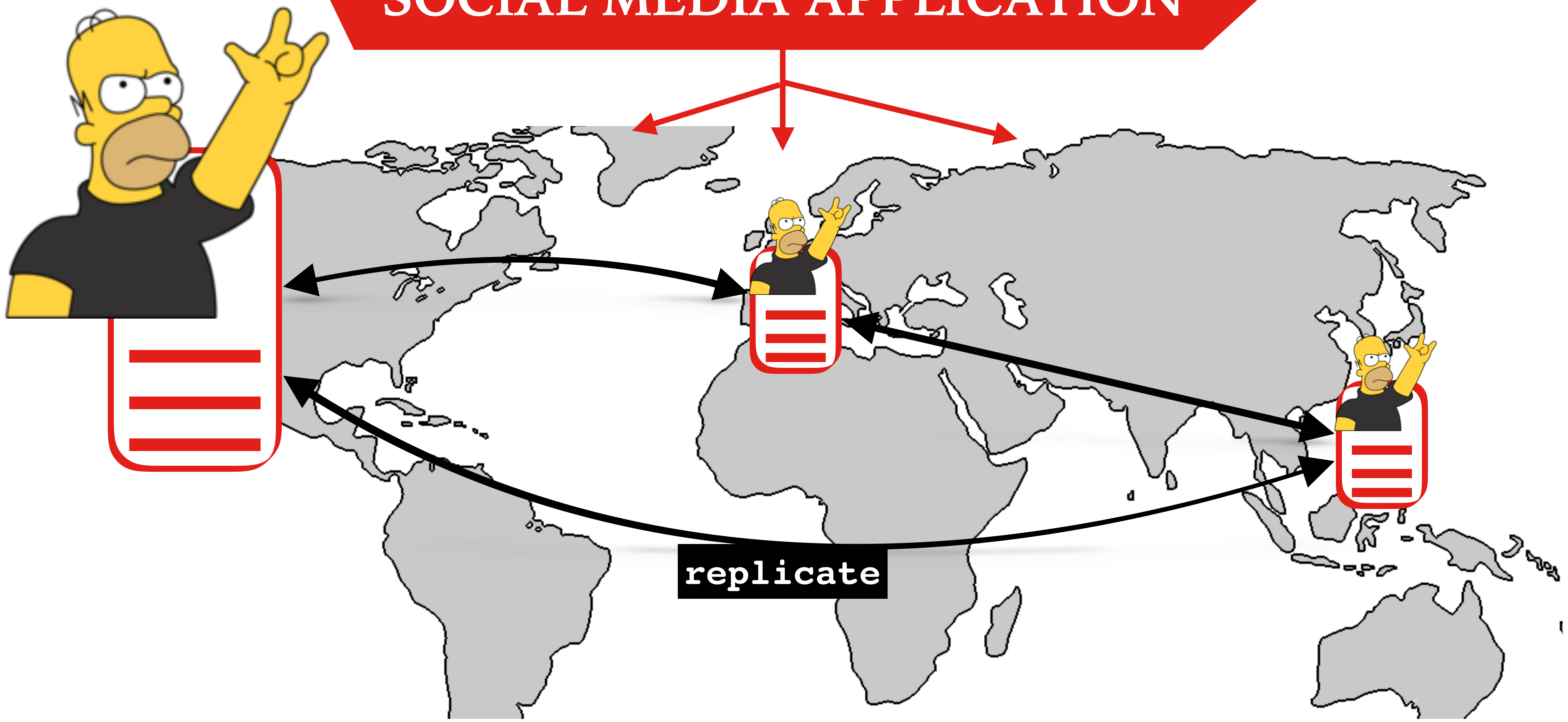
Incremental Consistency Guarantees

For Replicated Objects

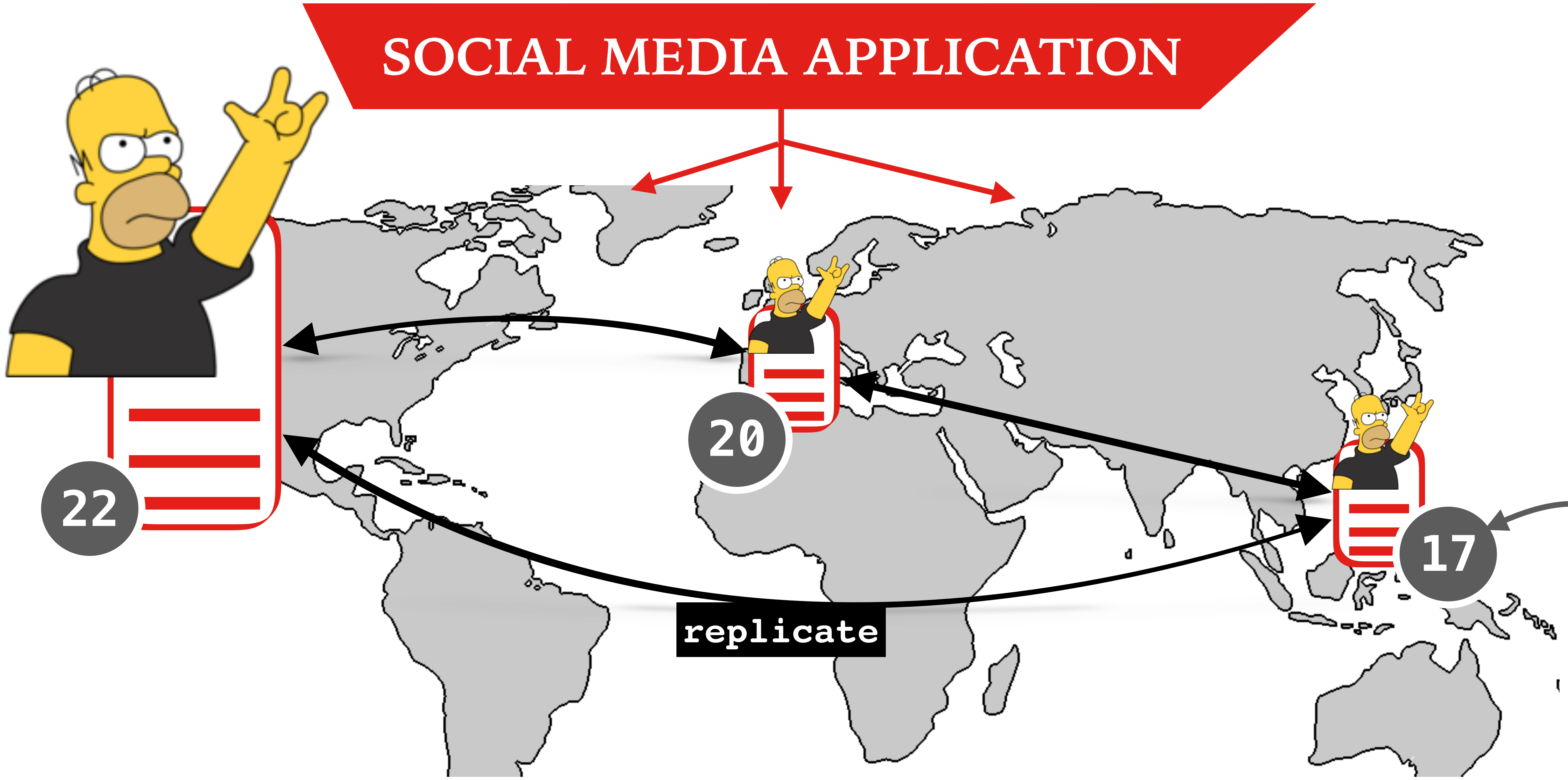
Rachid Guerraoui, Matej Pavlovic, Dragos-Adrian Seredinschi

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

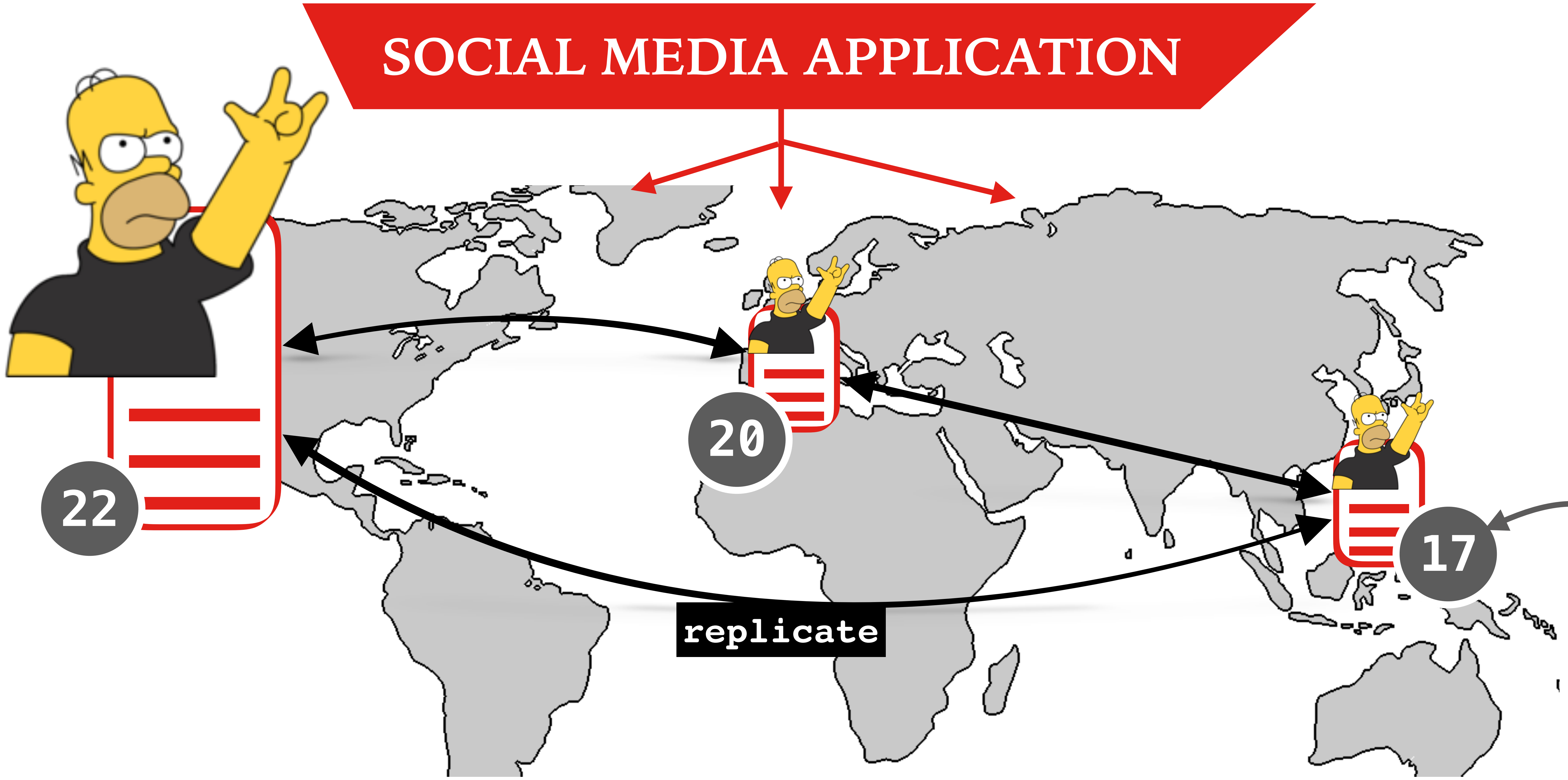
SOCIAL MEDIA APPLICATION



SOCIAL MEDIA APPLICATION



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



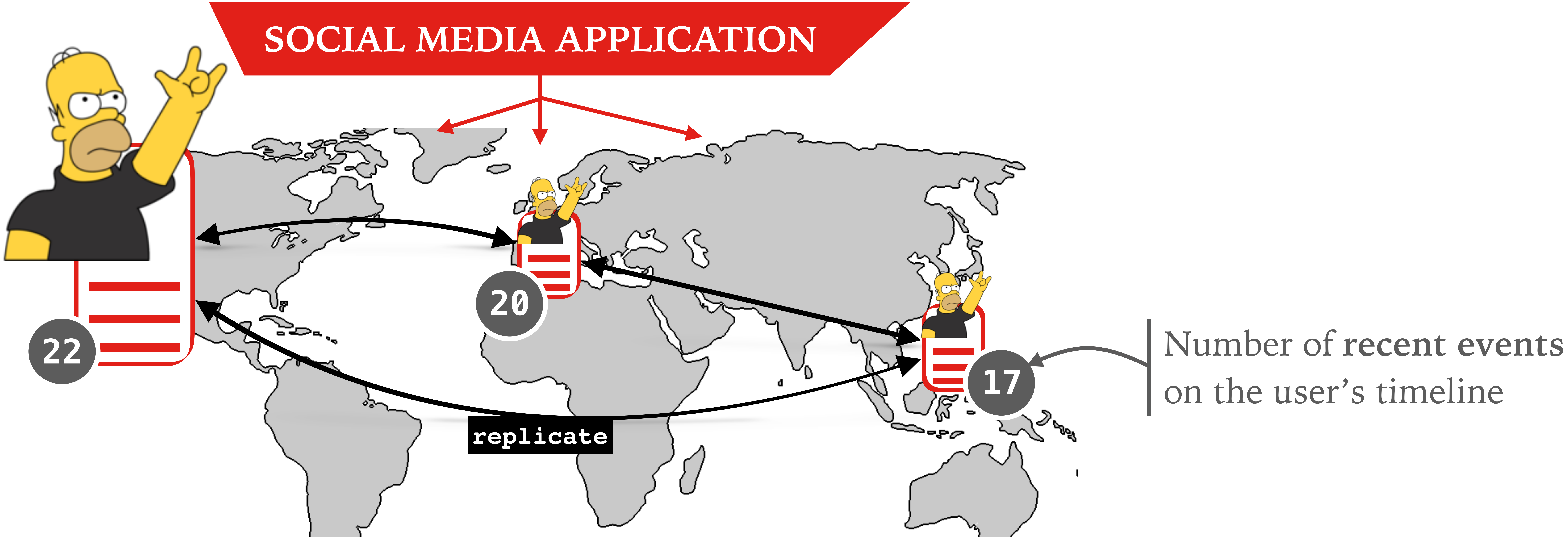
SOCIAL MEDIA APPLICATION

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

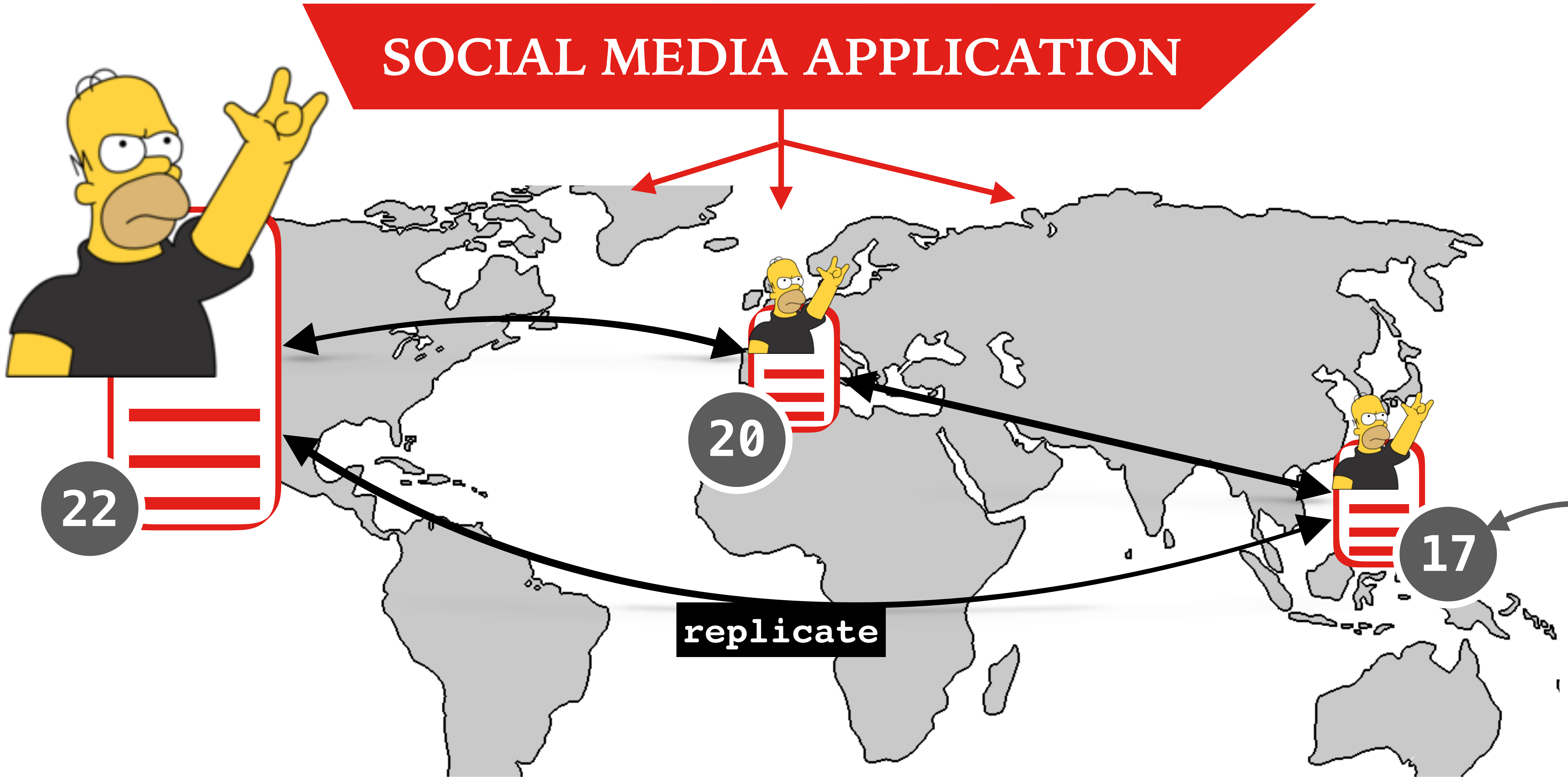
- Returns the **correct** data 22
- Latency: ~**200 ms**
- Can become **unavailable**

