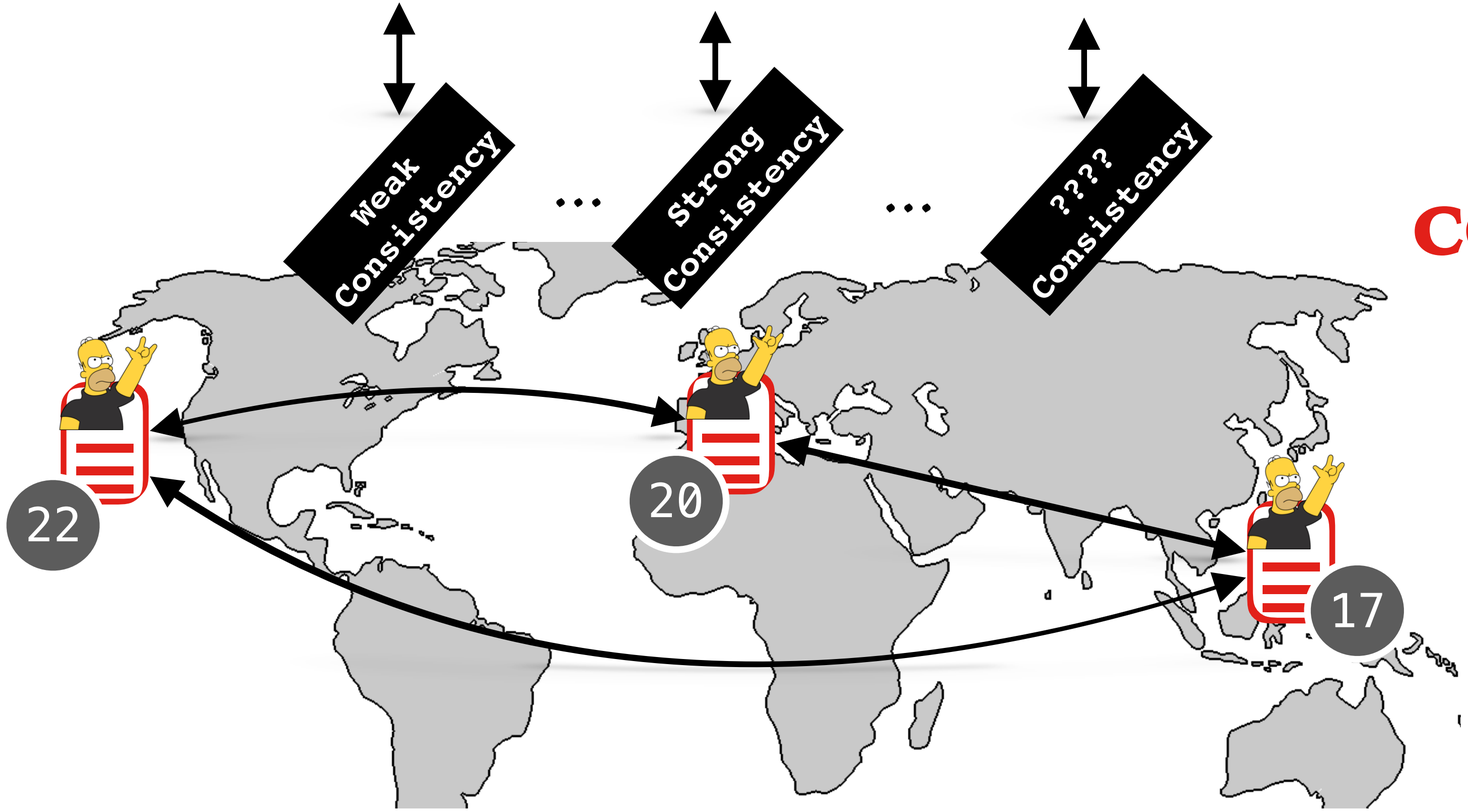
SOCIAL MEDIA APPLICATION

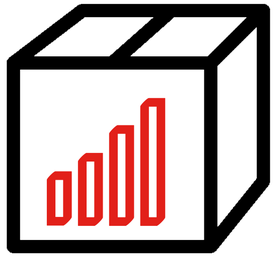
ABSTRACTION
FOR REPLICATED OBJECTS



SOCIAL MEDIA APPLICATION

ABSTRACTION
FOR REPLICATED OBJECTS





CORRECTABLE

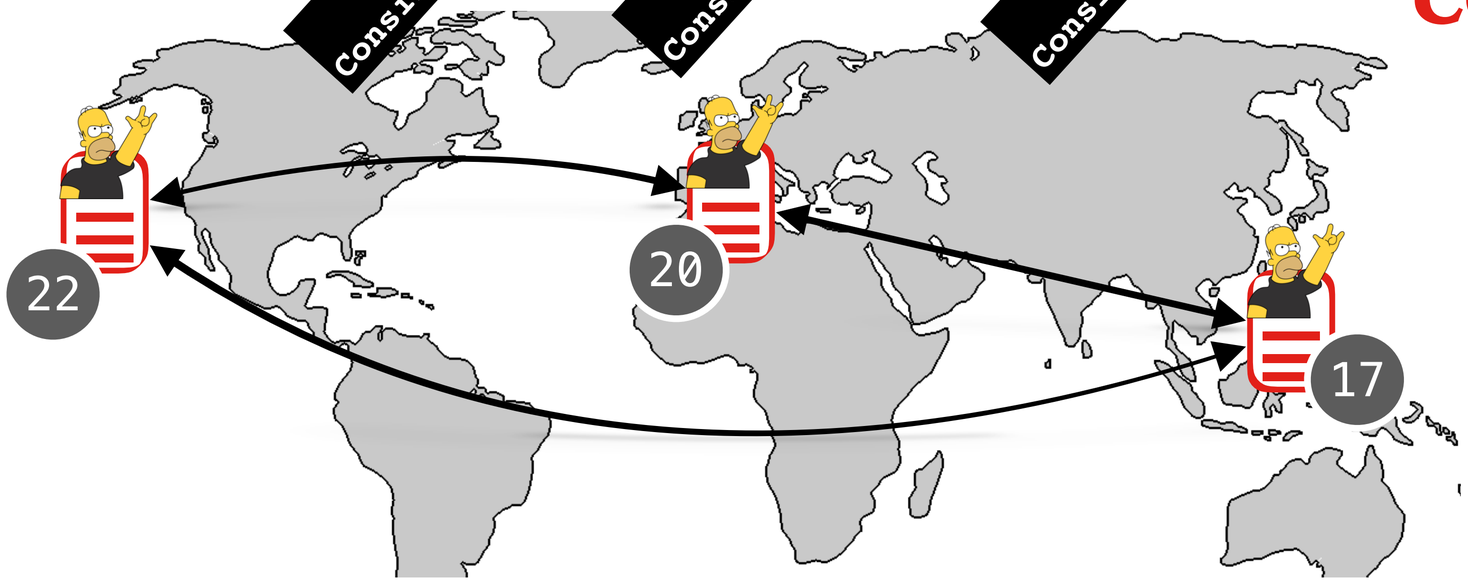
SOCIAL MEDIA APPLICATION

ABSTRACTION
FOR REPLICATED OBJECTS

Weak Consistency

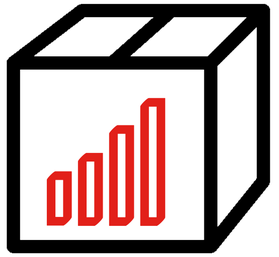
Strong Consistency

???? Consistency



Incremental Consistency Guarantees (ICG)

provides



CORRECTABLE

Correctables / **Design**

- Starting point: **Promises**
  - Placeholders for values
  - Becoming mainstream

Futures and Promises

Scala

finagle

JS

then Promises/A+

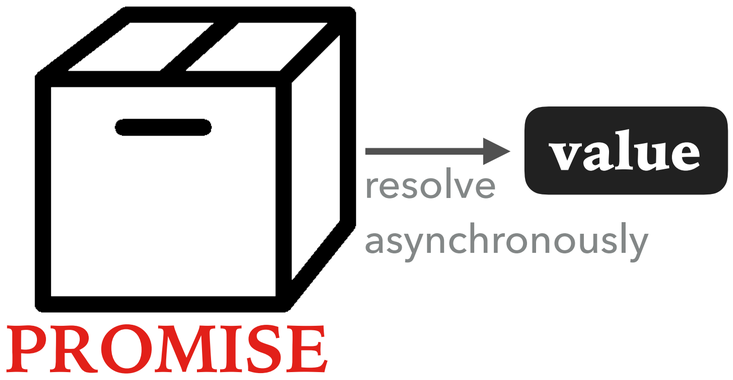
June 19, 2015 BACKEND

Futures for C++11 at Facebook

Hans Fugal

Google Guava

Core libraries for Java & Android



Correctables / **Design**

- Starting point: **Promises**
  - Placeholders for values
  - Becoming mainstream

Futures and Promises

Scala

finagle

JS

then Promises/A+

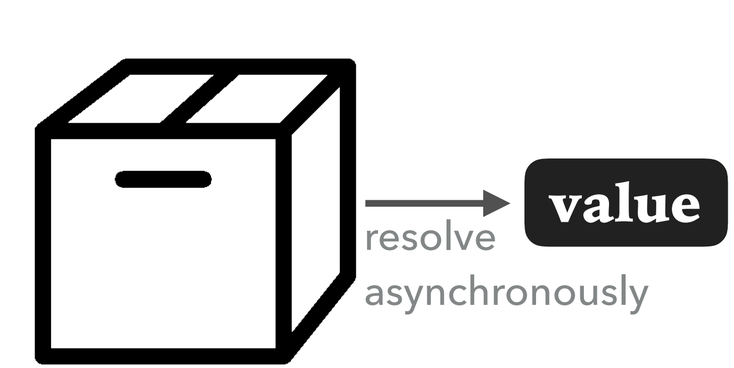
June 19, 2015 BACKEND

Futures for C++11 at Facebook

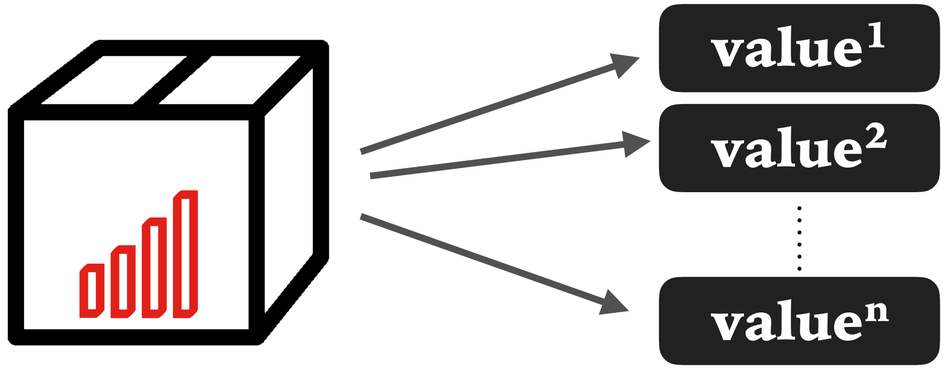
Hans Fugal

Google Guava

Core libraries for Java & Android



PROMISE



CORRECTABLE

Correctables / **Design**

- Starting point: **Promises**
  - Placeholders for values
  - Becoming mainstream