[CAP], [PACELC]

Weak Consistency

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

22 *or* 20 *or* 17

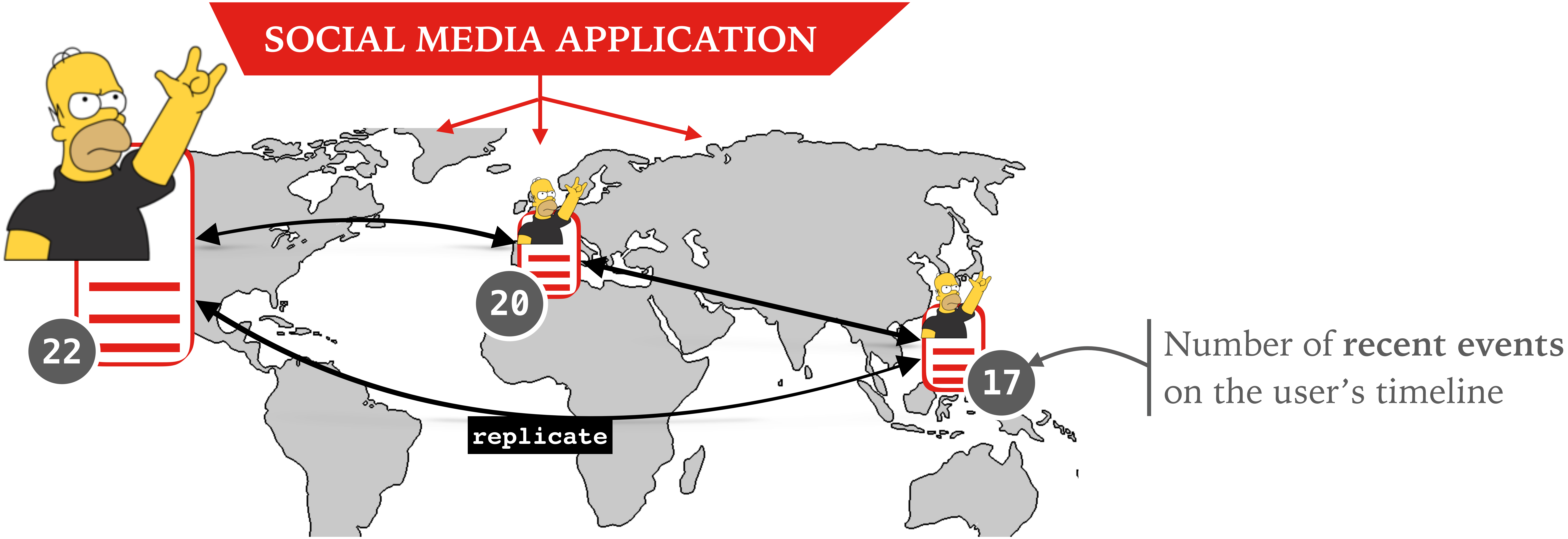


Number of **recent events** on the user's timeline

Strong Consistency

Weak Consistency

Neither model is ideal!



Strong Consistency

Weak Consistency

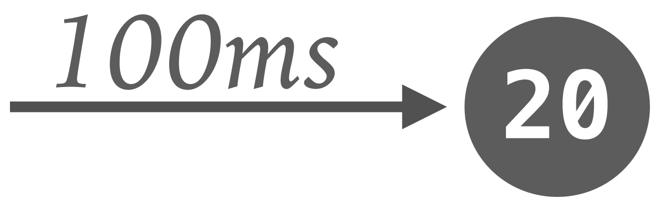
Neither model is ideal!

↓

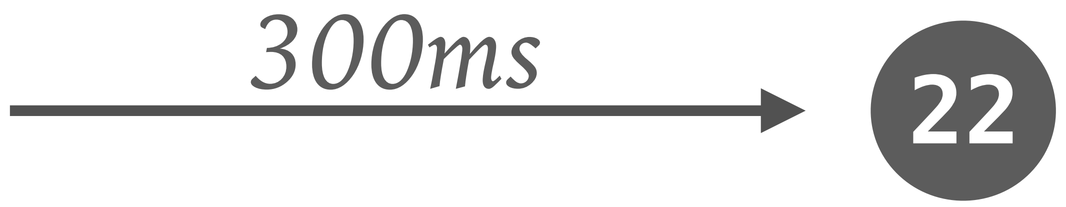
We use both models.

Multiple models

1. **Weak consistency**

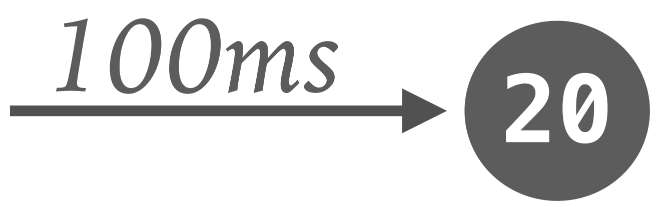


2. **Strong consistency**



Multiple models

1. **Weak consistency**



2. **Strong consistency**

Increasingly many systems expose multiple consistency models:

App Engine

cassandra

Dynamo [SOSP'07]

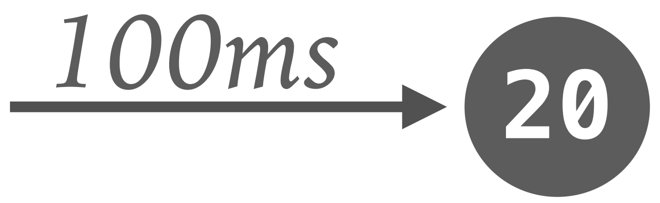
SimpleDB

riakKV

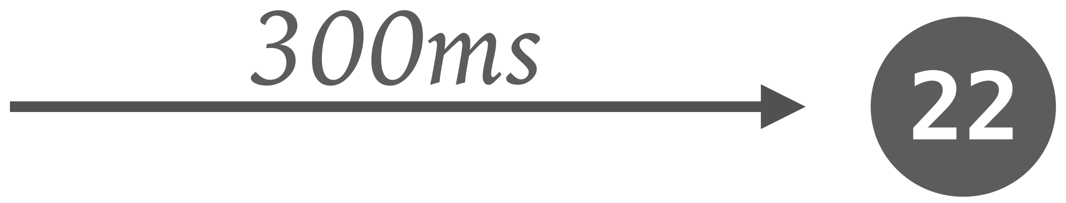
Pileus [SOSP'13]

Multiple models

1. **Weak consistency**



2. **Strong consistency**



Increasingly many systems expose multiple consistency models:

App Engine

cassandra

Dynamo [SOSP'07]

Pileus [SOSP'13]

SimpleDB

riakKV

Issues

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

Multiple models

1. **Weak consistency**

   100ms → 20

2. **Strong consistency**

   300ms → 22

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

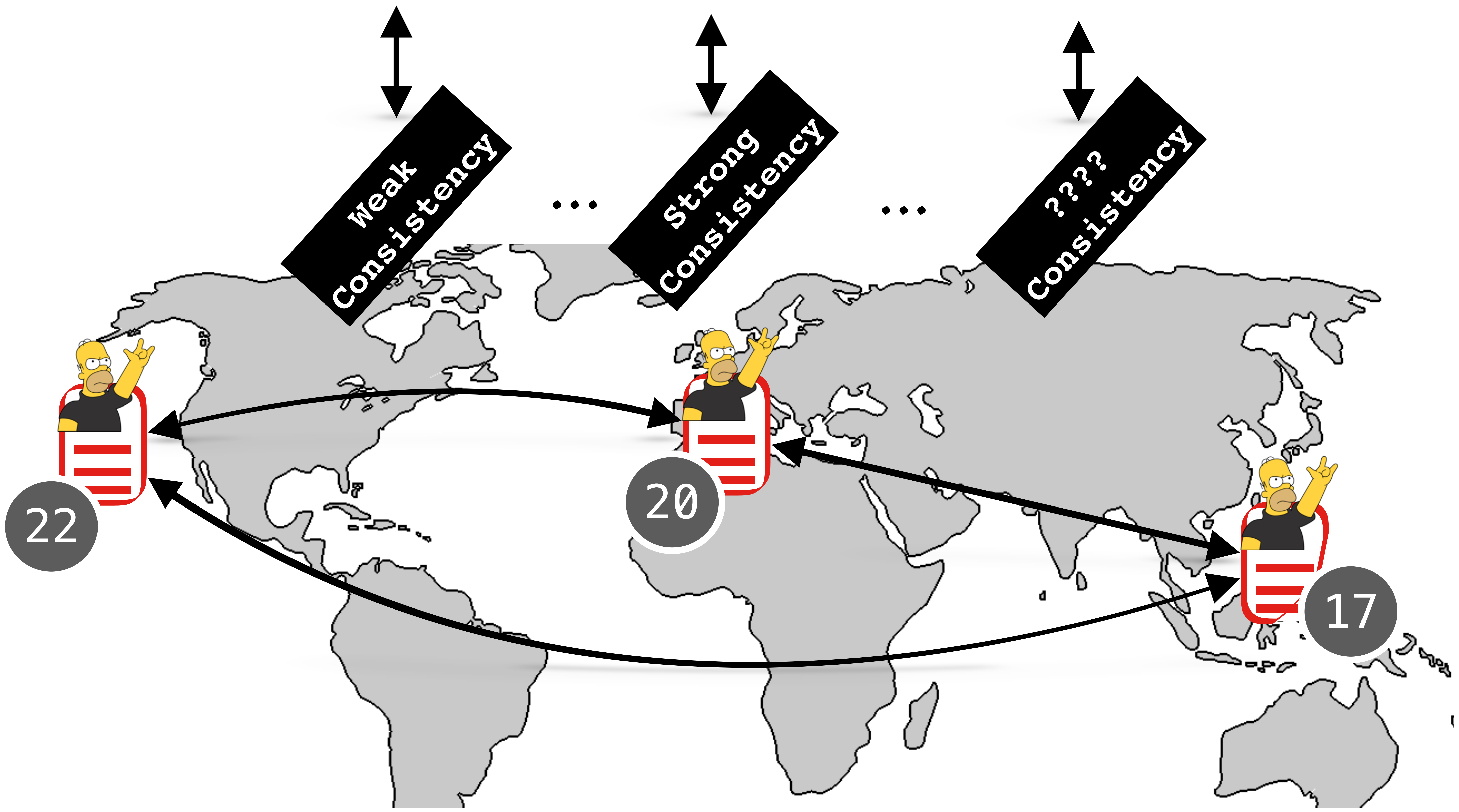
How do you program with

★ inconsistencies?

★ multiple values?