Futures and Promises

Scala

finagle

JS

then Promises/A+

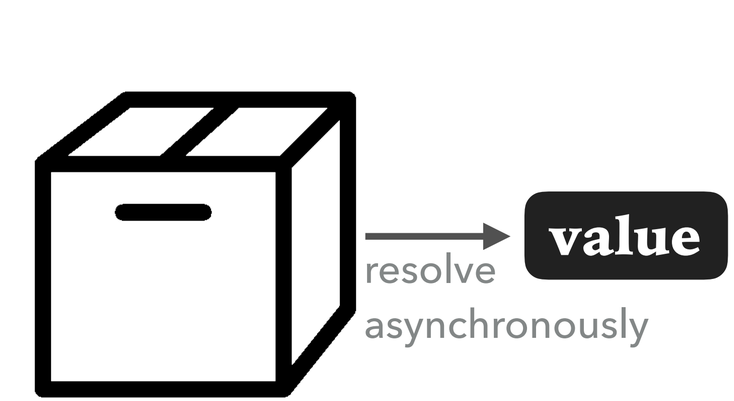
June 19, 2015 BACKEND

Futures for C++11 at Facebook

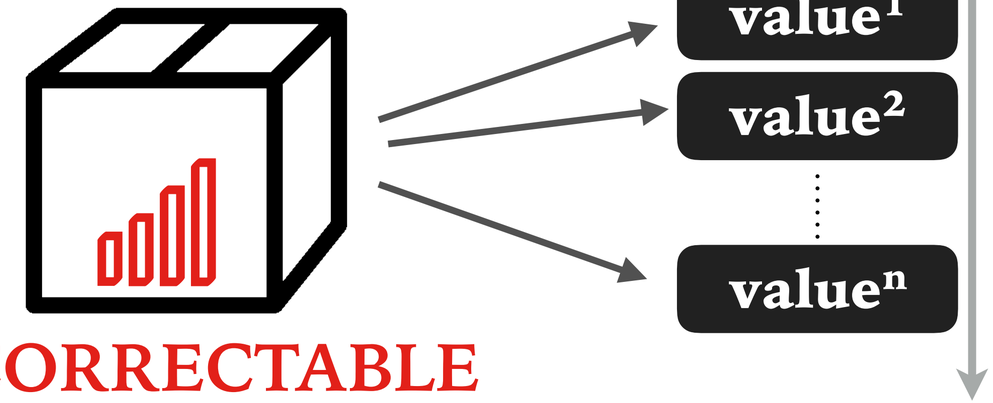
Hans Fugal

Google Guava

Core libraries for Java & Android



PROMISE

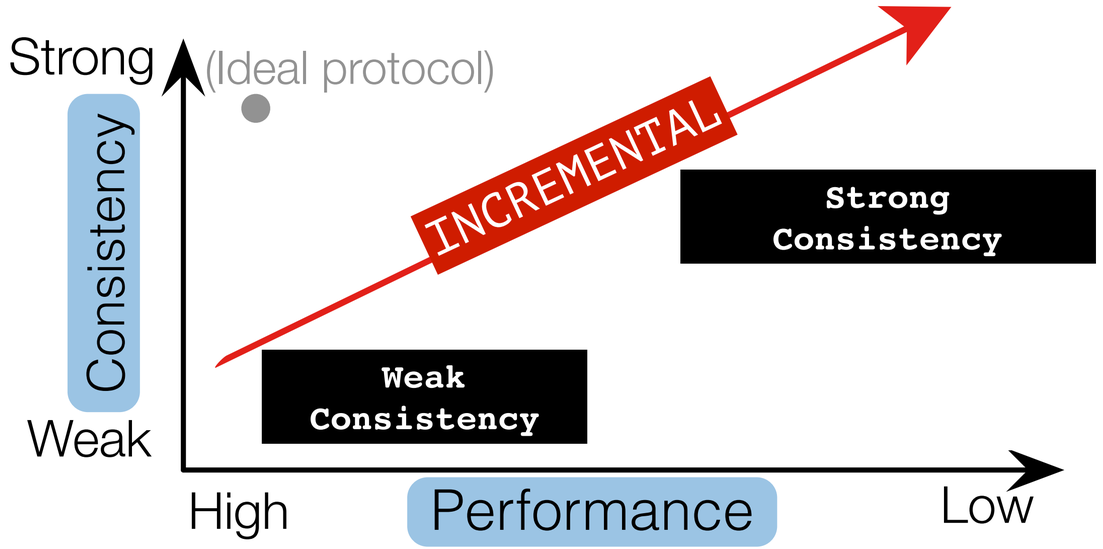


CORRECTABLE

PROGRESSIVELY STRONGER CONSISTENCY (INCREMENTAL)

PROGRESSIVELY HIGHER LATENCY

Consistency Models are **Complementary**



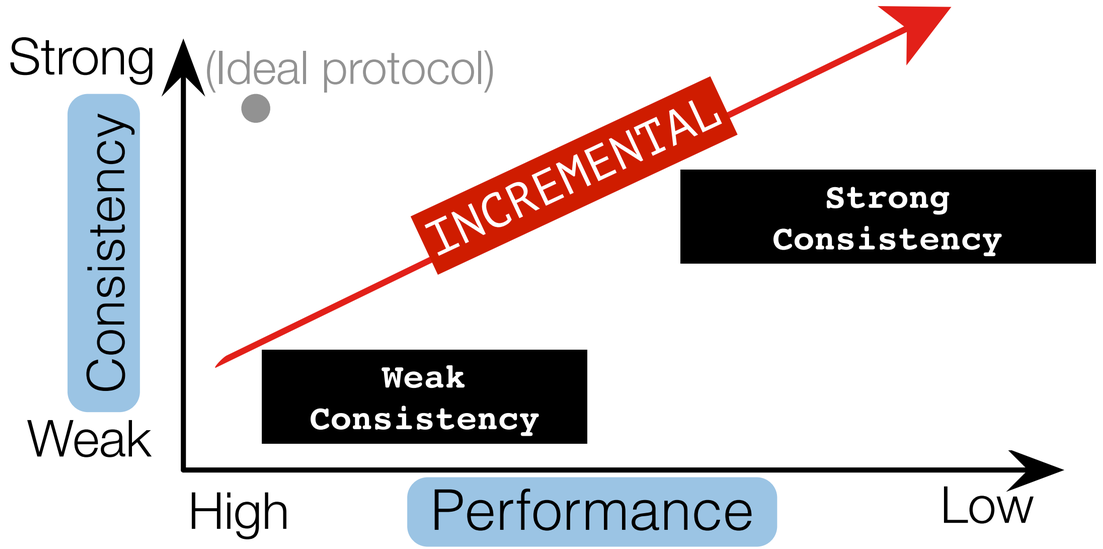
Weak consistency:

- ★ Fast
- ★ (Often correct)

Strong consistency:

- ★ Slower
- ★ (Correct with certainty)

Consistency Models are **Complementary**



Weak consistency:

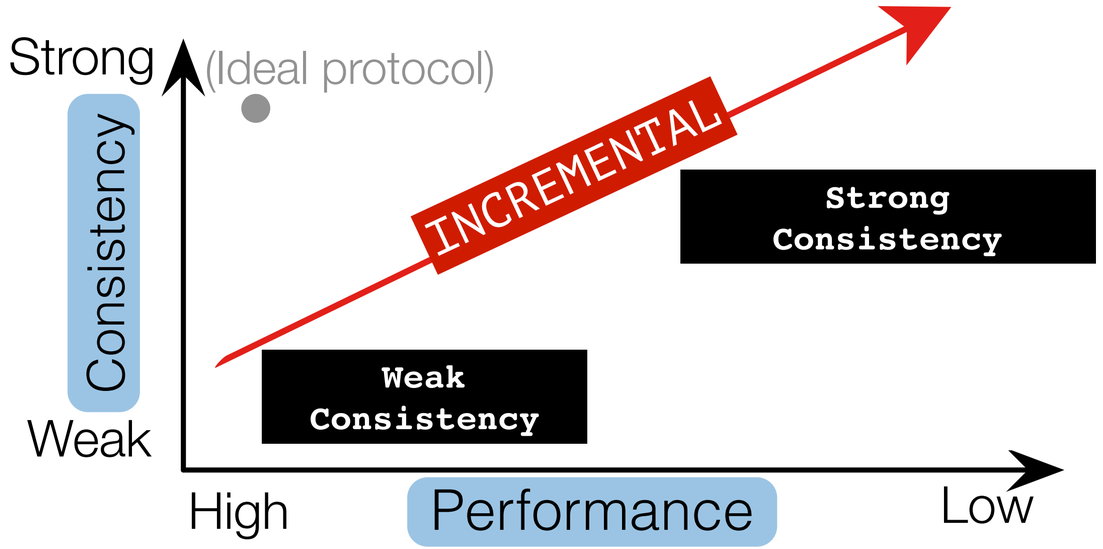
- ★ Fast
- ★ (Often correct)

Strong consistency:

- ★ Slower
- ★ (Correct with certainty)

So what?

Consistency Models are **Complementary**



Weak consistency:

- ★ Fast
- ★ (Often correct)

Strong consistency:

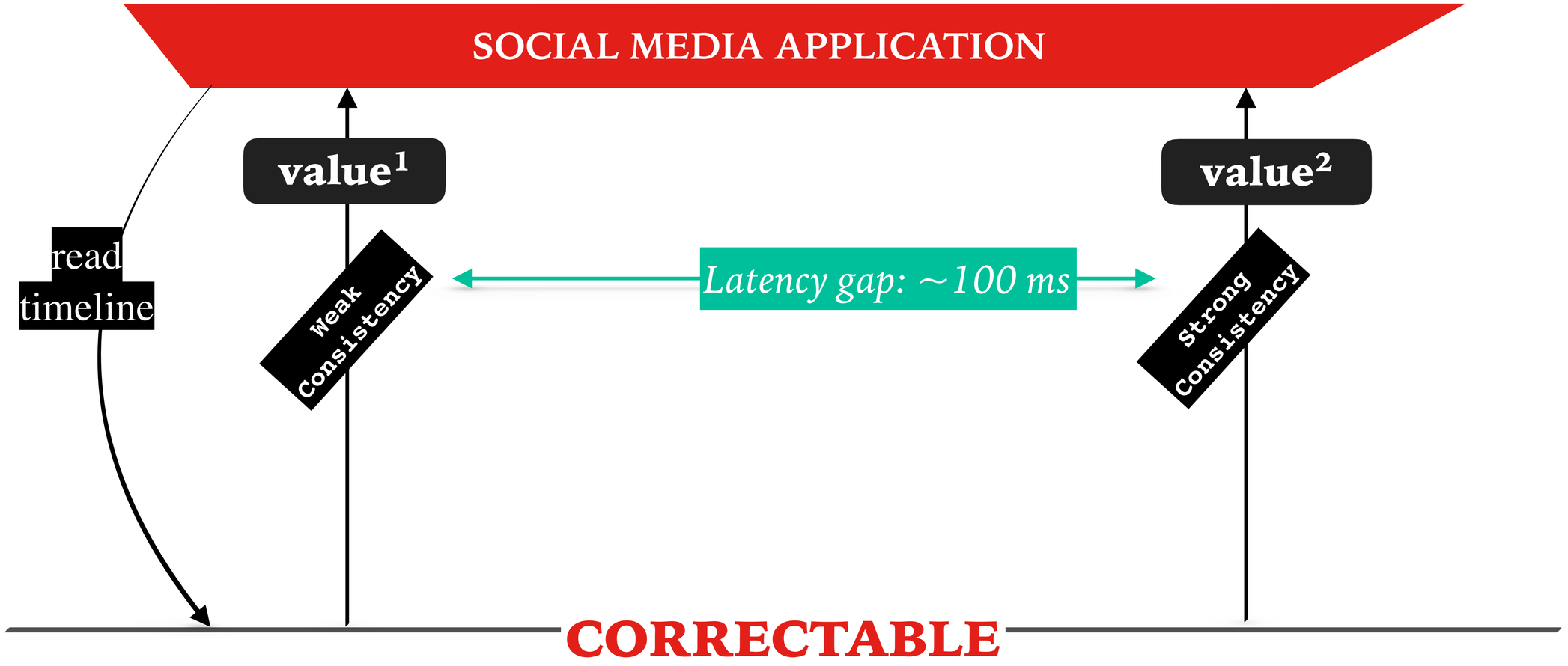
- ★ Slower
- ★ (Correct with certainty)

So what?