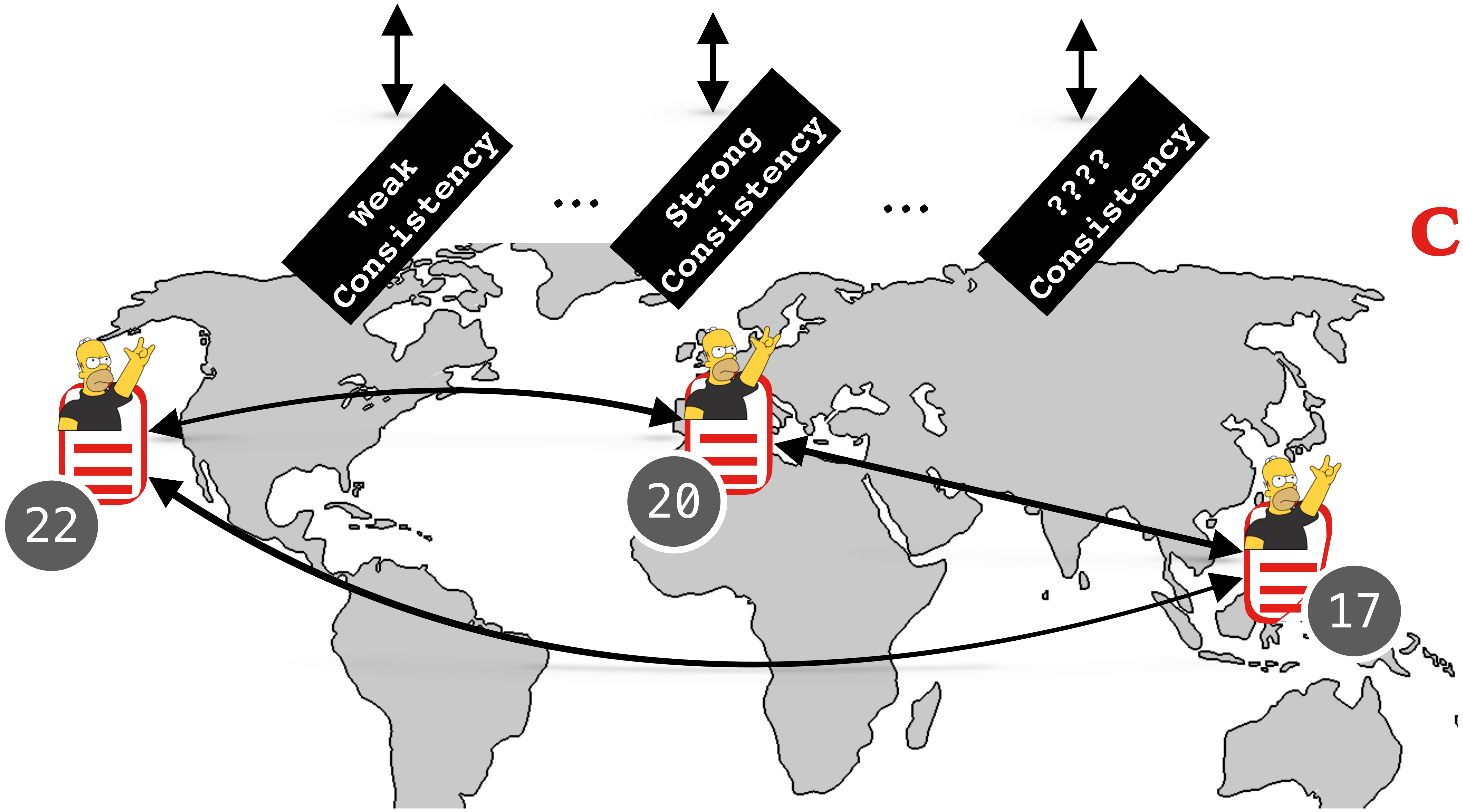
SOCIAL MEDIA APPLICATION

ABSTRACTION
FOR REPLICATED OBJECTS

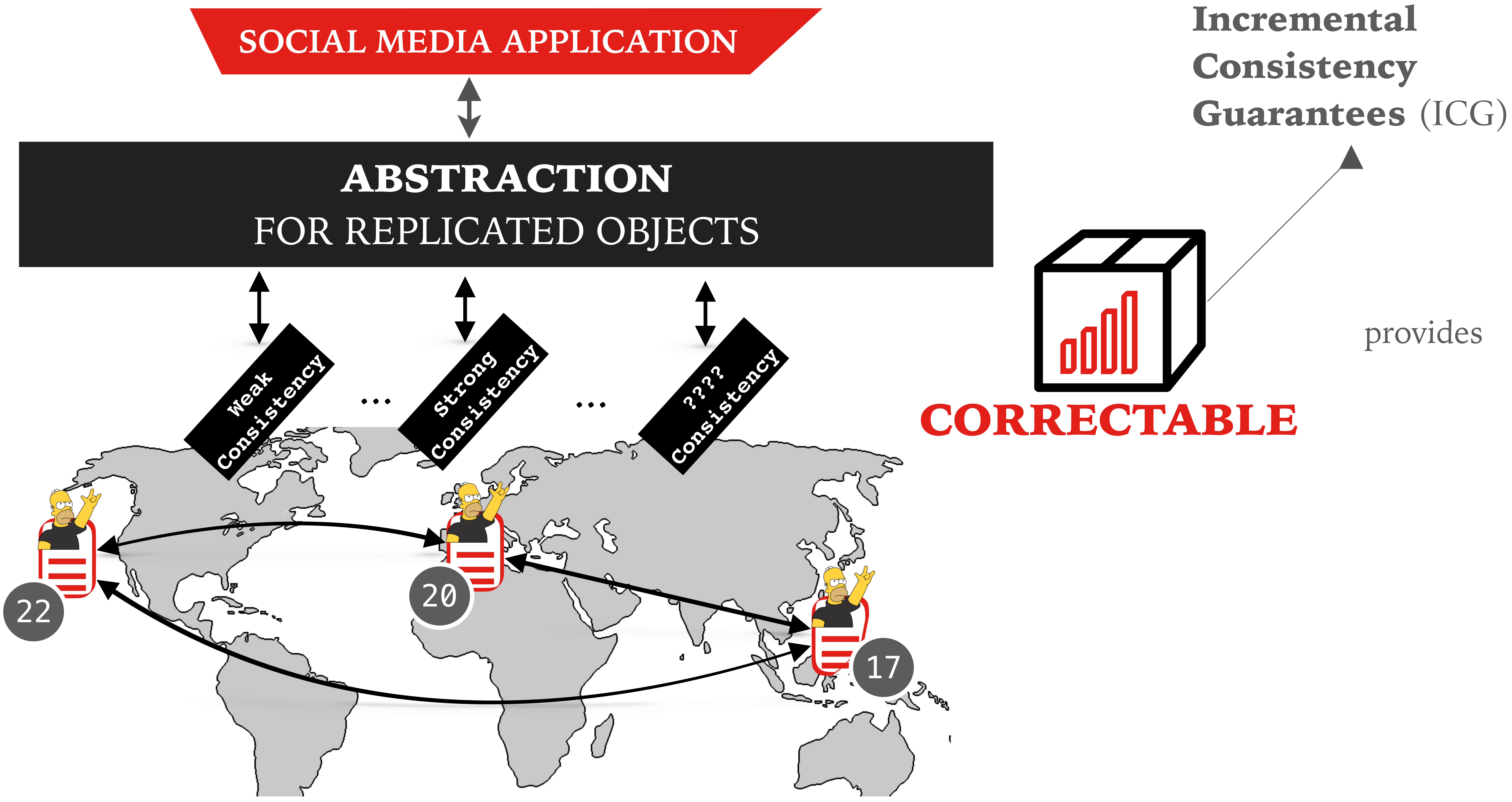


SOCIAL MEDIA APPLICATION

ABSTRACTION
FOR REPLICATED OBJECTS



CORRECTABLE



SOCIAL MEDIA APPLICATION

ABSTRACTION
FOR REPLICATED OBJECTS

Incremental Consistency Guarantees (ICG)

provides

CORRECTABLE

Correctables / **Design**

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

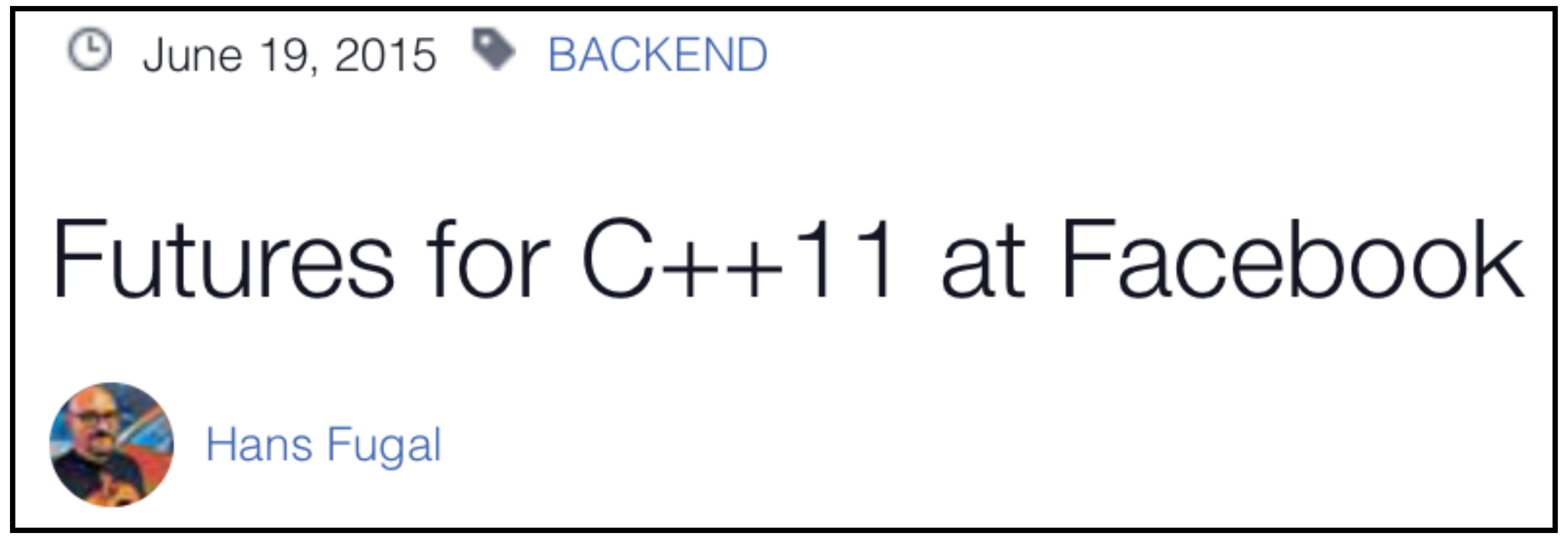
Futures and Promises

Scala

finagle

JS

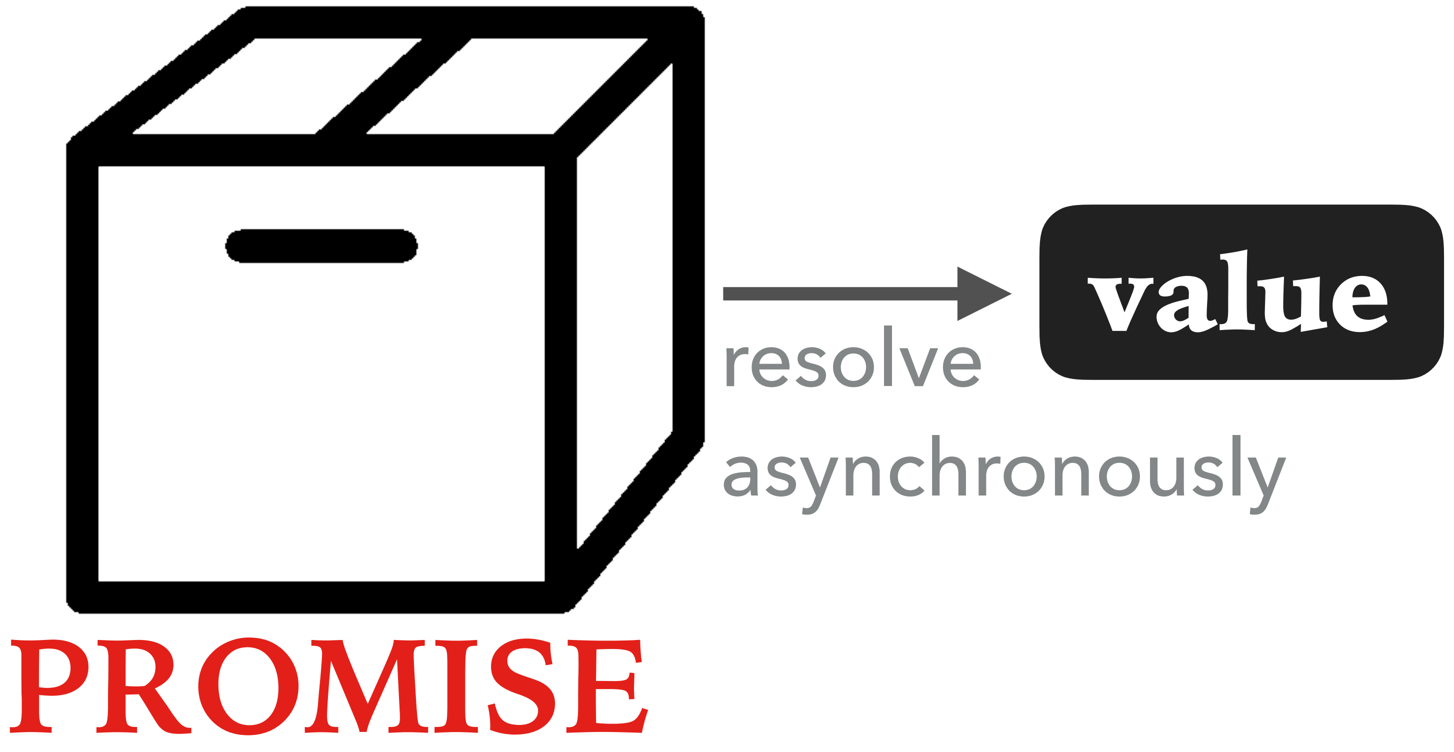
then Promises/A+


June 19, 2015 BACKEND

Futures for C++11 at Facebook

Hans Fugal

Google Guava
Core libraries for Java & Android


PROMISE

resolve asynchronously

value

Correctables / **Design**

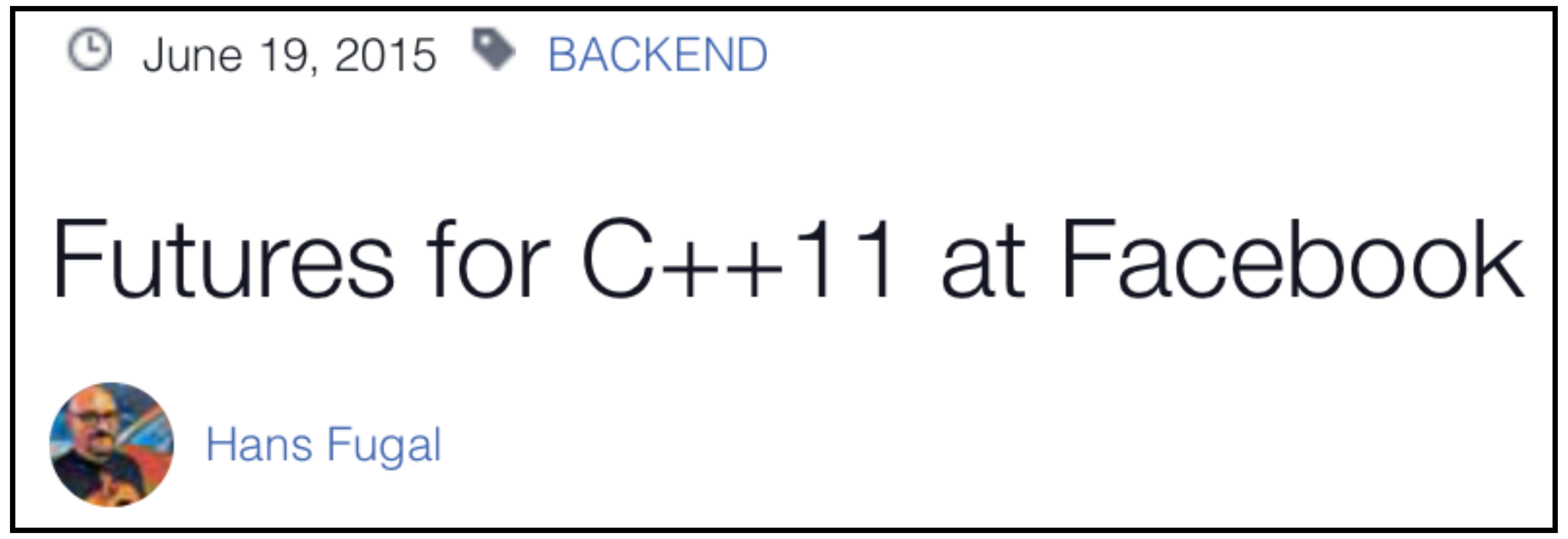
- Starting point: **Promises**
  - Placeholders for values
  - Becoming mainstream