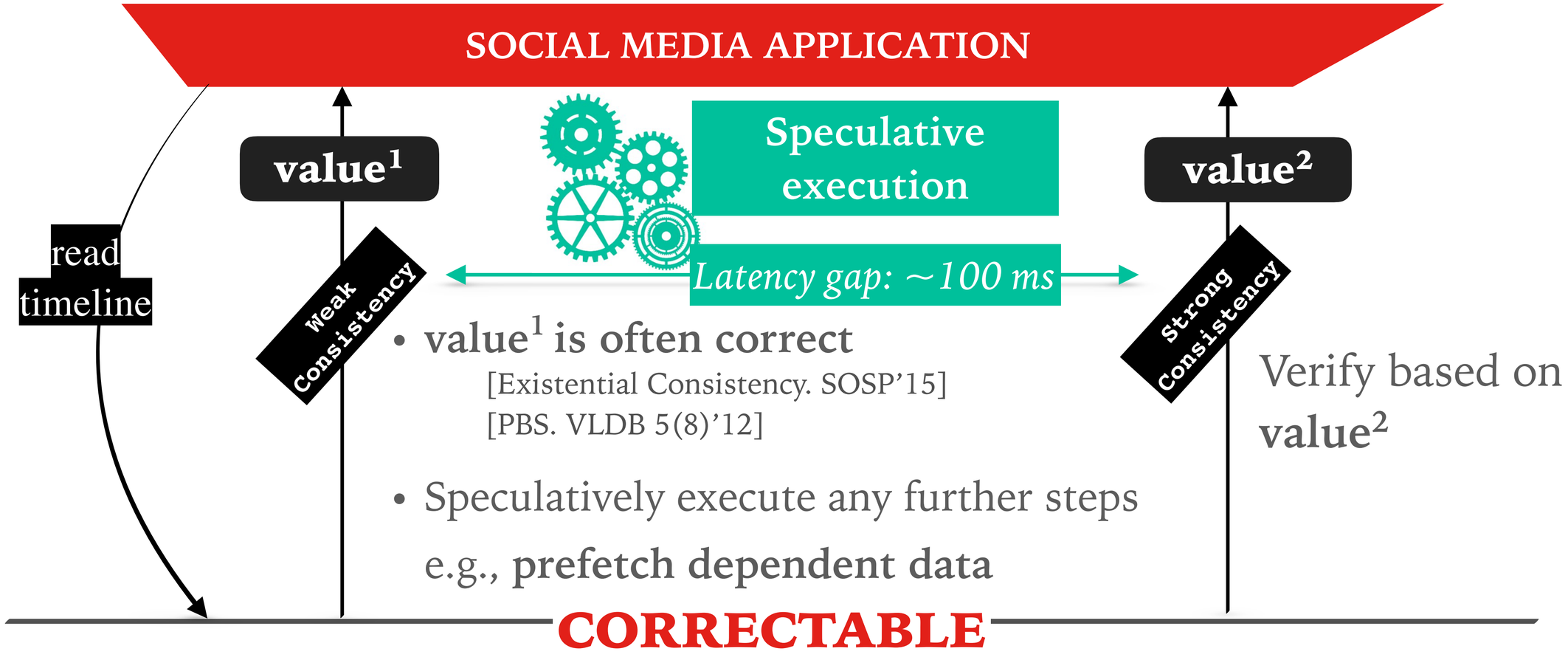
Latency optimizations

Speculating with Correctables

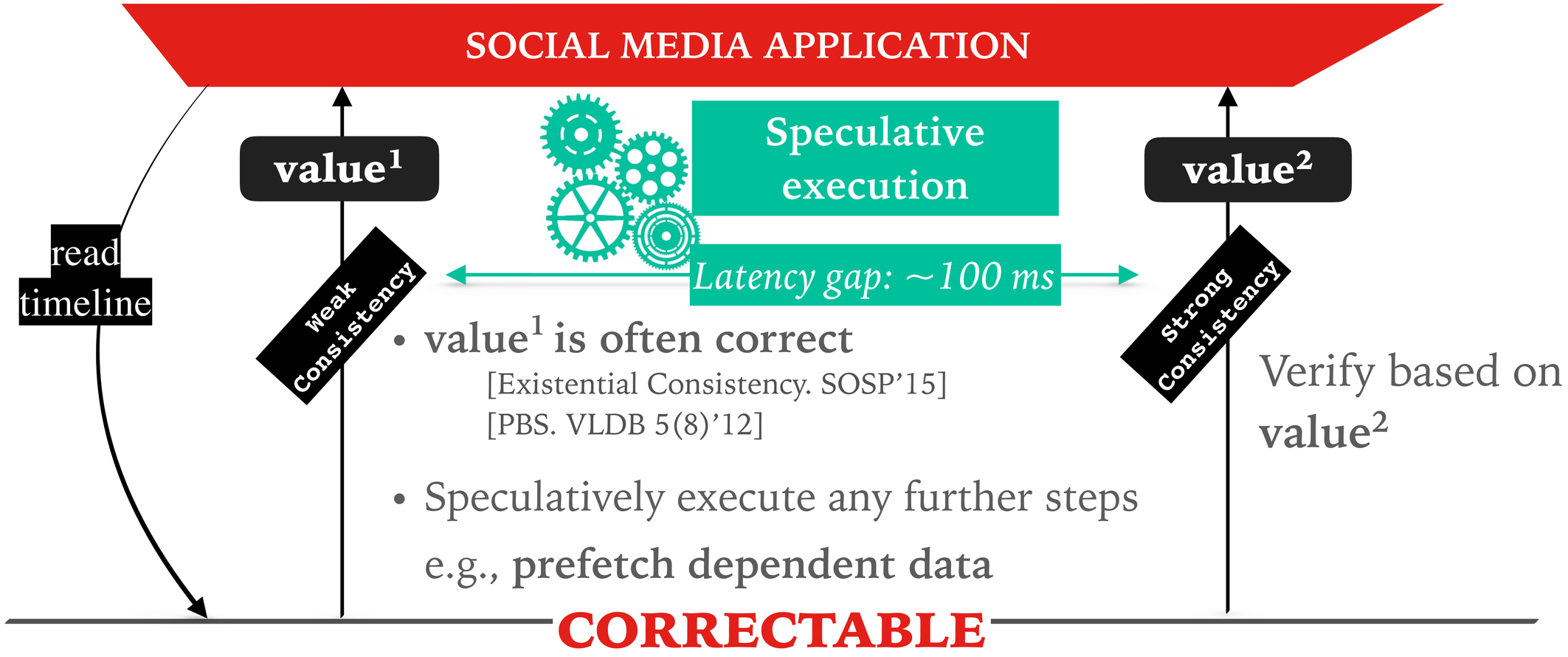
Speculating with Correctables



Speculating with Correctables

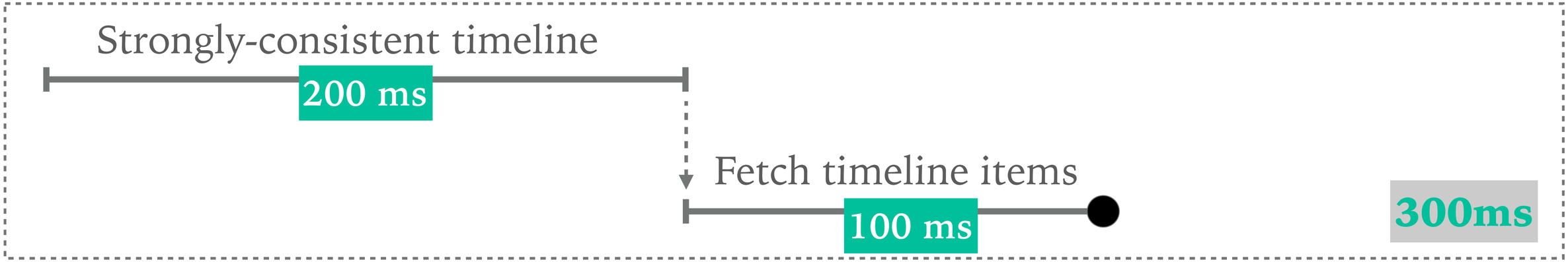


Speculating with Correctables

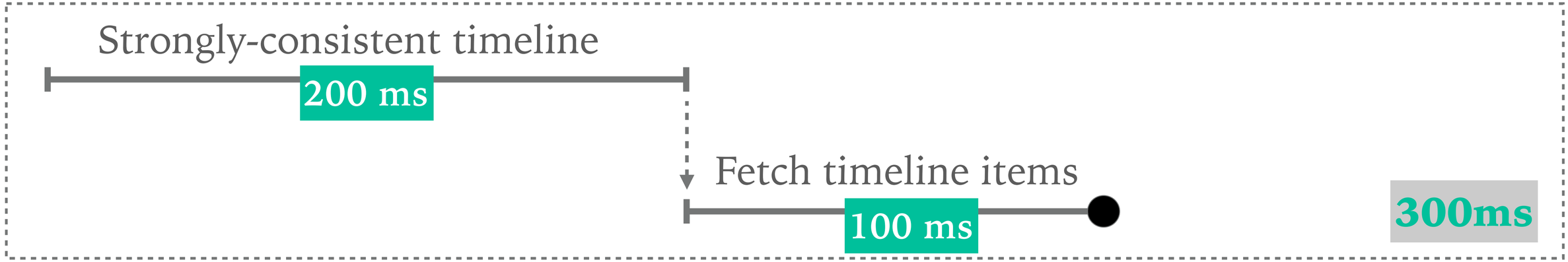


Lower latency of strong consistency

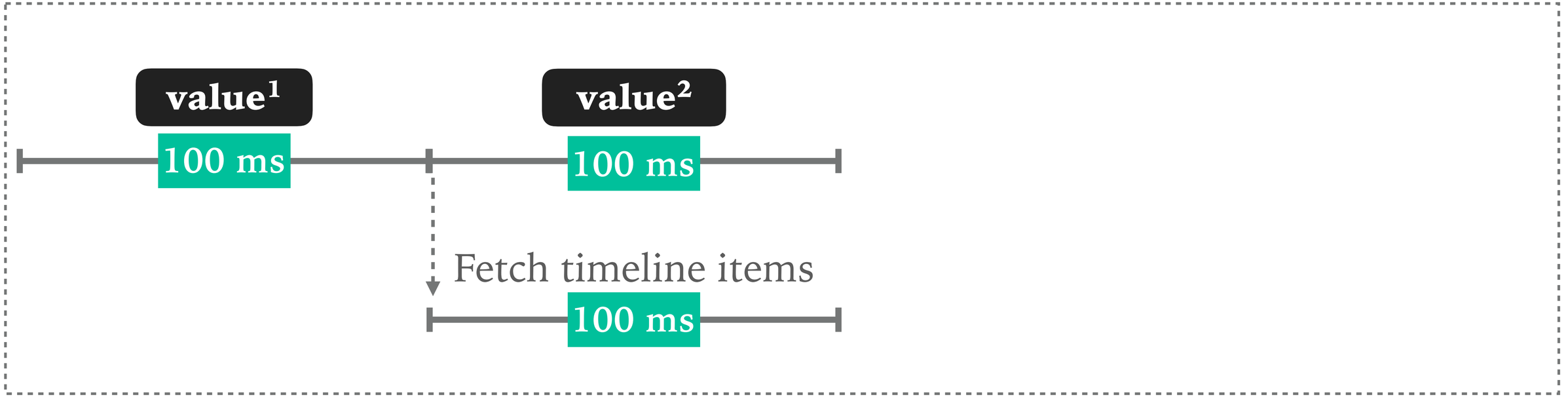
Traditional operation:



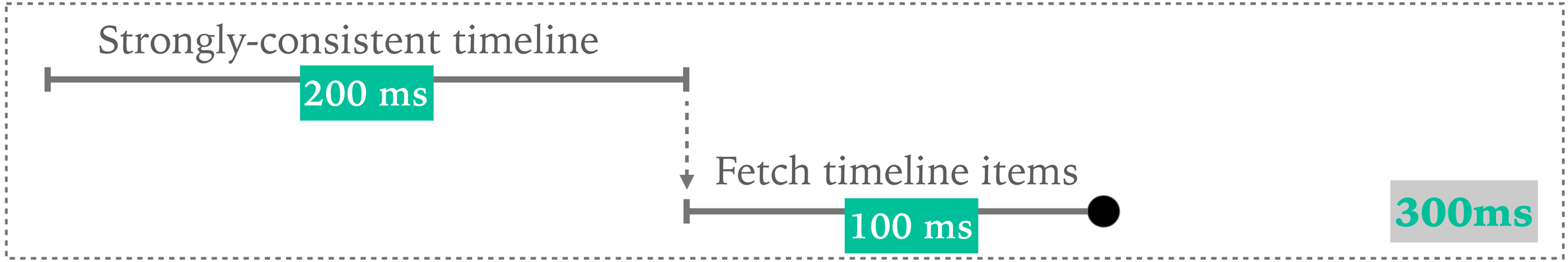
Traditional operation:



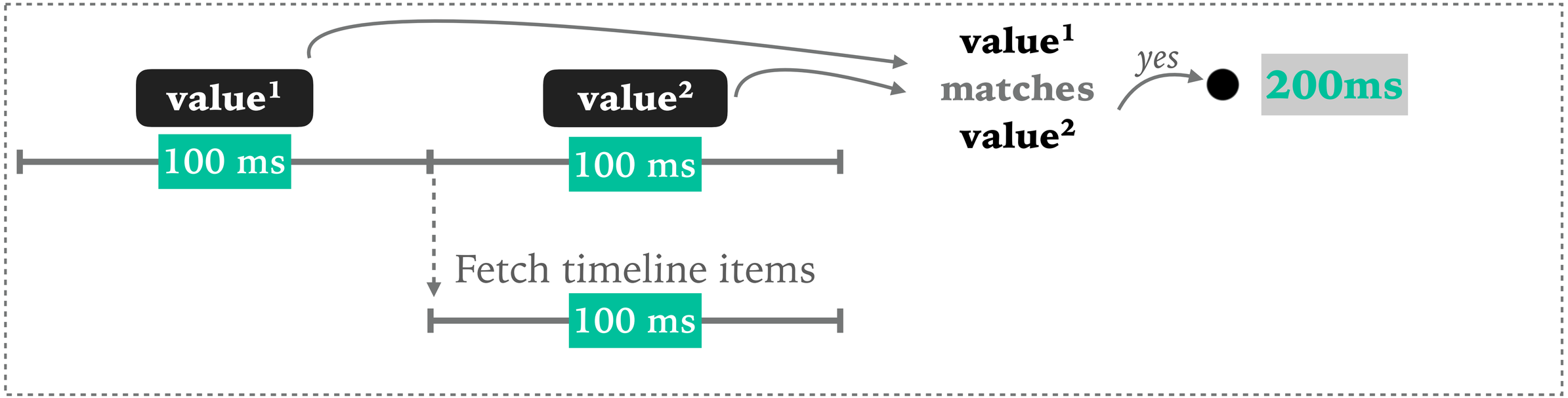
Speculative operation with ICG:



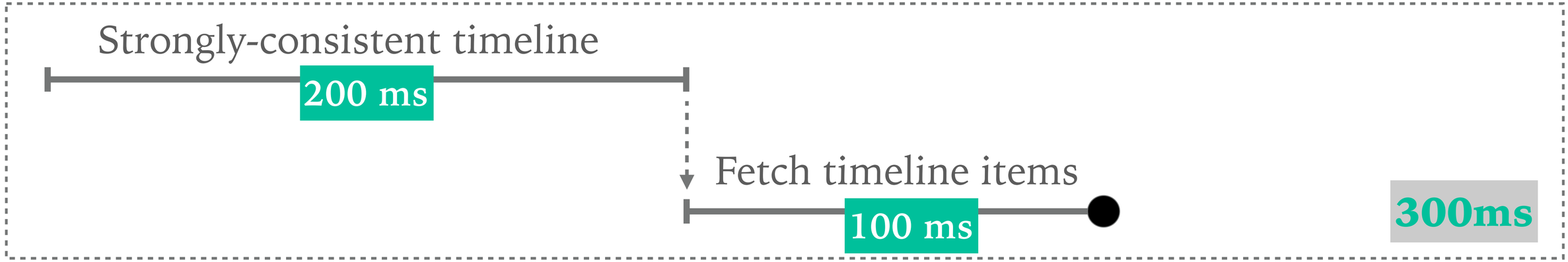
Traditional operation:



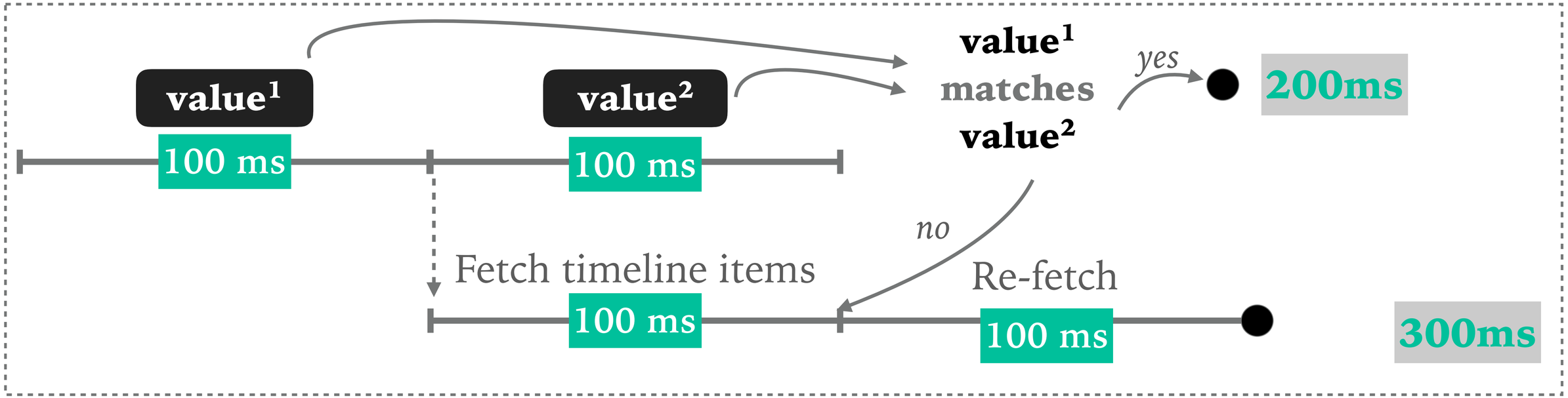
Speculative operation with ICG:



Traditional operation:



Speculative operation with ICG:



Speculation case-study

- Application: **Twissandra**
- Workload generated via YCSB
- Clients in Ireland
- Geo-replication on Amazon's EC2

Speculation case-study

- Application: **Twissandra**
- Workload generated via YCSB
- Clients in Ireland
- Geo-replication on Amazon's EC2

Incremental Consistency Guarantees for Replicated Objects

Rachid Guerraoui, Matej Pavlovic, and Dragos-Adrian Seredinschi*

School of Computer and Communication Sciences,
École Polytechnique Fédérale de Lausanne (EPFL), Switzerland
{rachid.guerraoui, matej.pavlovic, dragos-adrian.seredinschi}@epfl.ch

check the paper

★ Advertising System
— Speculation case-study

★ Ticket-selling System
— Exploiting application semantics

★ Overheads evaluation
& Optimizations

★ Latency gaps between consistency models

Decreasing latency of strong consistency

What is the
latency of the **fetch_timeline()** operation?

Decreasing latency of strong consistency

What is the
latency of the **fetch_timeline()** operation?

Baseline

Read using a quorum of 2/3 replicas

vs.

ICG

1. *Weak:* Read with 1/3 replicas
2. *"Strong:"* Read with quorum of 2/3 replicas

Decreasing latency of strong consistency

What is the
latency of the **fetch_timeline()** operation?

Workload A (50:50 read/write) **Workload B (95:5 read/write)** **Workload C (read-only)**

Baseline

Read using a quorum of 2/3 replicas

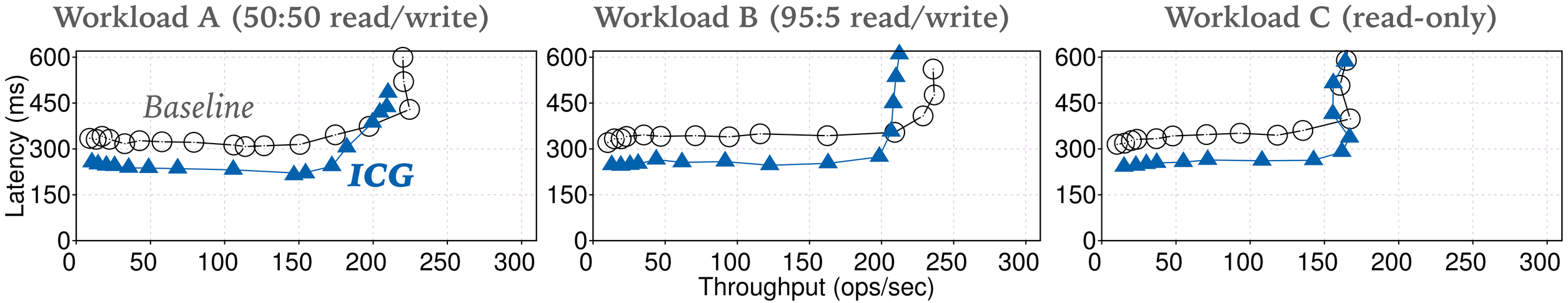
vs.

ICG

1. *Weak:* Read with 1/3 replicas
2. *"Strong:"* Read with quorum of 2/3 replicas

Decreasing latency of strong consistency

What is the
latency of the **fetch_timeline()** operation?



Baseline

Read using a quorum of 2/3 replicas

vs.

ICG

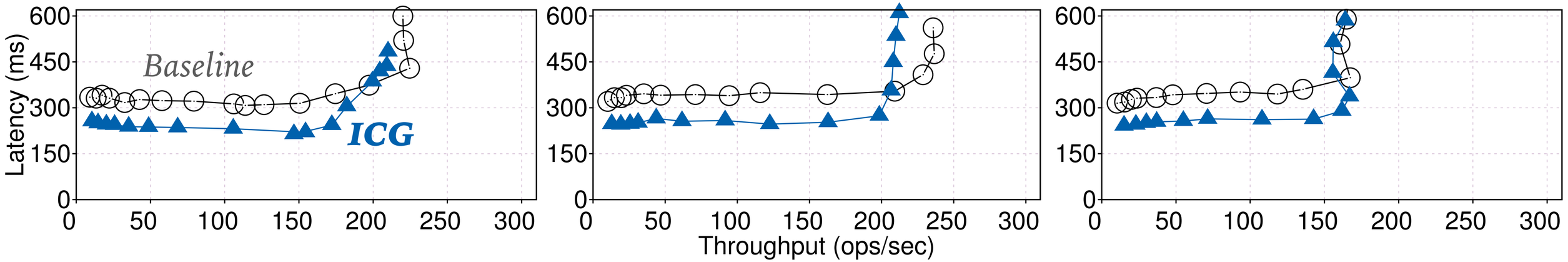
1. *Weak:* Read with 1/3 replicas
2. *"Strong:"* Read with quorum of 2/3 replicas

Decreasing latency of strong consistency

What is the
latency of the **fetch_timeline()** operation?

- ★ ***Latency decrease by 40%***
- ★ *Throughput drop by 6%*
- ★ *Same consistency model (2/3 replicas)*

Workload A (50:50 read/write) | **Workload B (95:5 read/write)** | **Workload C (read-only)**



Baseline

Read using a quorum of 2/3 replicas

vs.

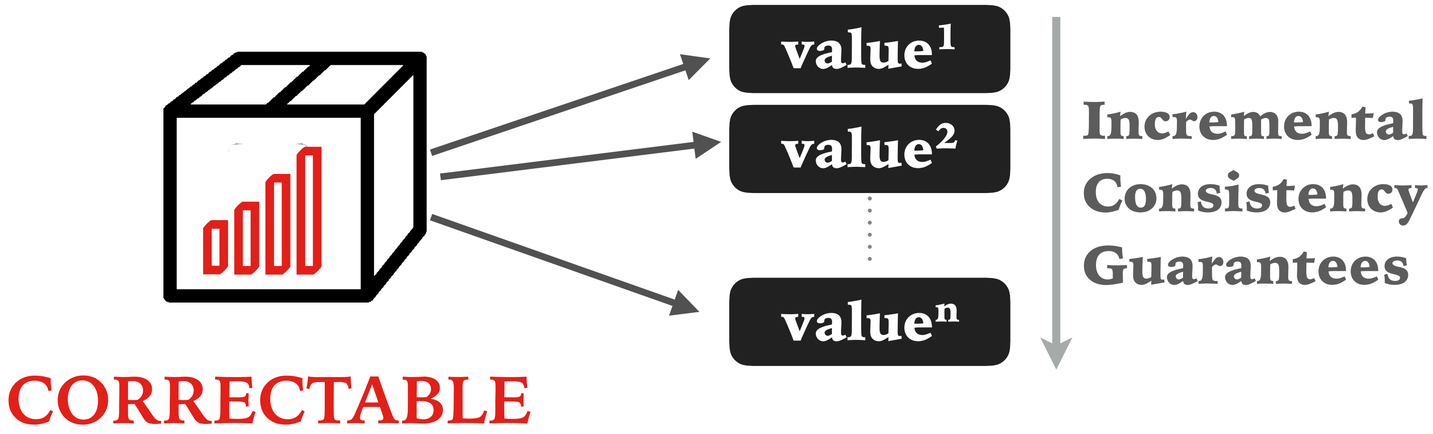
ICG

1. *Weak:* Read with 1/3 replicas
2. *"Strong:"* Read with quorum of 2/3 replicas

Conclusion

The Correctables abstraction enables you to:

1. **Leverage consistency models incrementally**
2. **Lower latency of strong consistency**



@adizere

backup slides

Speculation // Syntactic sugar

```
1 invoke(read(...))
2   .speculate(speculationFunc[, abortFunc])
3   .setCallbacks(onFinal = (res) => deliver(res))
```

Listing 3: Generic speculation with Correctables. The square brackets indicate that `abortFunc` is optional.

Legacy code vs. Correctables

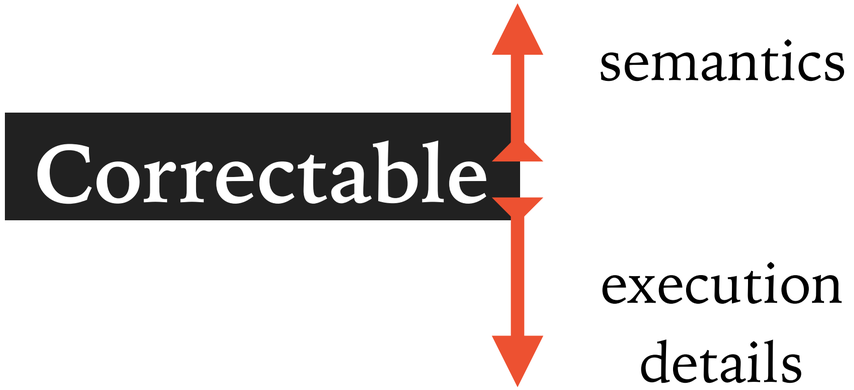
```
1  from pylons import app_globals as g # cache access
2  from r2.lib.db import queries       # backend access

4  def user_messages(user, update = False):
5    key = messages_key(user._id)
6    trees = g.permacache.get(key)
7    if not trees or update:
8        trees = user_messages_nocache(user)
9        g.permacache.set(key, trees) # cache coherence
10   return trees
11 def user_messages_nocache(user):
12   # Just like user_messages , but avoiding the cache...
```

Listing 1: Different consistency guarantees in Reddit [13], as an example of tight coupling between applications and storage. Developers must manually handle the cache and the backend.

```
1 def user_messages(user, strong = False):
2   key = messages_key(user._id)
3   # coherence handled by invoke* functions in bindings
4   if strong: return invokeStrong(get(key))
5        else: return invokeWeak(get(key))
```

Listing 2: Reddit code rewritten using Correctables.



Legacy code vs. Correctables

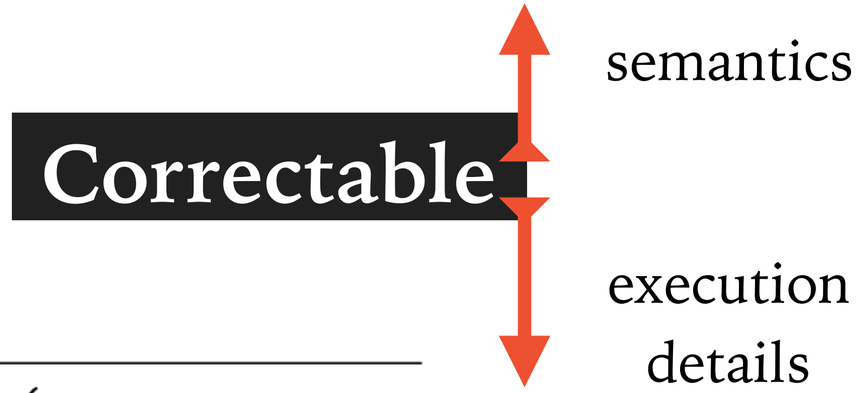
```
1  from pylons import app_globals as g # cache access
2  from r2.lib.db import queries       # backend access

4  def user_messages(user, update = False):
5    key = messages_key(user._id)
6    trees = g.permacache.get(key)
7    if not trees or update:
8        trees = user_messages_nocache(user)
9        g.permacache.set(key, trees) # cache coherence
10   return trees
11 def user_messages_nocache(user):
12   # Just like user_messages, but avoiding the cache...
```

Listing 1: Different consistency guarantees in Reddit [13], as an example of tight coupling between applications and storage. Developers must manually handle the cache and the backend.

```
1 def user_messages(user, strong = False):
2   key = messages_key(user._id)
3   # coherence handled by invoke* functions in bindings
4   if strong: return invokeStrong(get(key))
5        else: return invokeWeak(get(key))
```

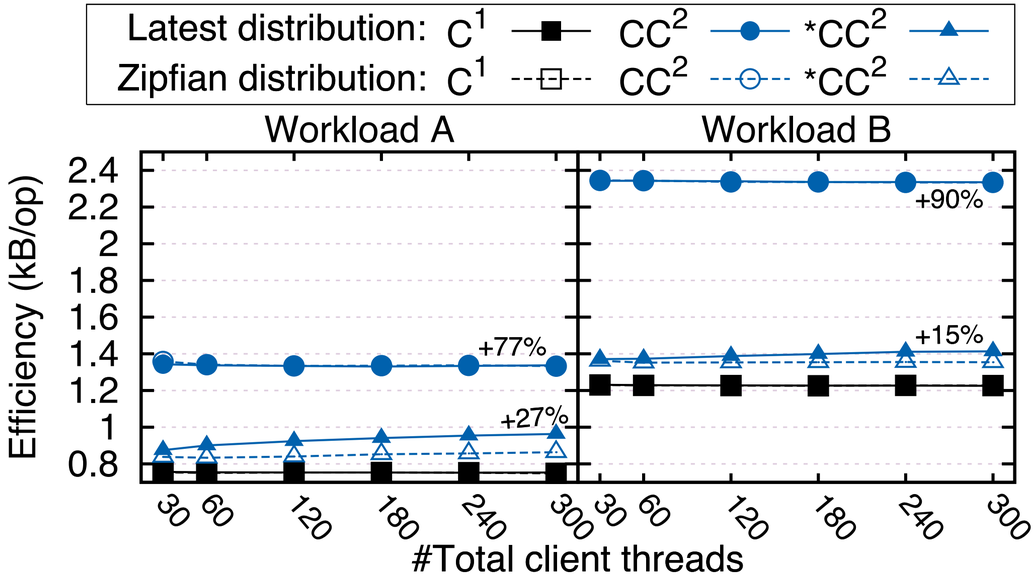
Listing 2: Reddit code rewritten using Correctables.




```
1 invoke(getLatestNews()).setCallbacks(
2     onUpdate = (items) => refreshDisplay(items))
```

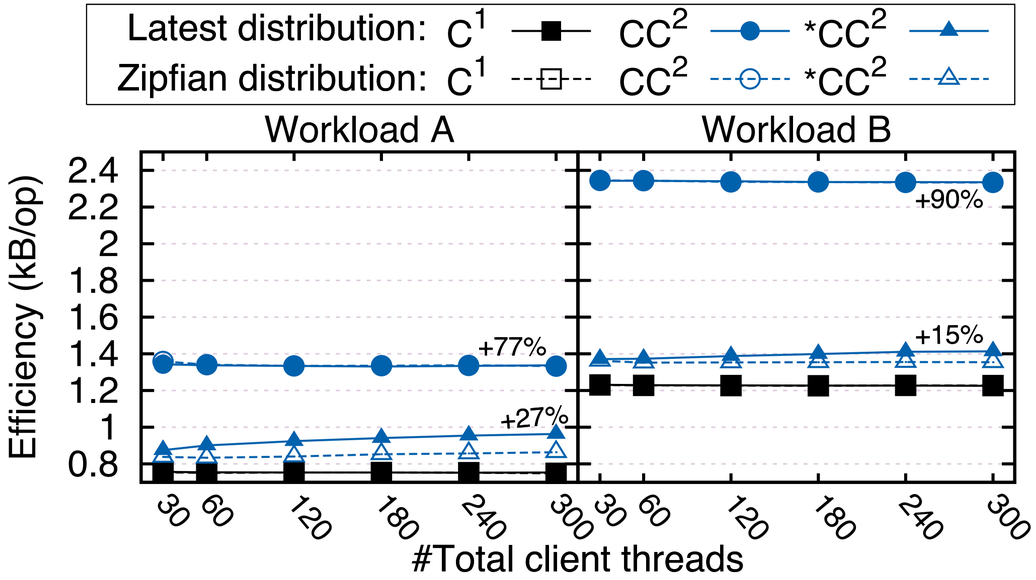
Listing 6: Progressive display of news items using Correctables. The `refreshDisplay` function triggers with every update on the news items.

Overheads



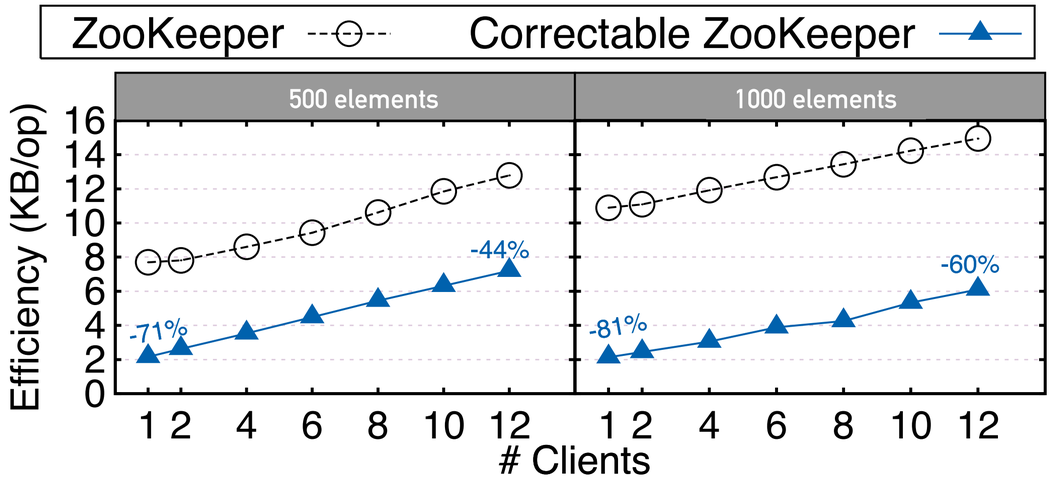
- Cassandra
- YCSB workload, various configurations
- Client in Ireland
- Replicas in Virginia, Frankfurt, and Ireland