Futures and Promises

Scala

finagle

then Promises/A+ JS

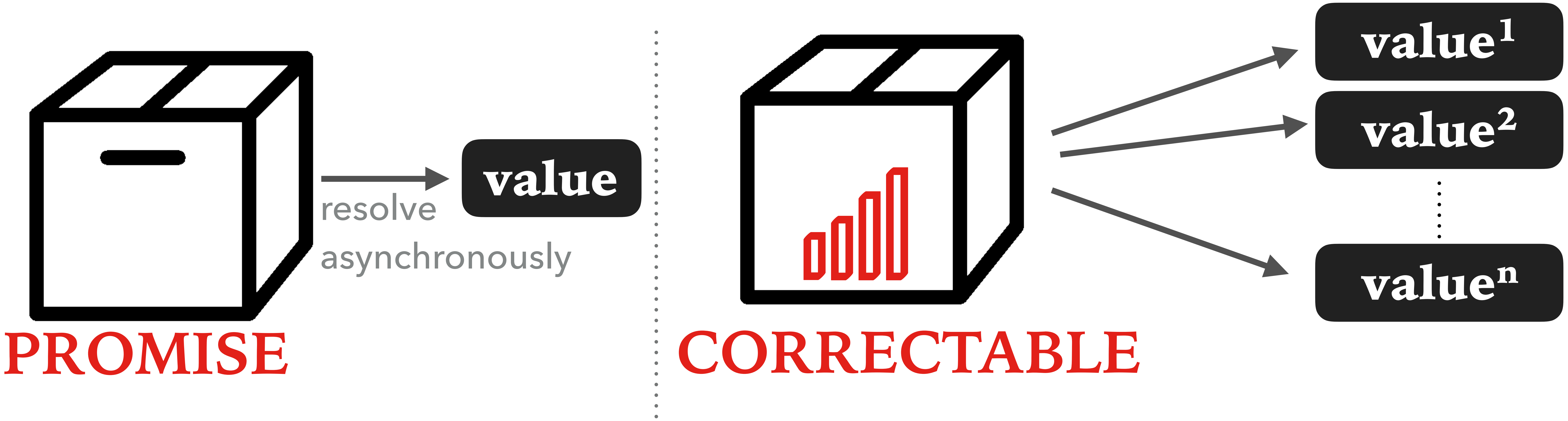


June 19, 2015 BACKEND

Futures for C++11 at Facebook

Hans Fugal

Google Guava
Core libraries for Java & Android



Correctables / **Design**

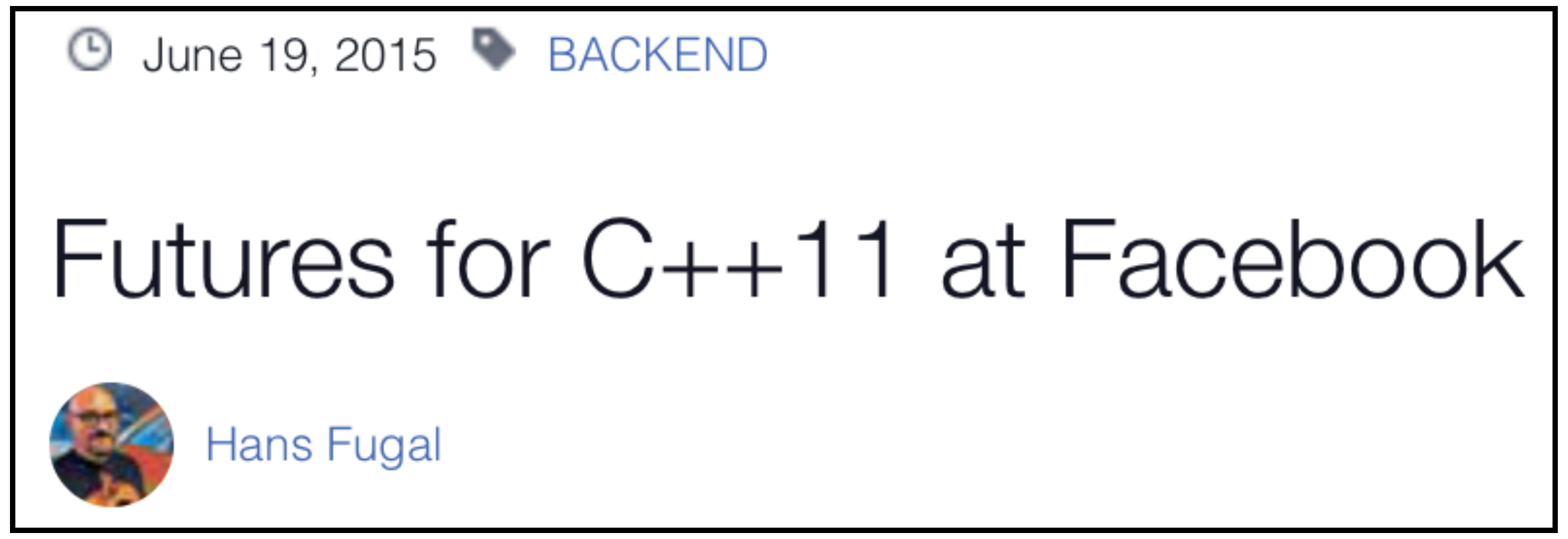
- Starting point: **Promises**
  - Placeholders for values
  - Becoming mainstream

Futures and Promises

Scala

finagle

then Promises/A+ JS



June 19, 2015 BACKEND

Futures for C++11 at Facebook

Hans Fugal

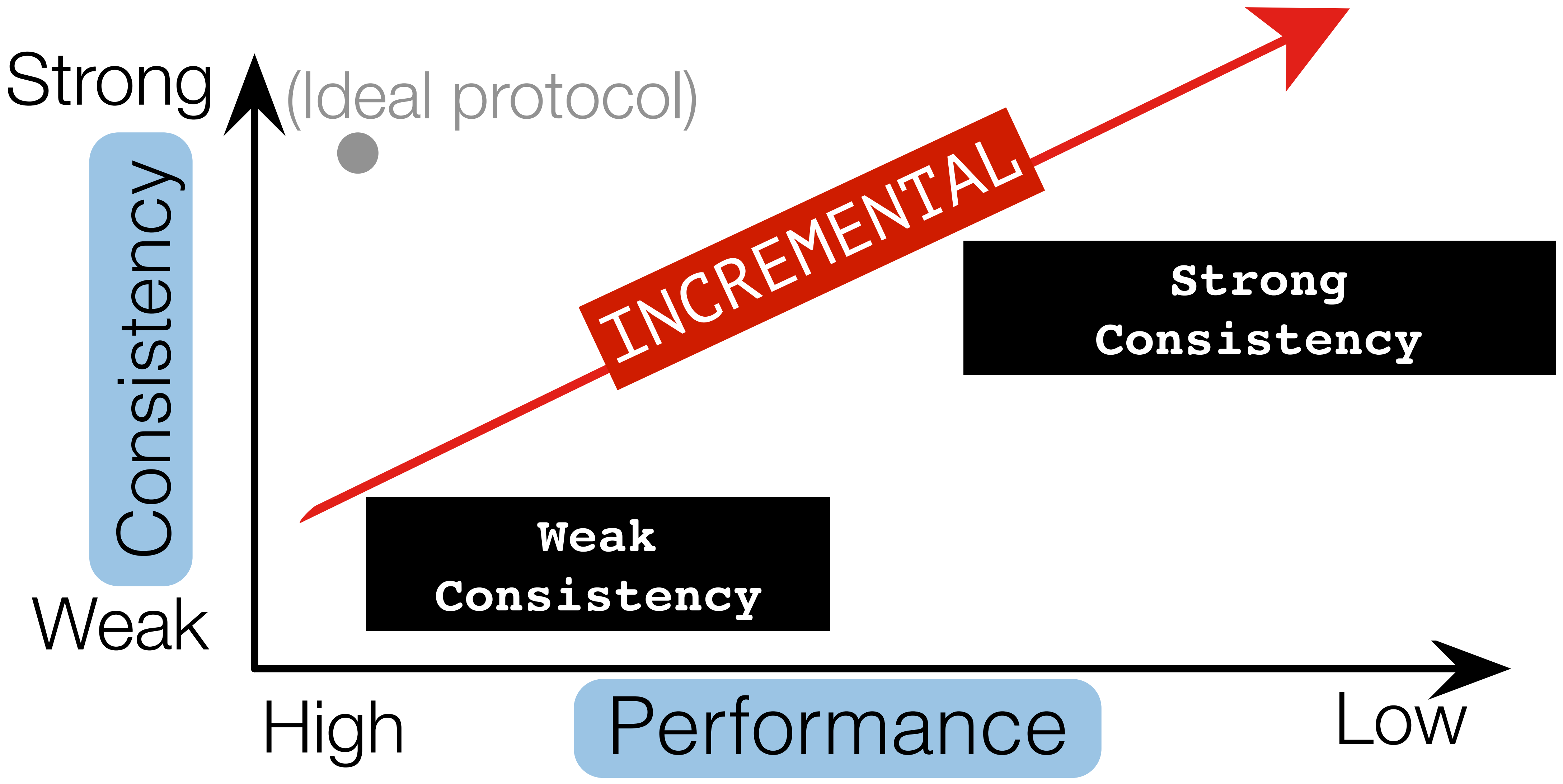
Google Guava
Core libraries for Java & Android

resolve
asynchronously
value
PROMISE
CORRECTABLE
value^1
value^2
value^n

PROGRESSIVELY STRONGER CONSISTENCY (INCREMENTAL)

PROGRESSIVELY HIGHER LATENCY

Consistency Models are **Complementary**



Weak consistency:

★ Fast

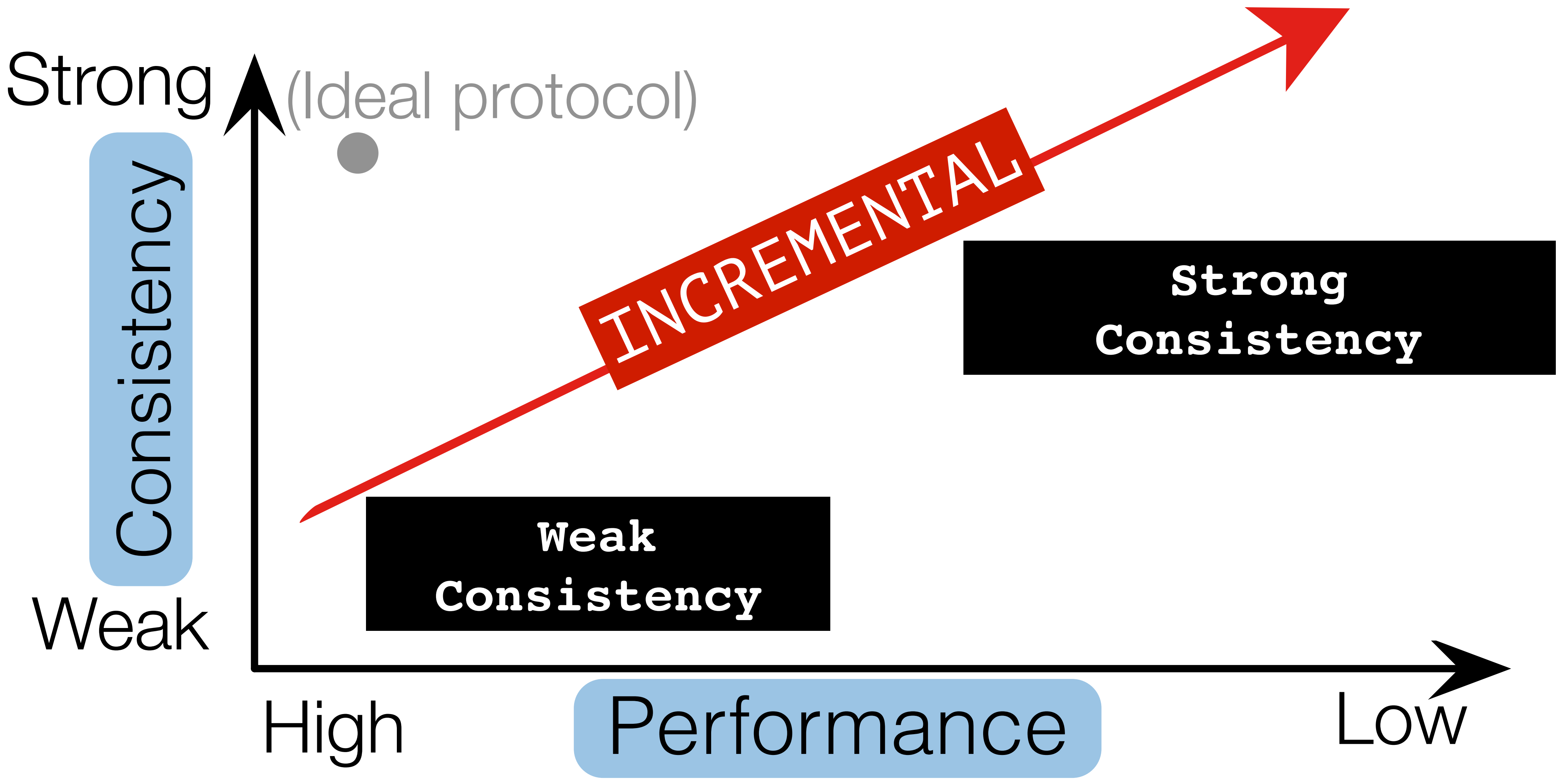
★ (Often correct)

Strong consistency:

★ Slower

★ (Correct with certainty)

Consistency Models are **Complementary**



Weak consistency:

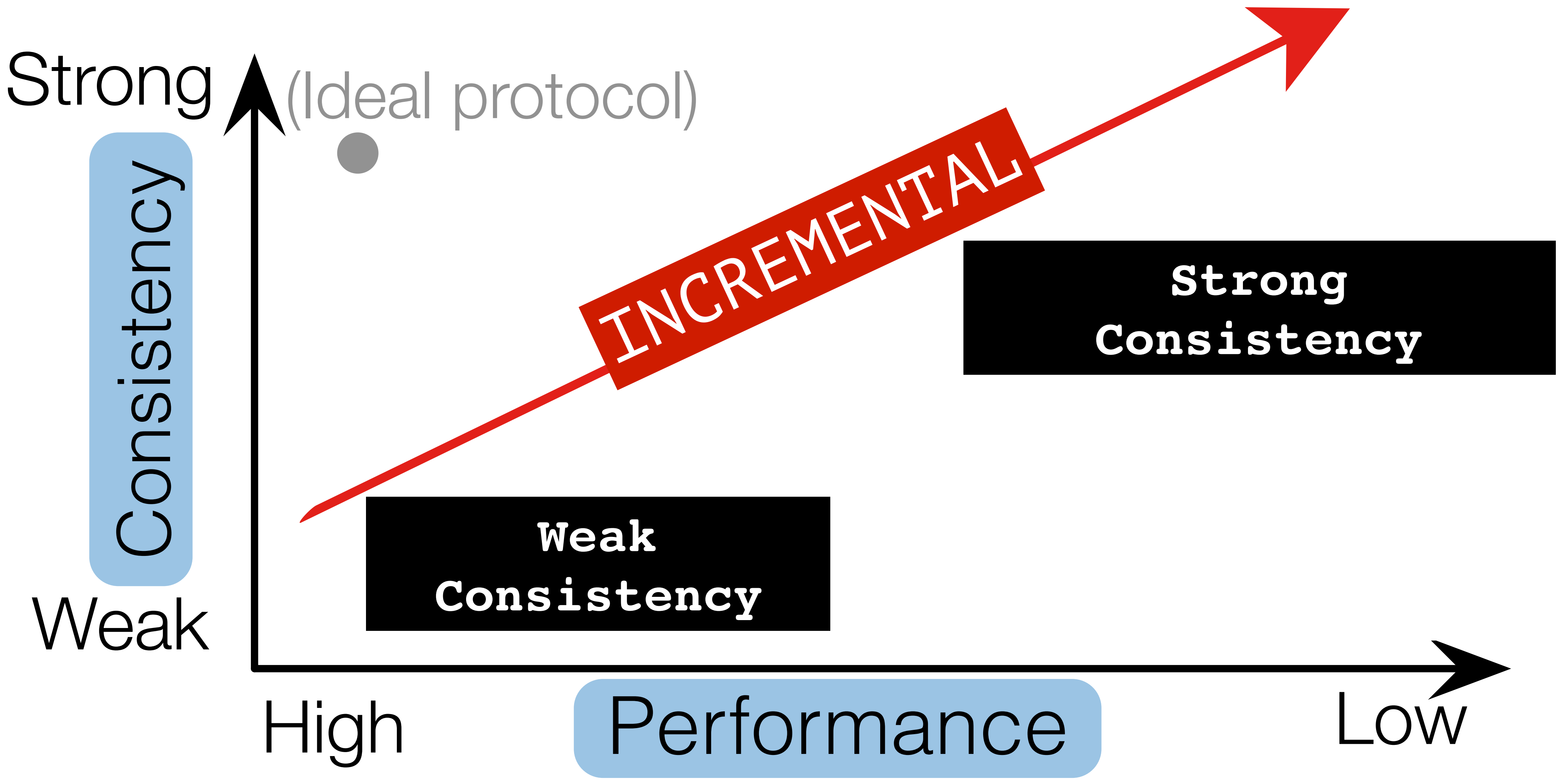
- ★ Fast
- ★ (Often correct)

Strong consistency:

- ★ Slower
- ★ (Correct with certainty)

So what?