Overheads



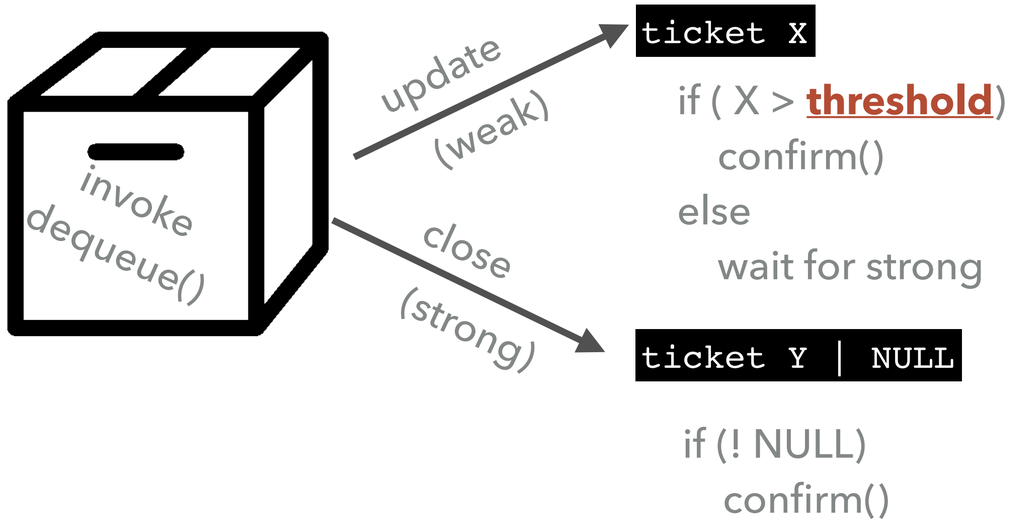
- Cassandra
- YCSB workload, various configurations
- Client in Ireland
- Replicas in Virginia, Frankfurt, and Ireland

- ZooKeeper queue implementation
- Wasteful implementation (by default)
- We were able to improve
  — negative overhead



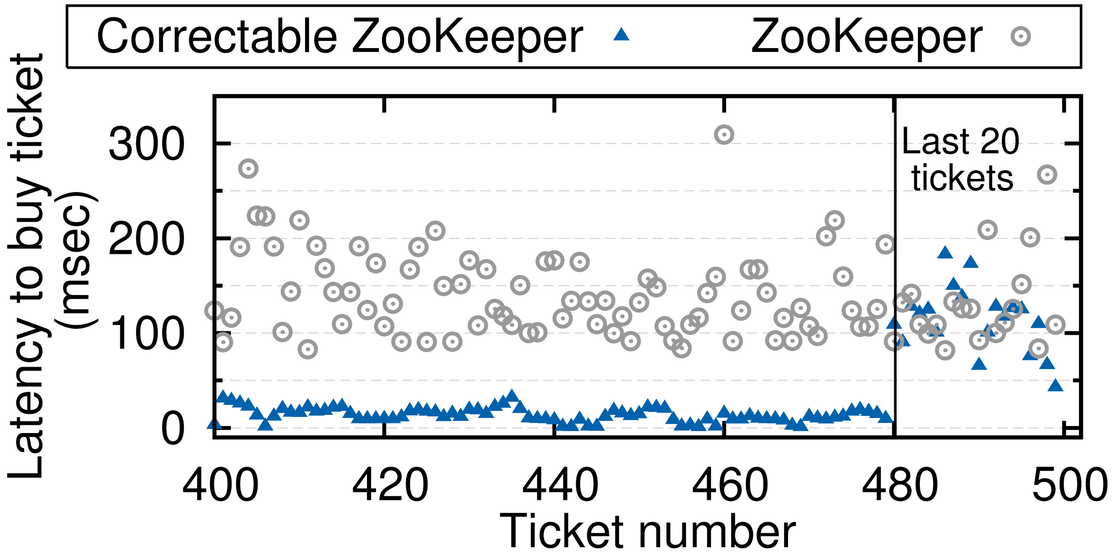
Exploiting application semantics

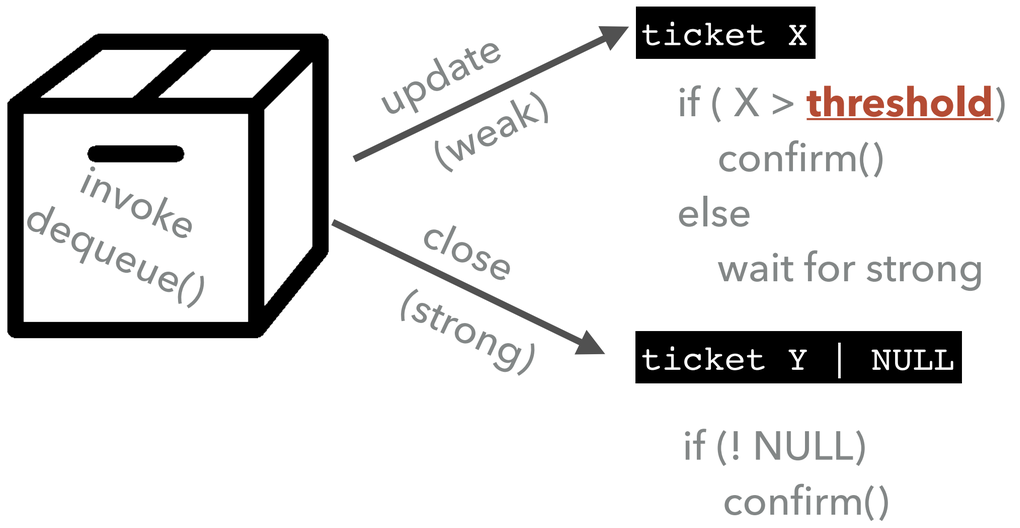
- Ticket selling application
  - Implemented through a ZooKeeper queue
  - Buy ticket = dequeue operation



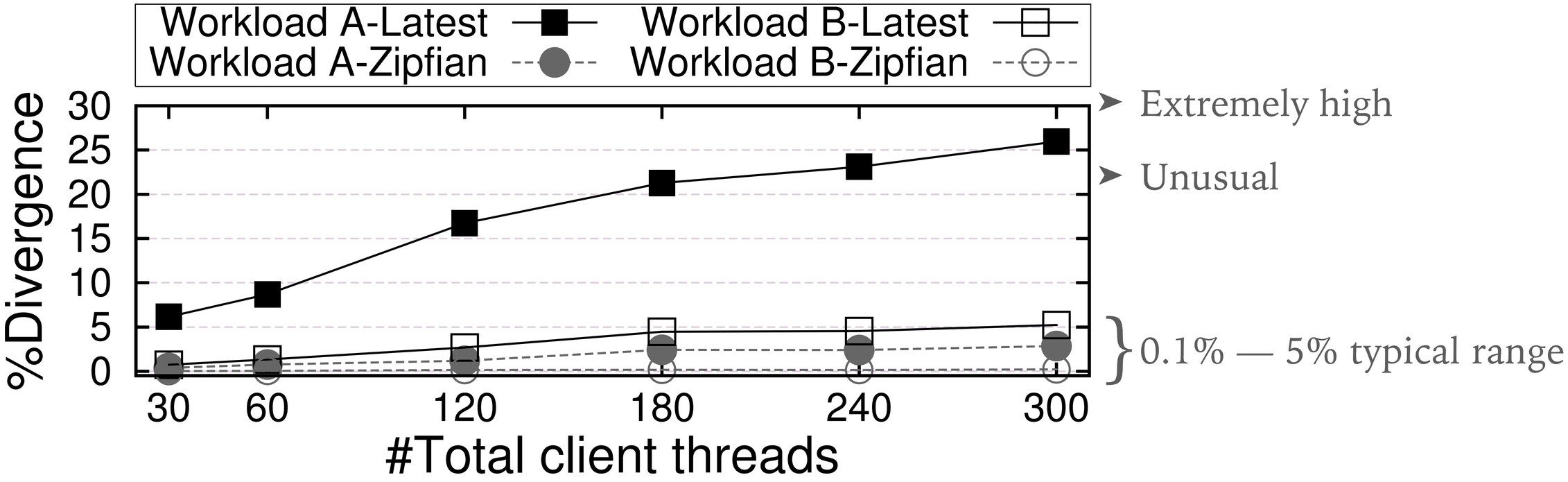
Exploiting application semantics

- Ticket selling application
  - Implemented through a ZooKeeper queue
  - Buy ticket = dequeue operation