Consistency Models are **Complementary**



Weak consistency:

- ★ Fast
- ★ (Often correct)

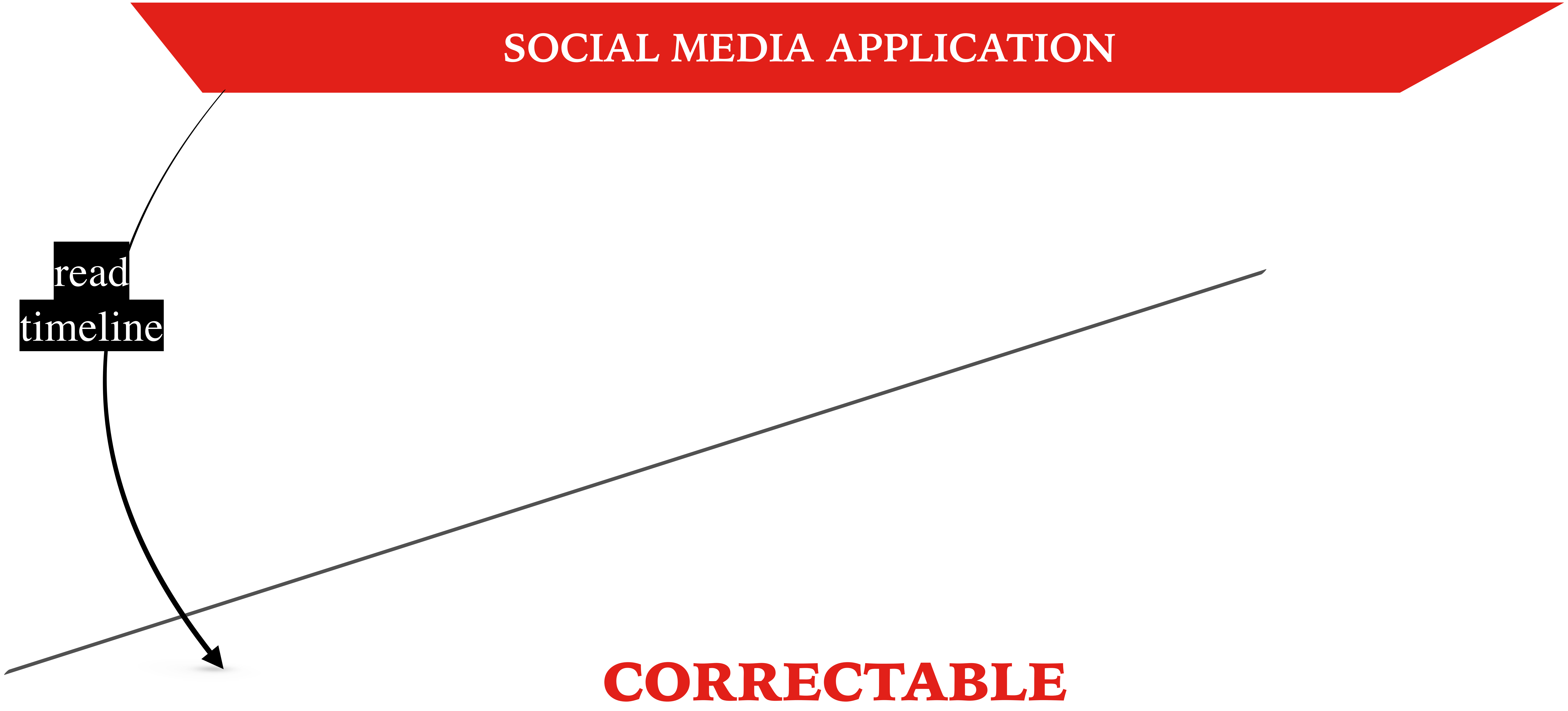
Strong consistency:

- ★ Slower
- ★ (Correct with certainty)

So what?

Latency optimizations

Speculating with Correctables

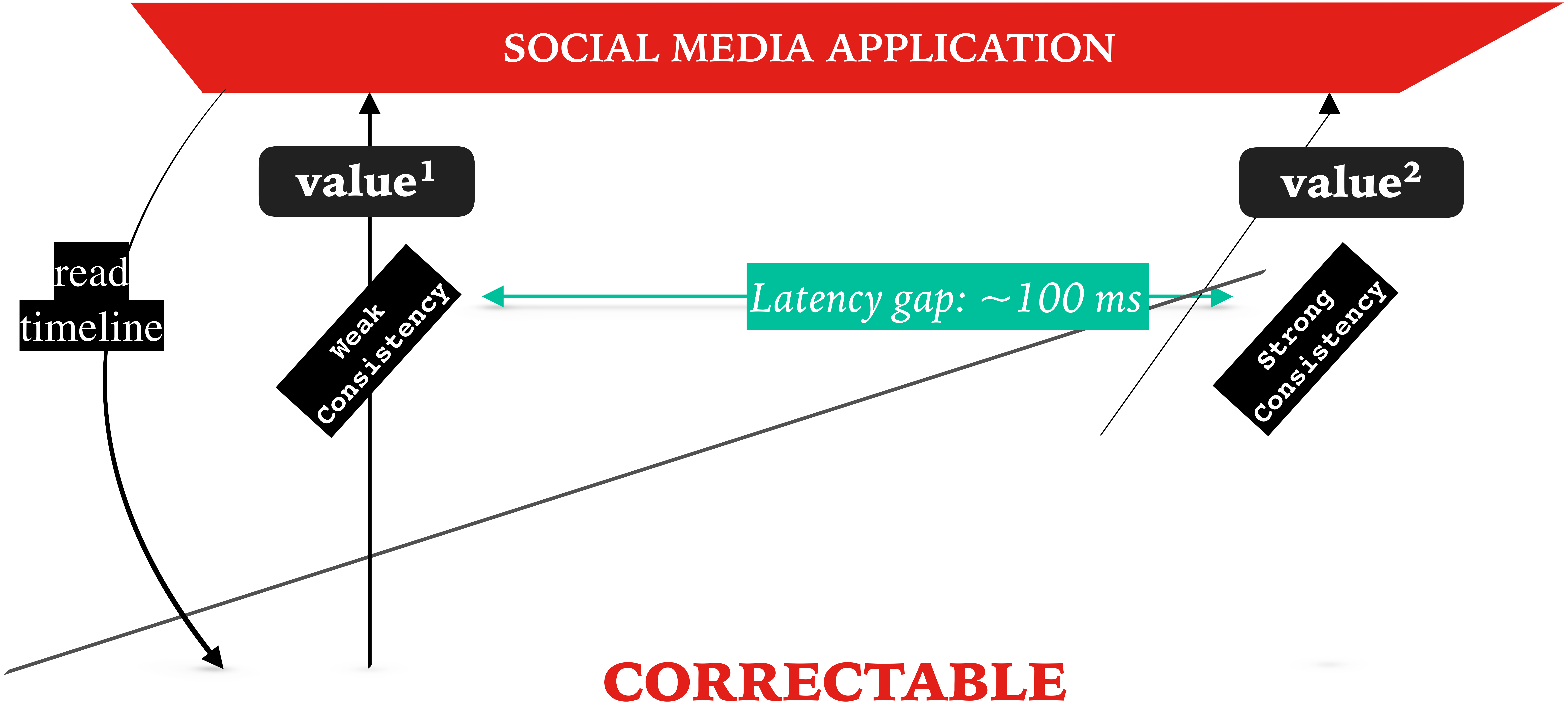


SOCIAL MEDIA APPLICATION

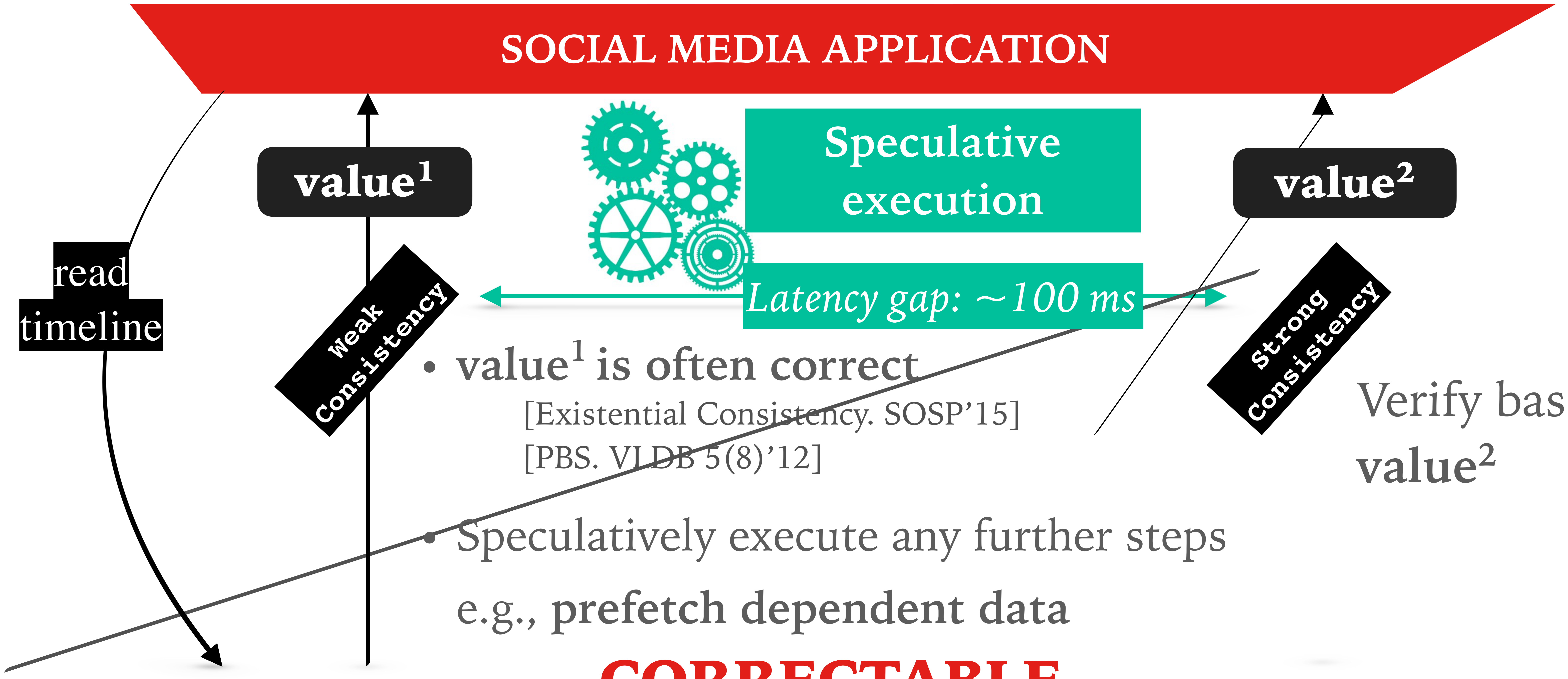
read timeline

CORRECTABLE

Speculating with Correctables



Speculating with Correctables

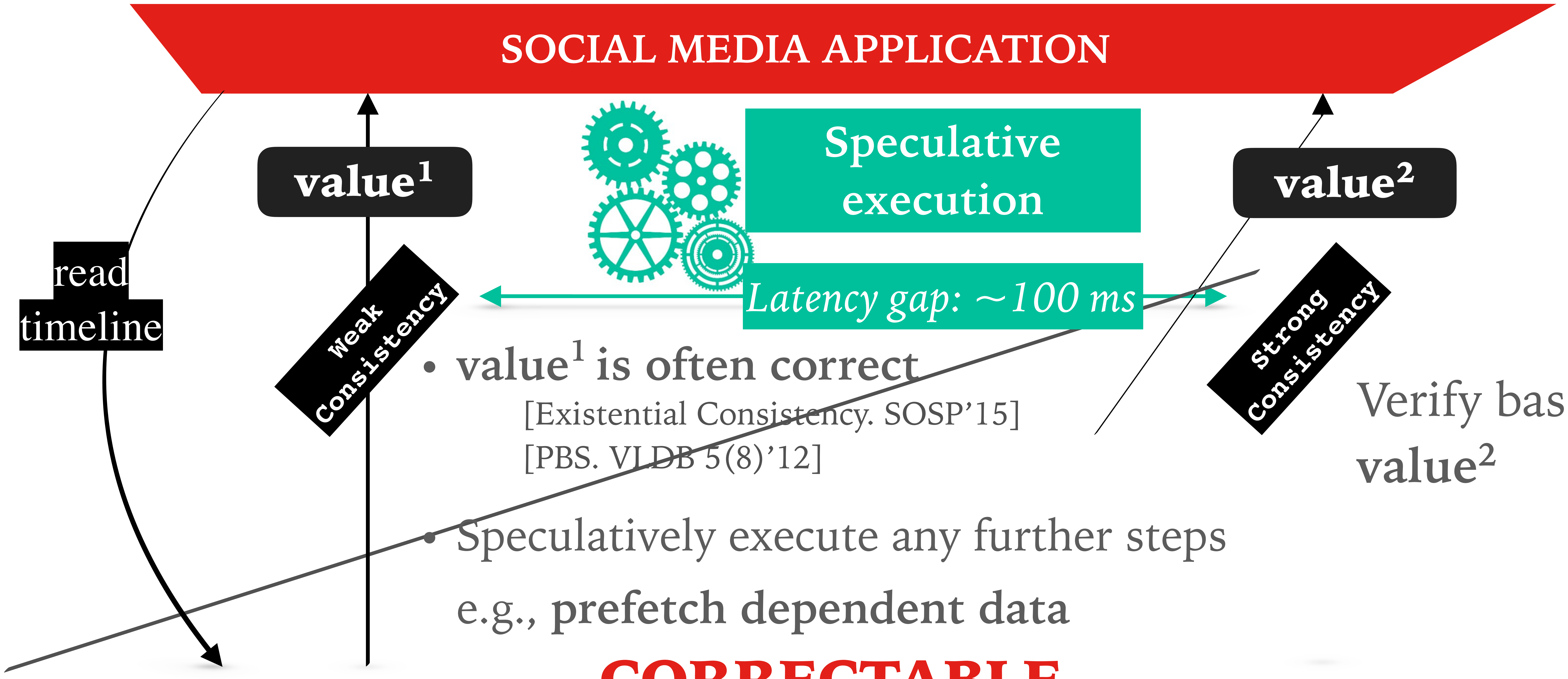


- **value^1 is often correct**
  [Existential Consistency. SOSP'15]
  [PBS. VLDB 5(8)'12]
- Speculatively execute any further steps e.g., **prefetch dependent data**

Verify based on **value^2**

CORRECTABLE

Speculating with Correctables



SOCIAL MEDIA APPLICATION

read timeline

value^1

Weak Consistency

Speculative execution

Latency gap: ~100 ms

value^2

Strong Consistency

- **value^1 is often correct**
  [Existential Consistency. SOSP'15]
  [PBS. VLDB 5(8)'12]
- Speculatively execute any further steps
  e.g., **prefetch dependent data**