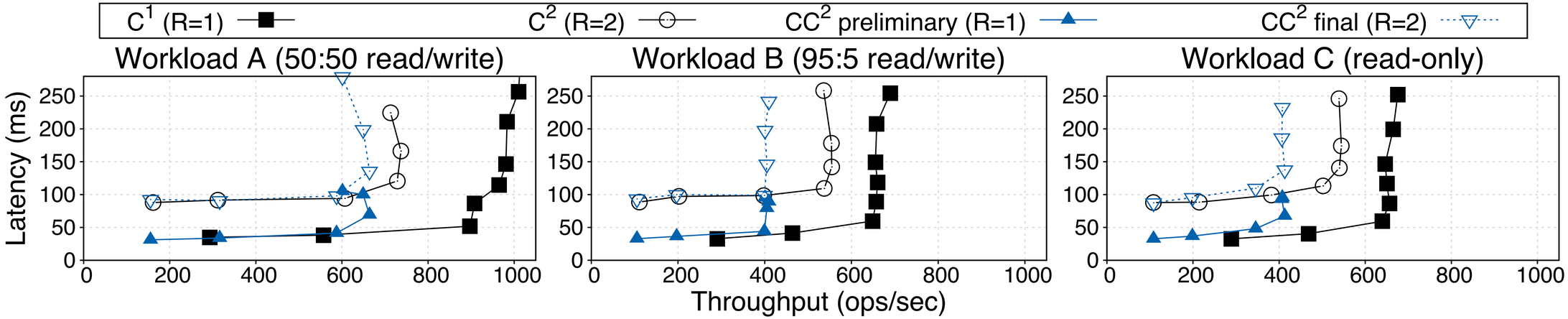


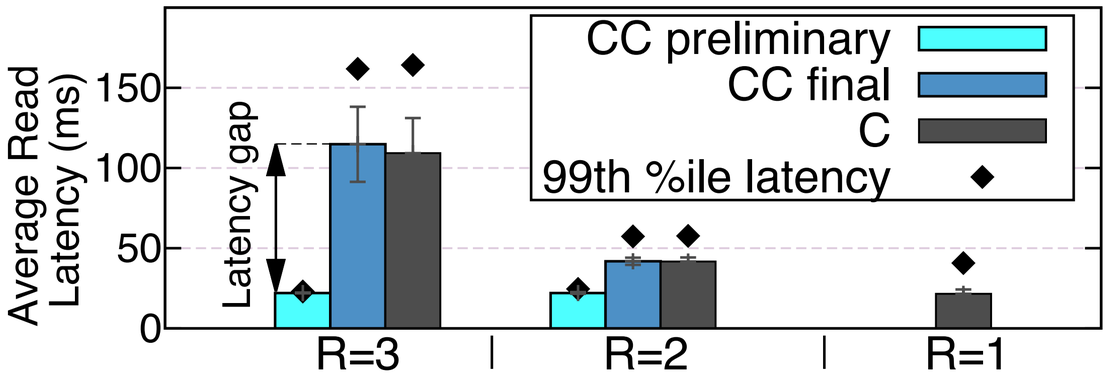


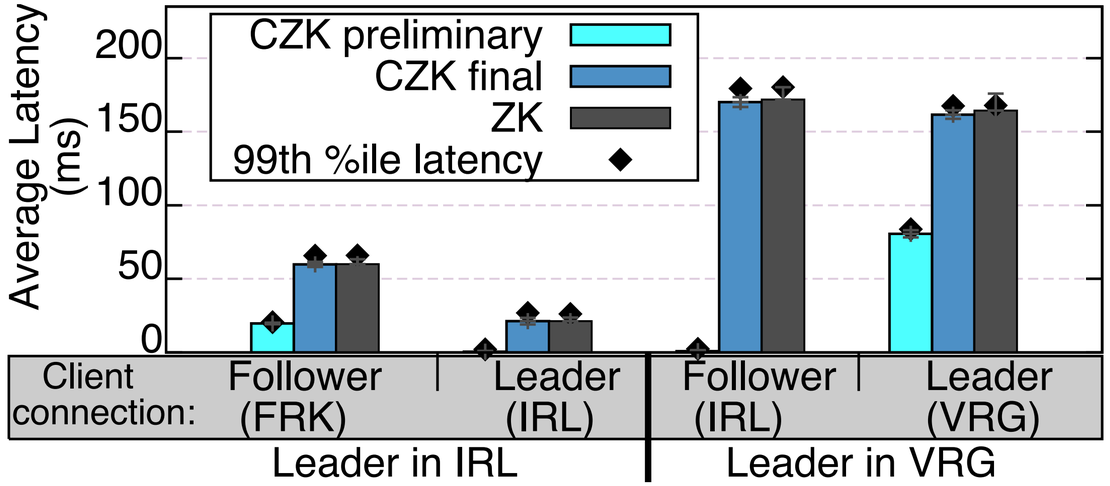
Divergence between weak and strong consistency



Latency gaps between consistency models







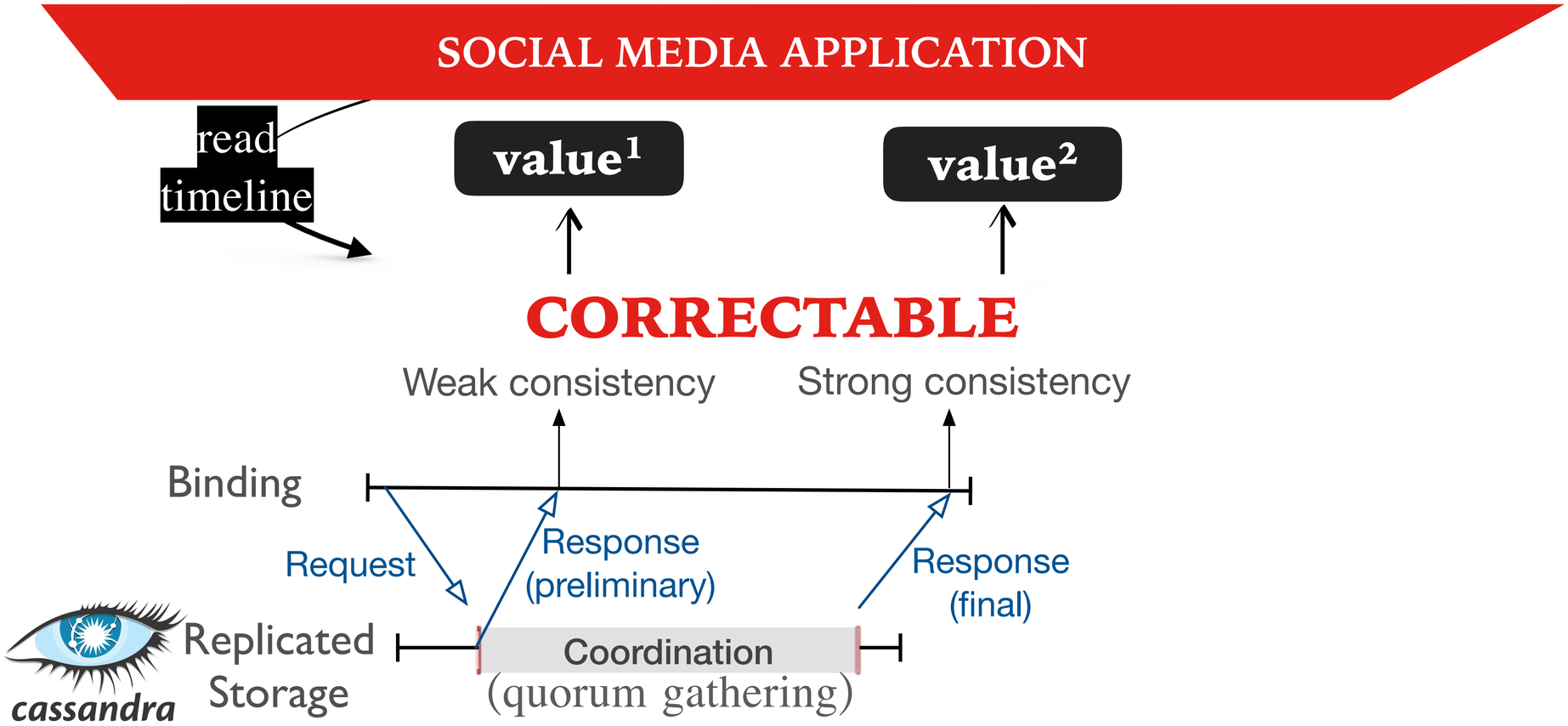
Efficiency of Multiple Responses

Efficiency of Multiple Responses

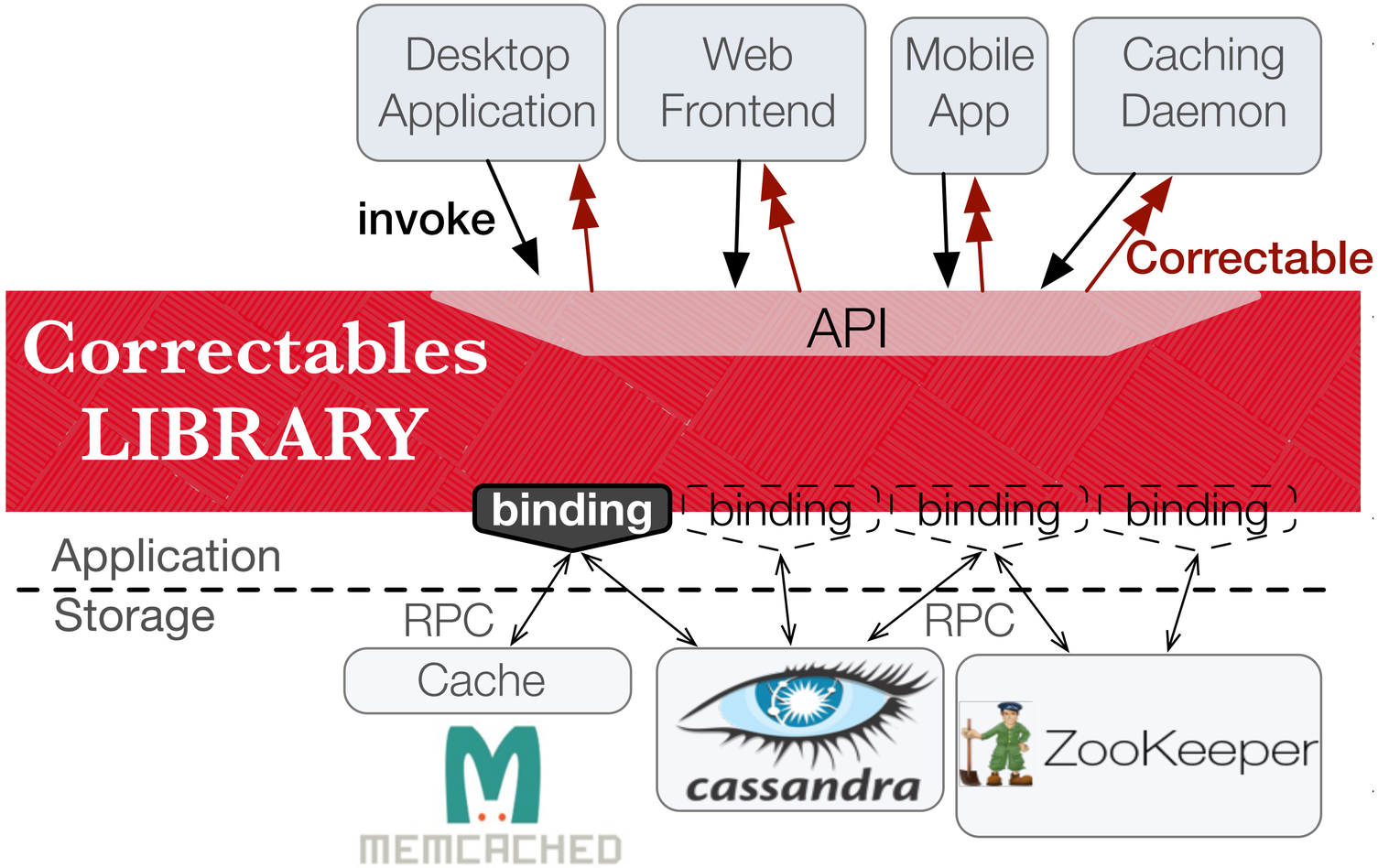
Binding

Replicated Storage

Efficiency of Multiple Responses



Correctables / **Library**



SimpleDB

MySQL