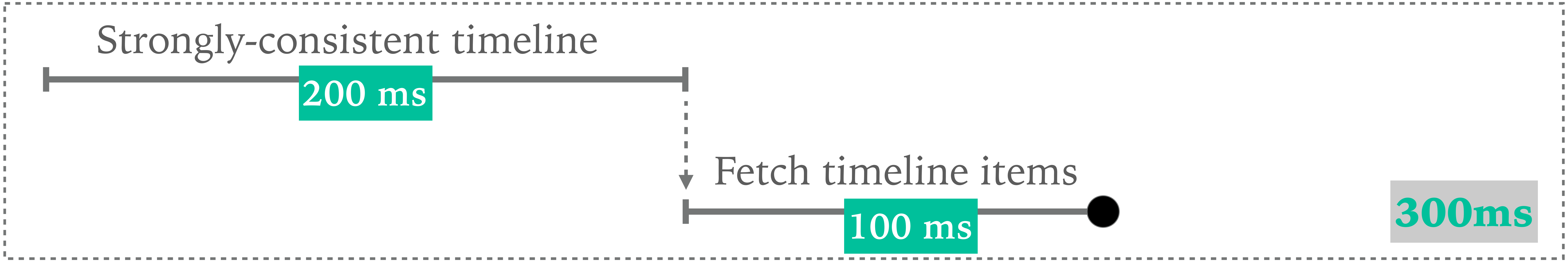
Verify based on **value^2**

CORRECTABLE

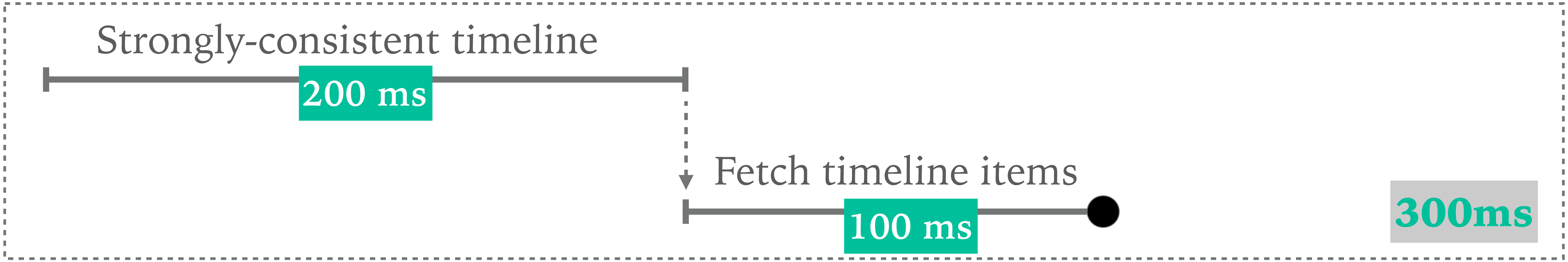
Lower latency of strong consistency

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

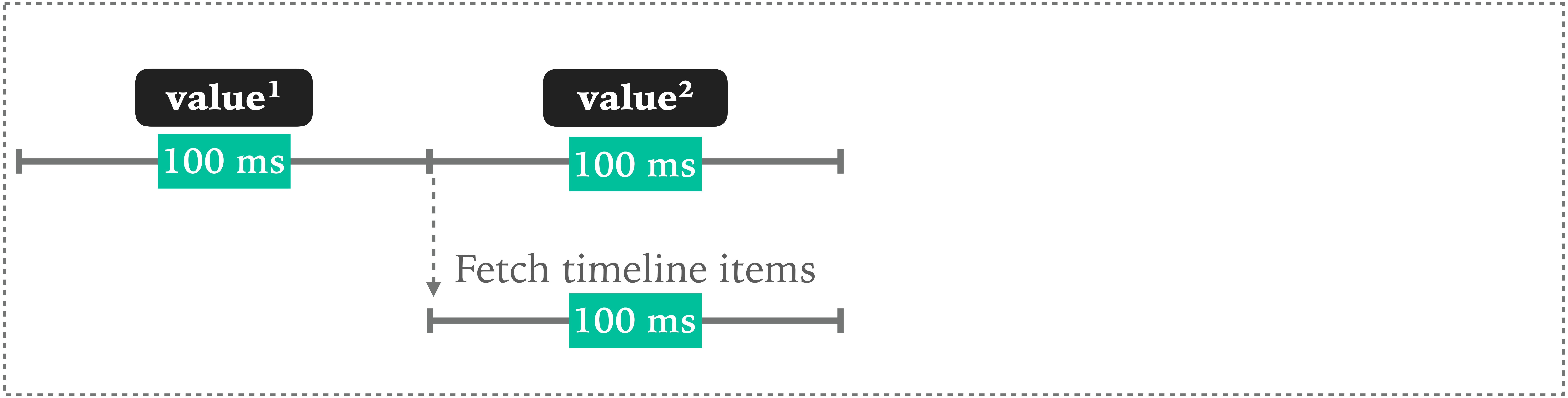
Traditional operation:



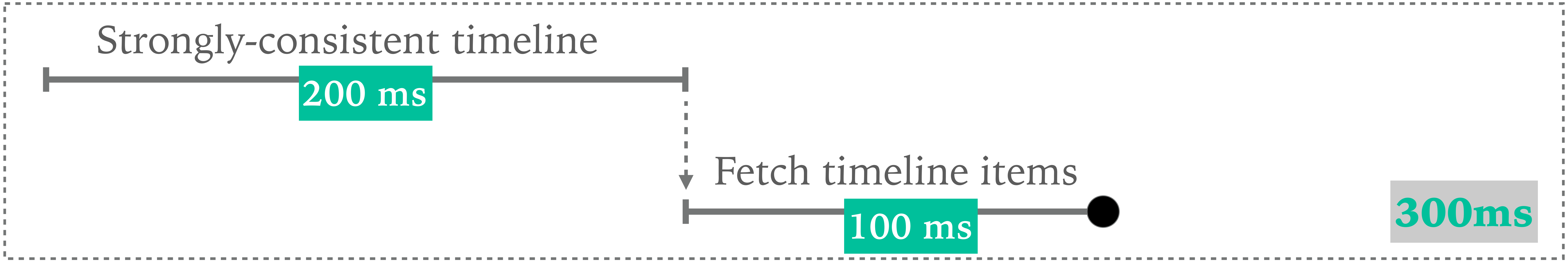
Traditional operation:



Speculative operation with ICG:



Traditional operation:



Strongly-consistent timeline

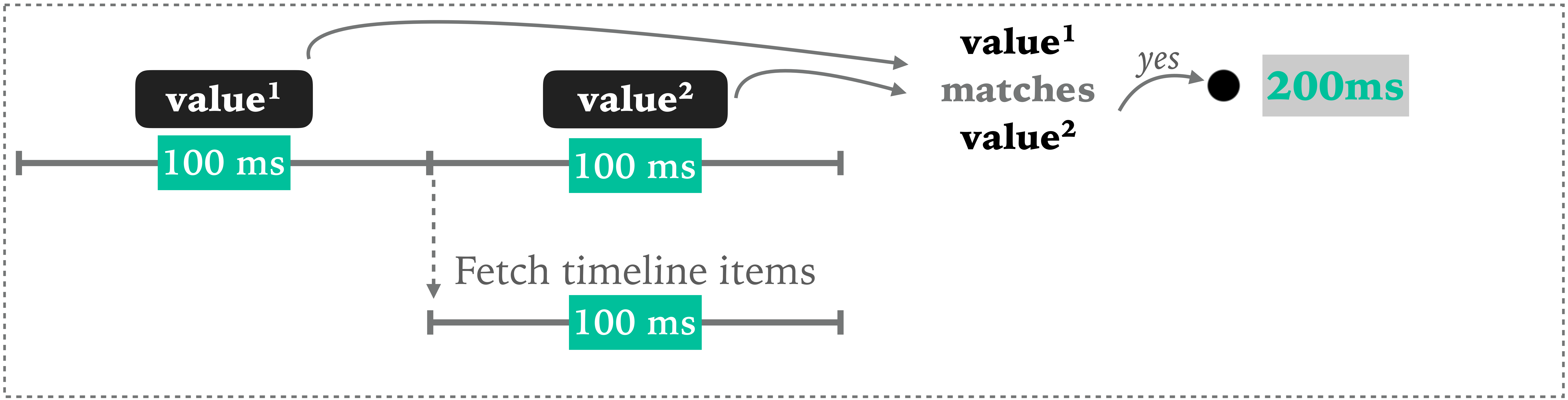
200 ms

Fetch timeline items

100 ms

300ms

Speculative operation with ICG:



value^{1}

100 ms

value^{2}

100 ms

Fetch timeline items

100 ms

value^{1}

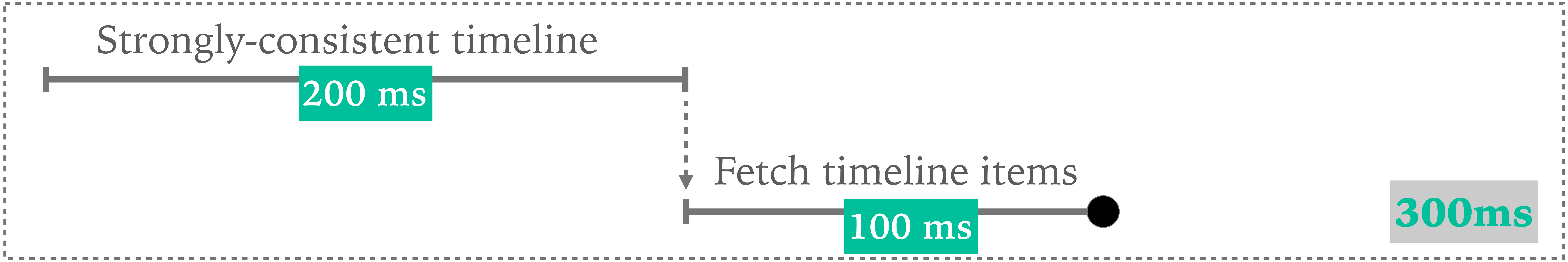
matches

value^{2}

yes

200ms

Traditional operation:



Strongly-consistent timeline

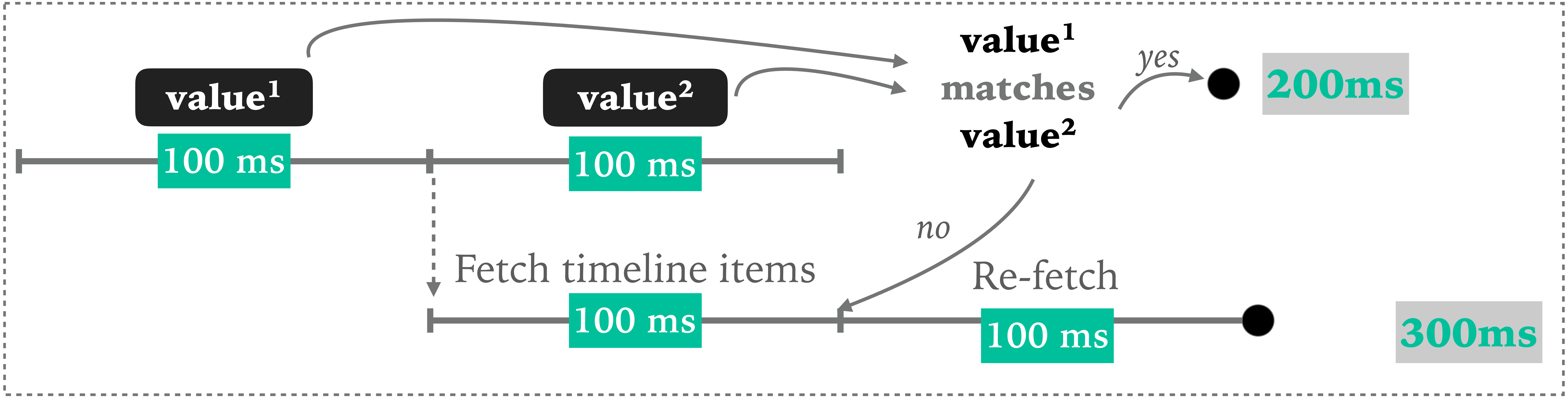
200 ms

Fetch timeline items

100 ms

300ms

Speculative operation with ICG:



value^{1}

100 ms

value^{2}

100 ms

value^{1} matches value^{2}

yes

200ms

no

Fetch timeline items

100 ms

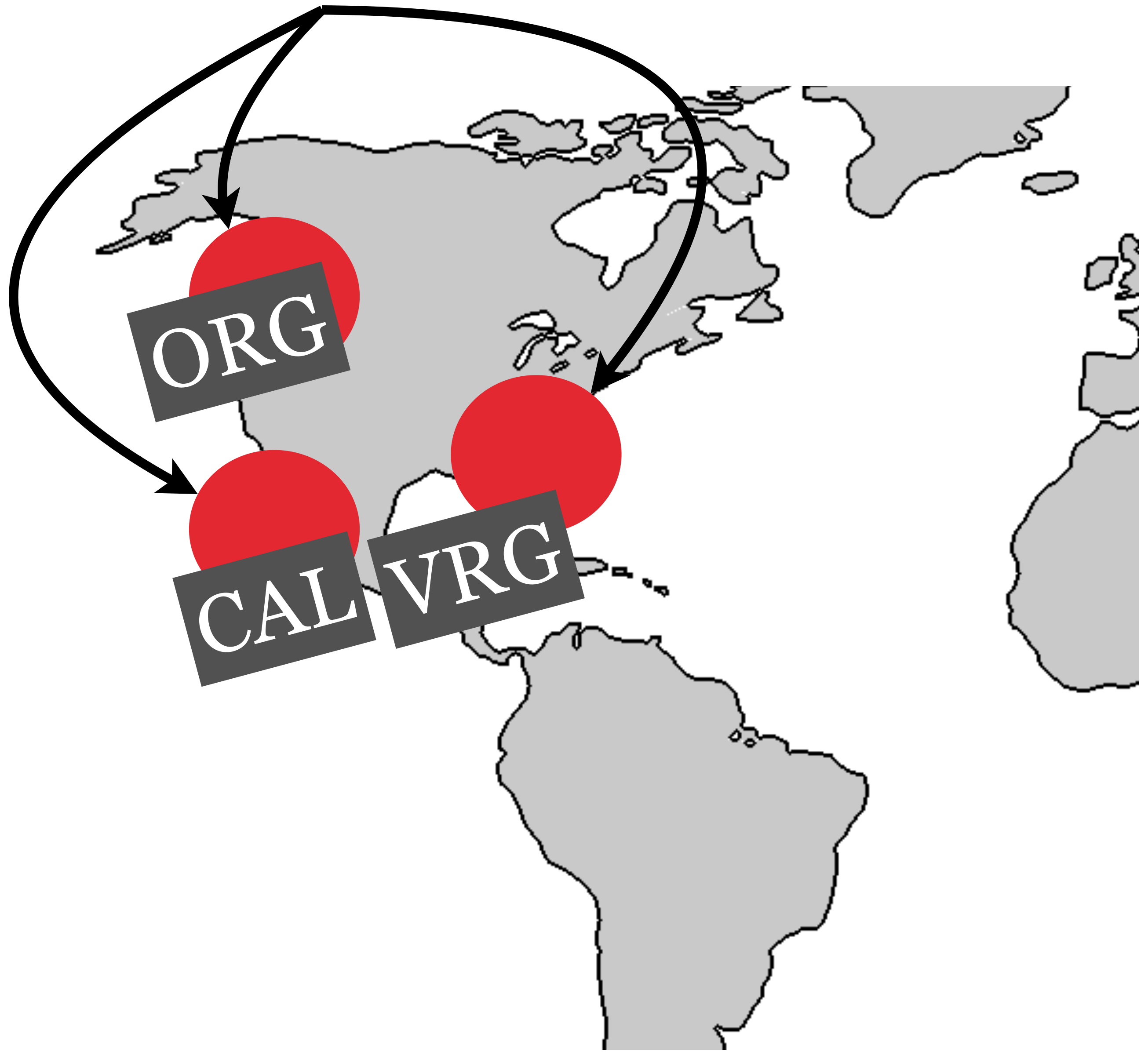
Re-fetch

100 ms

300ms

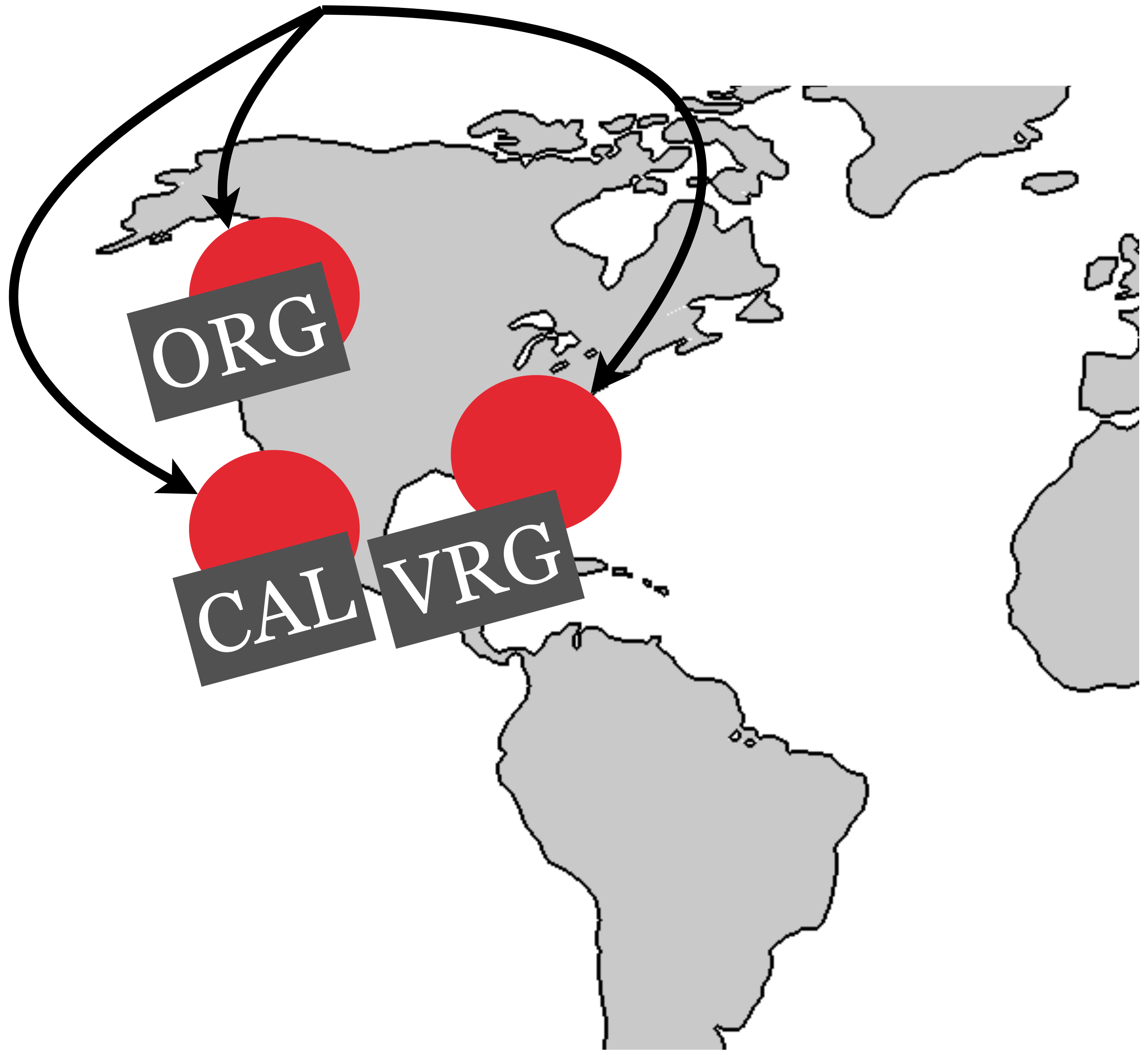
Speculation case-study

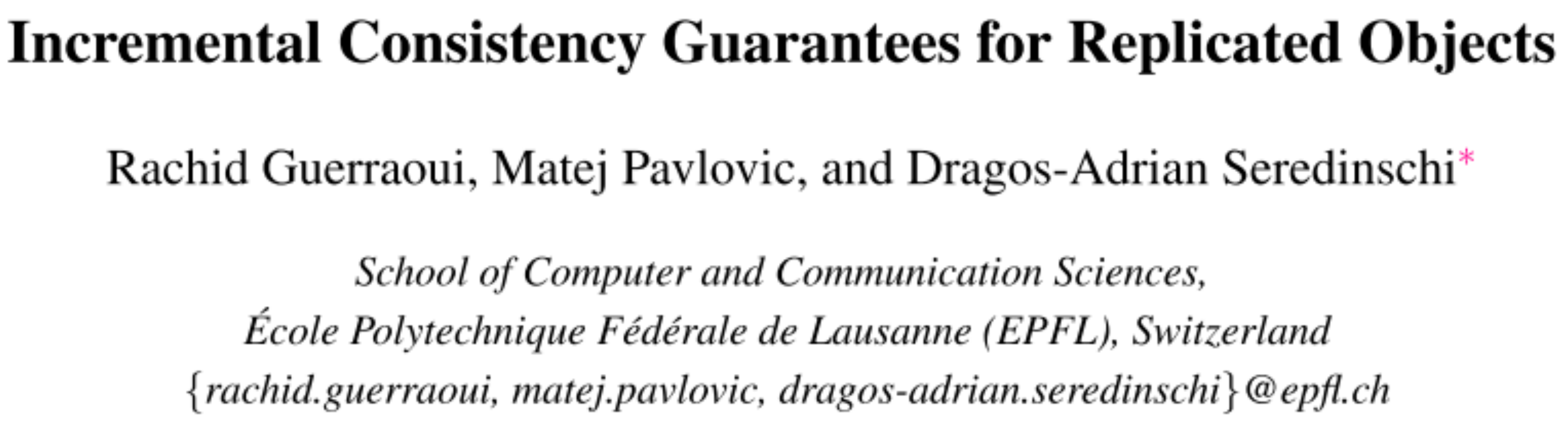
- Application: **Twissandra**
- Workload generated via YCSB
- Clients in Ireland
- Geo-replication on Amazon's EC2



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.

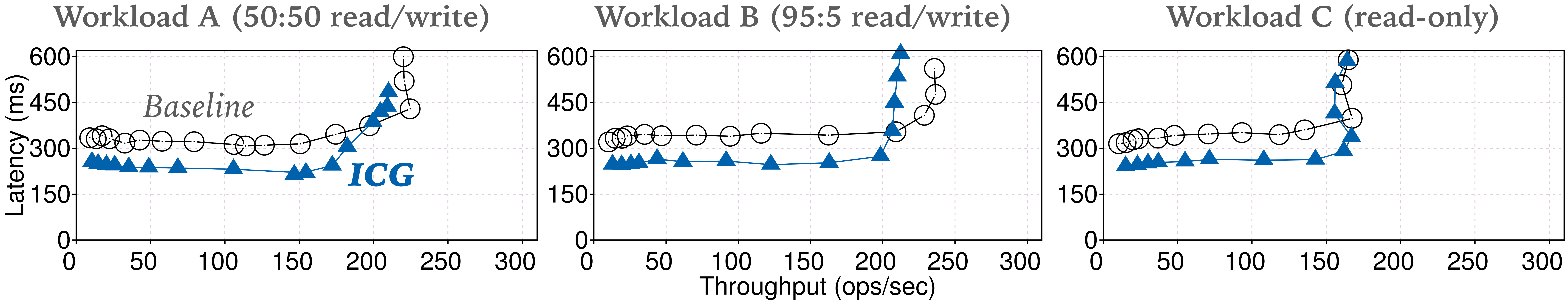
ICG

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

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Decreasing latency of strong consistency

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



Baseline

Read using a quorum of 2/3 replicas

vs.

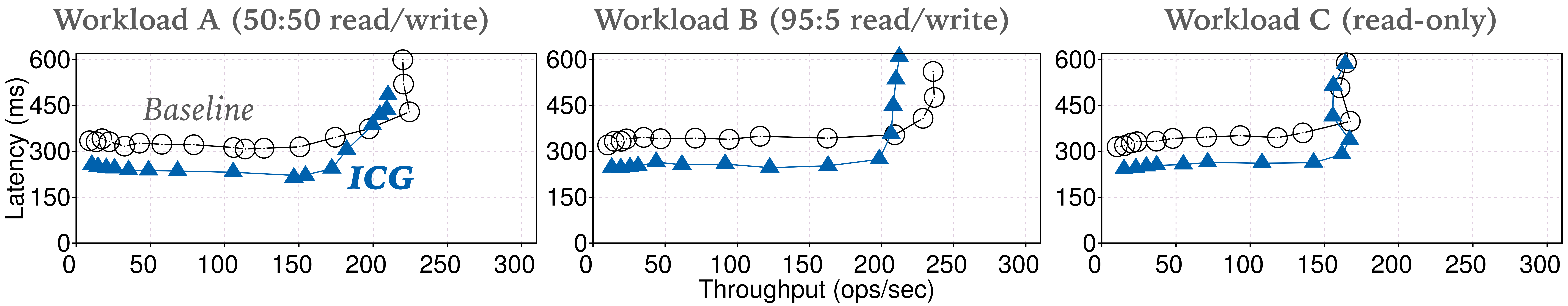
ICG

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)*



Baseline

Read using a quorum of 2/3 replicas

vs.

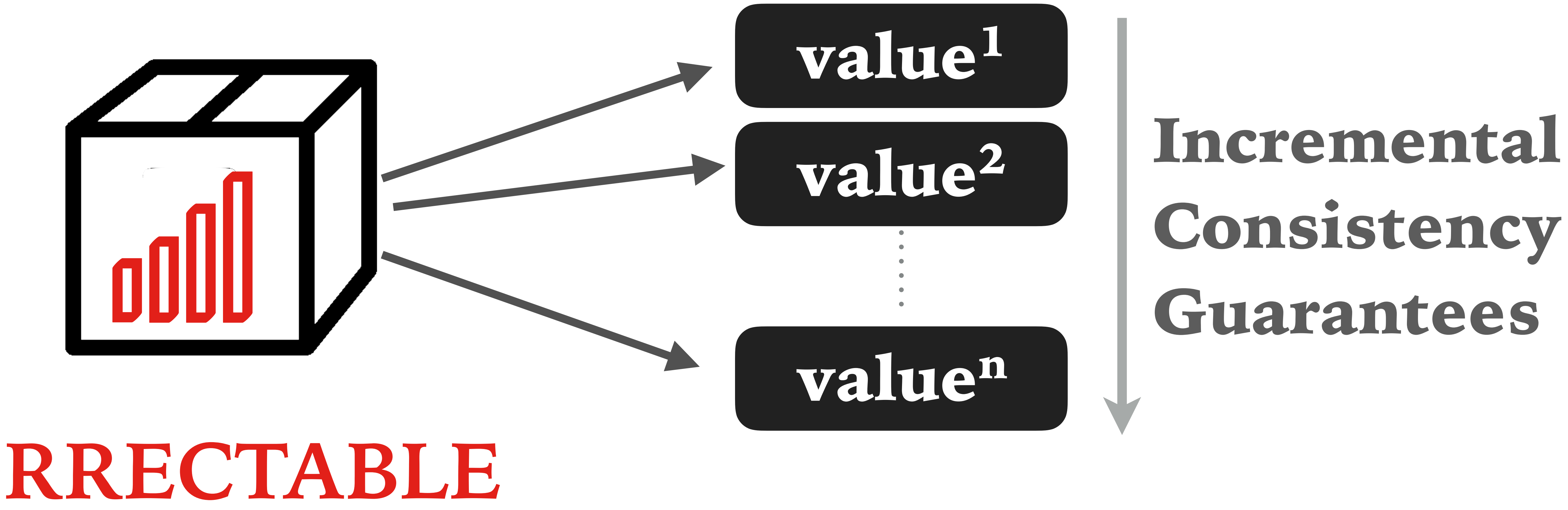
ICG

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**



CORRECTABLE

value^1

value^2

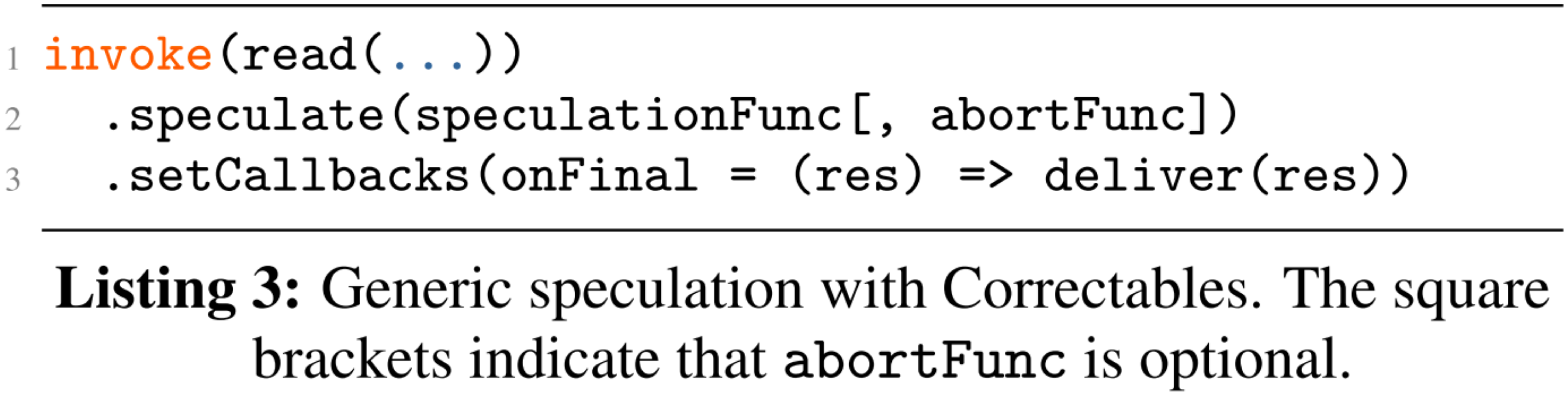
value^n

Incremental Consistency Guarantees

@adizere

backup slides

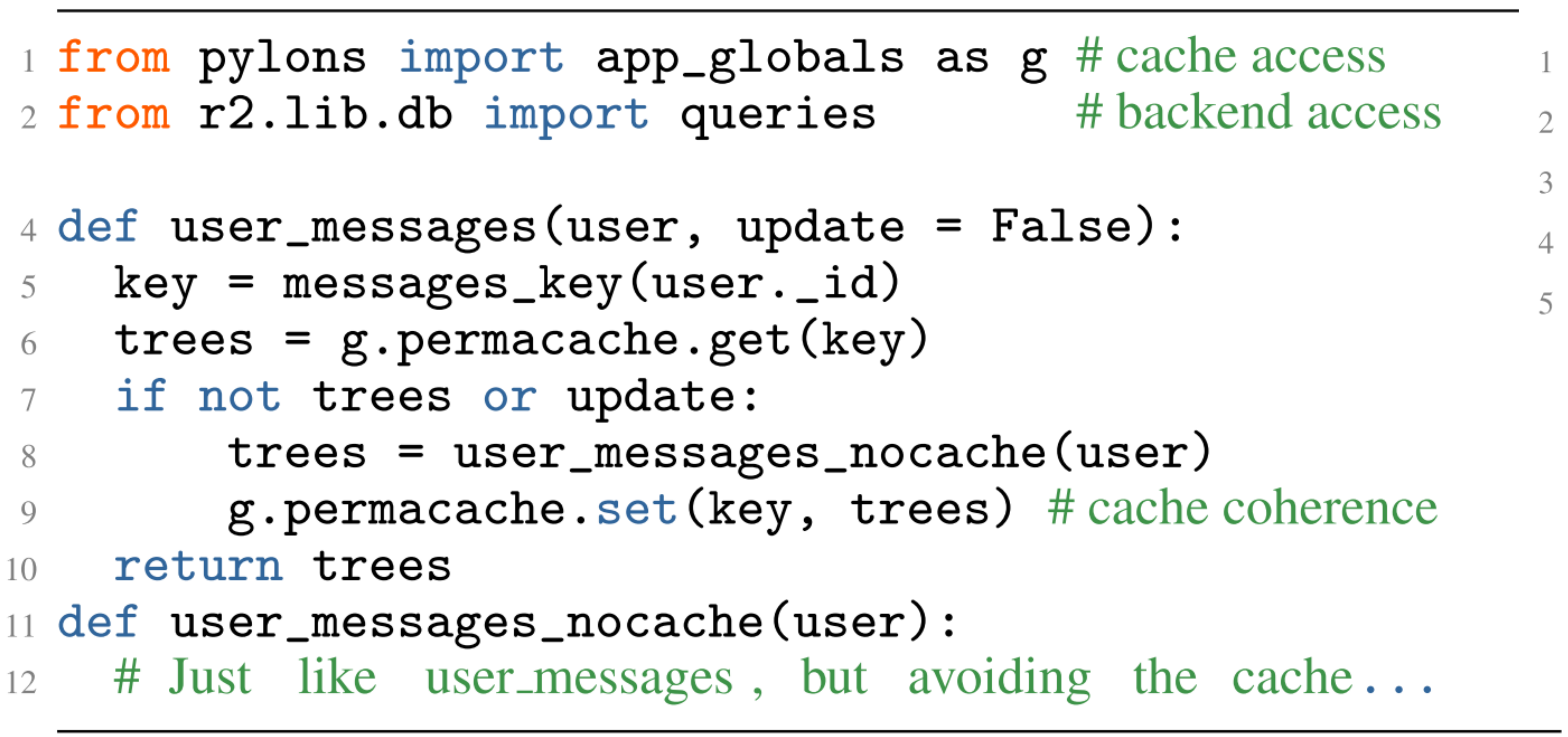
Speculation // Syntactic sugar



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

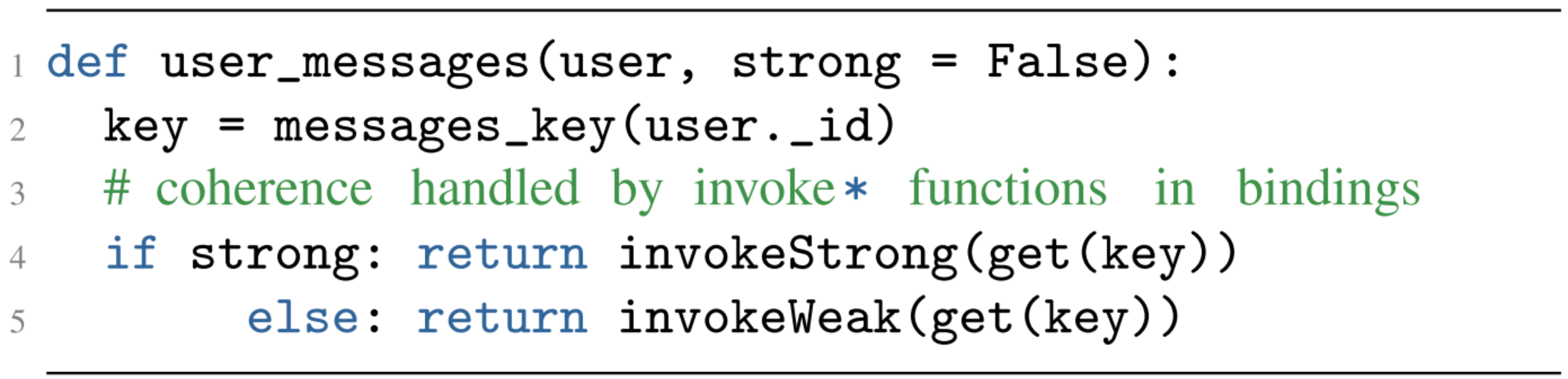
Listing 3: Generic speculation with Correctables. The square brackets indicate that `abortFunc` is optional.

Legacy code vs. Correctables



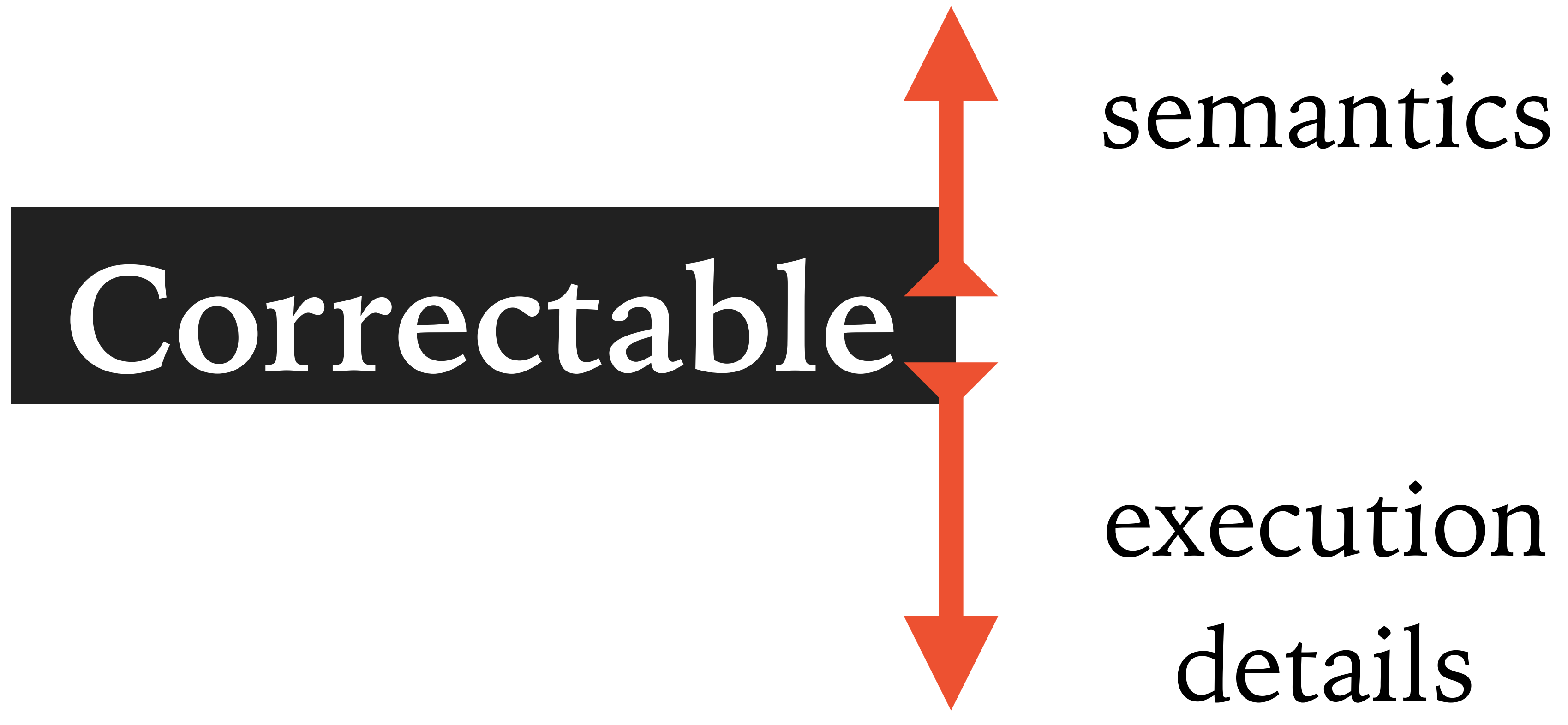
```
from pylons import app_globals as g # cache access
from r2.lib.db import queries       # backend access

def user_messages(user, update = False):
  key = messages_key(user._id)
  trees = g.permacache.get(key)
  if not trees or update:
      trees = user_messages_nocache(user)
      g.permacache.set(key, trees) # cache coherence
  return trees
def user_messages_nocache(user):
  # 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.

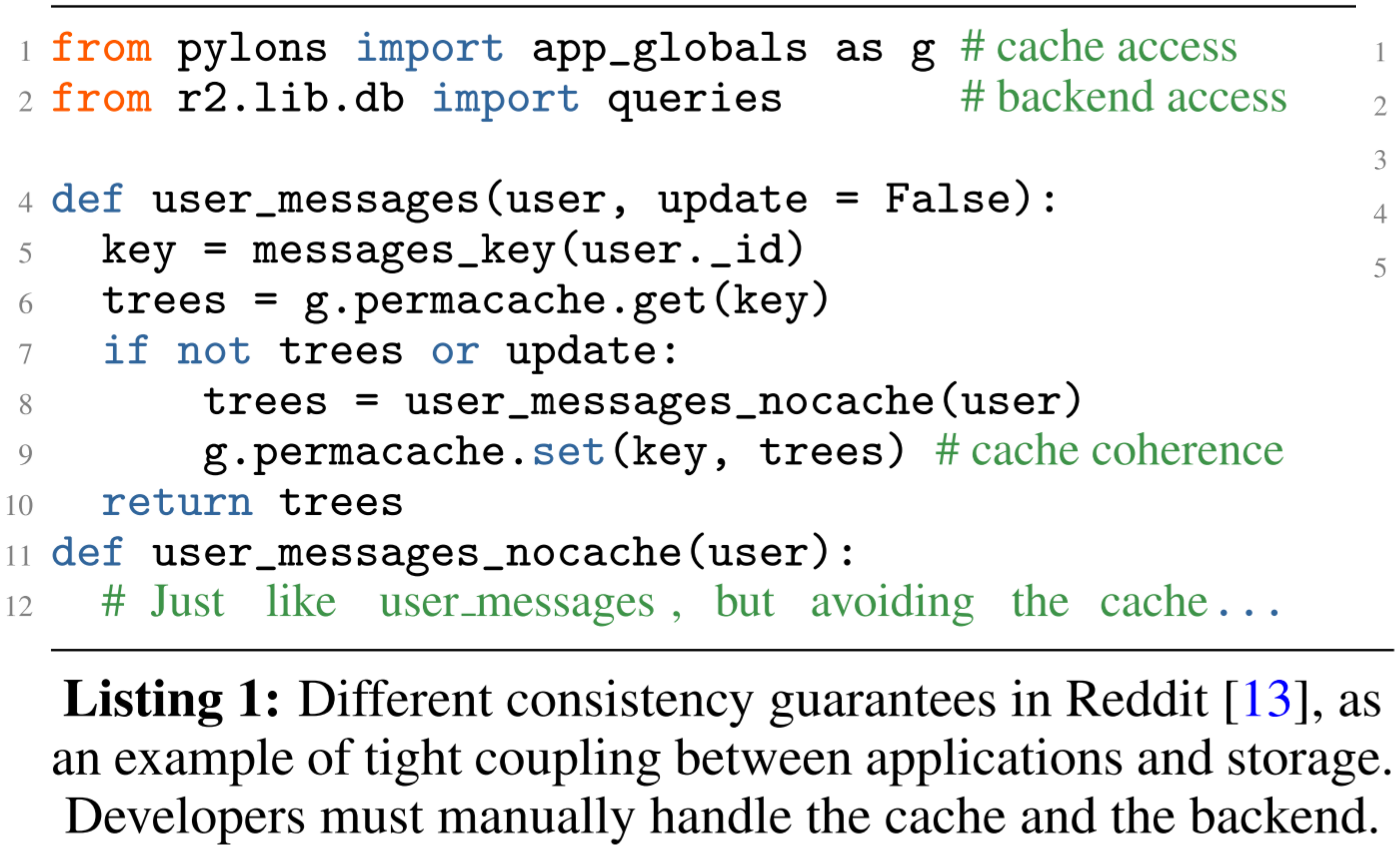


```
def user_messages(user, strong = False):
  key = messages_key(user._id)
  # coherence handled by invoke* functions in bindings
  if strong: return invokeStrong(get(key))
         else: return invokeWeak(get(key))
```

Listing 2: Reddit code rewritten using Correctables.

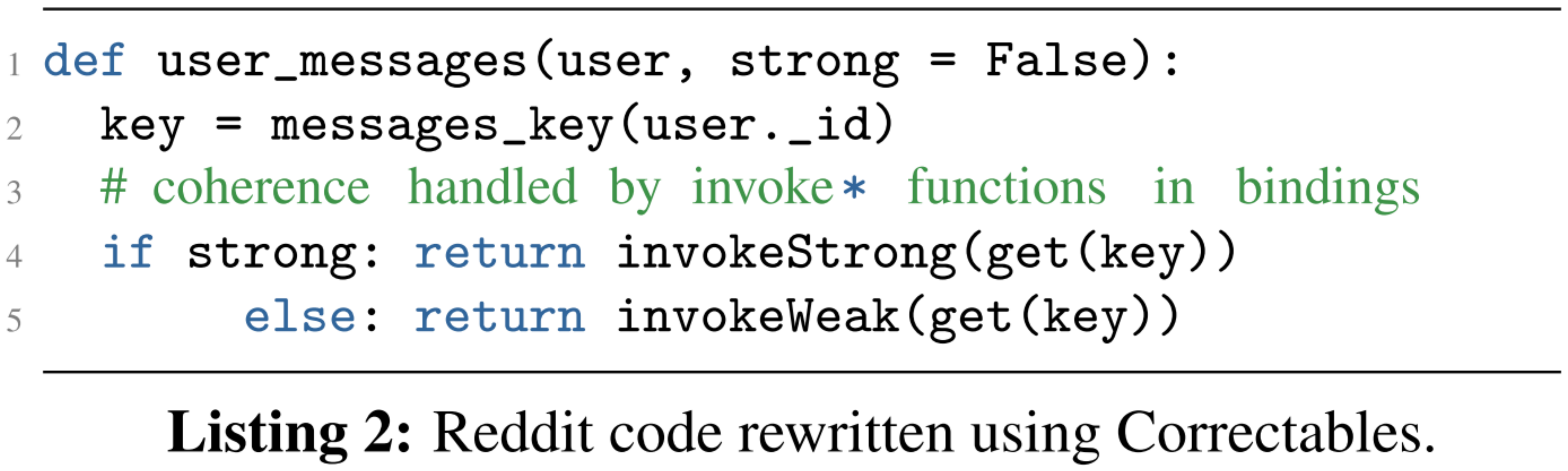


Legacy code vs. Correctables



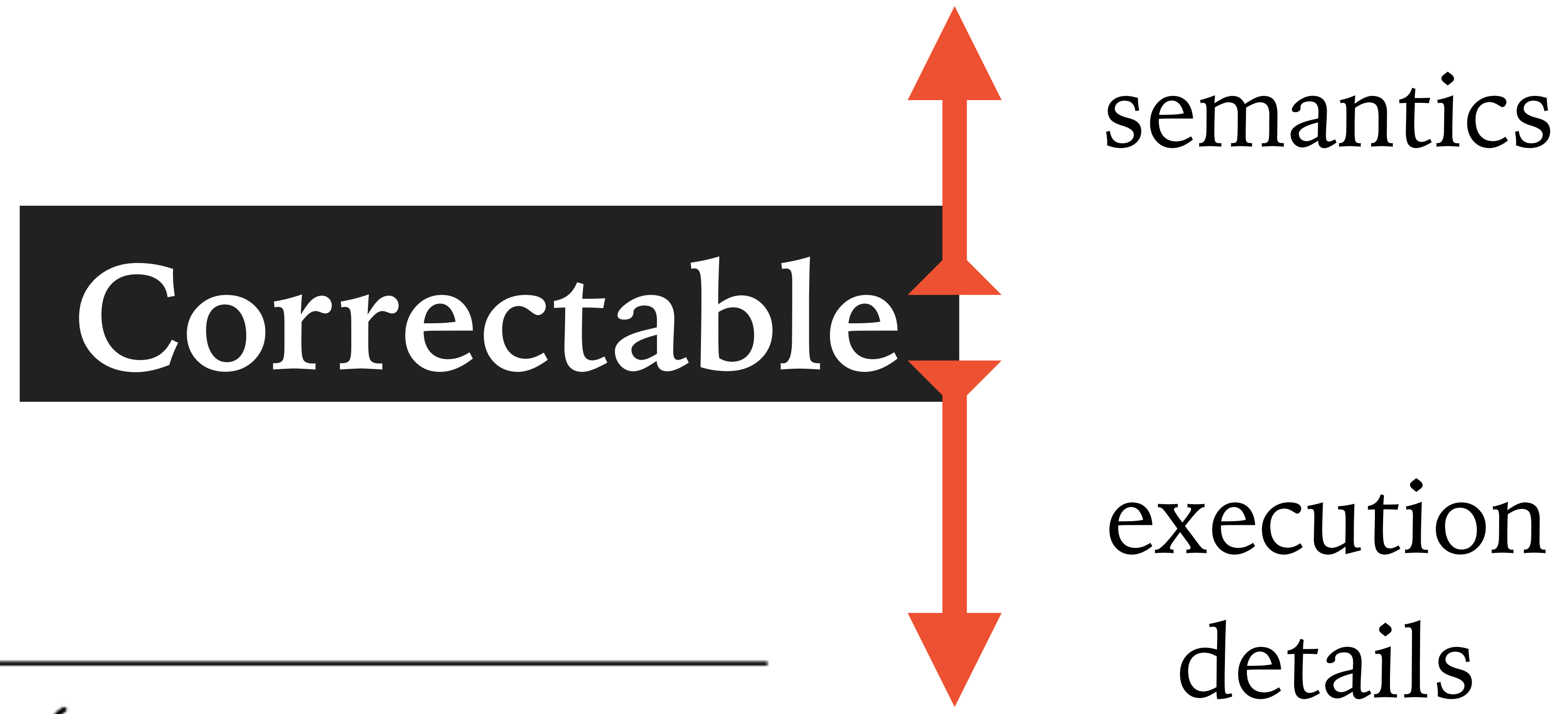
```
from pylons import app_globals as g # cache access
from r2.lib.db import queries       # backend access

def user_messages(user, update = False):
  key = messages_key(user._id)
  trees = g.permacache.get(key)
  if not trees or update:
      trees = user_messages_nocache(user)
      g.permacache.set(key, trees) # cache coherence
  return trees
def user_messages_nocache(user):
  # 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.



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

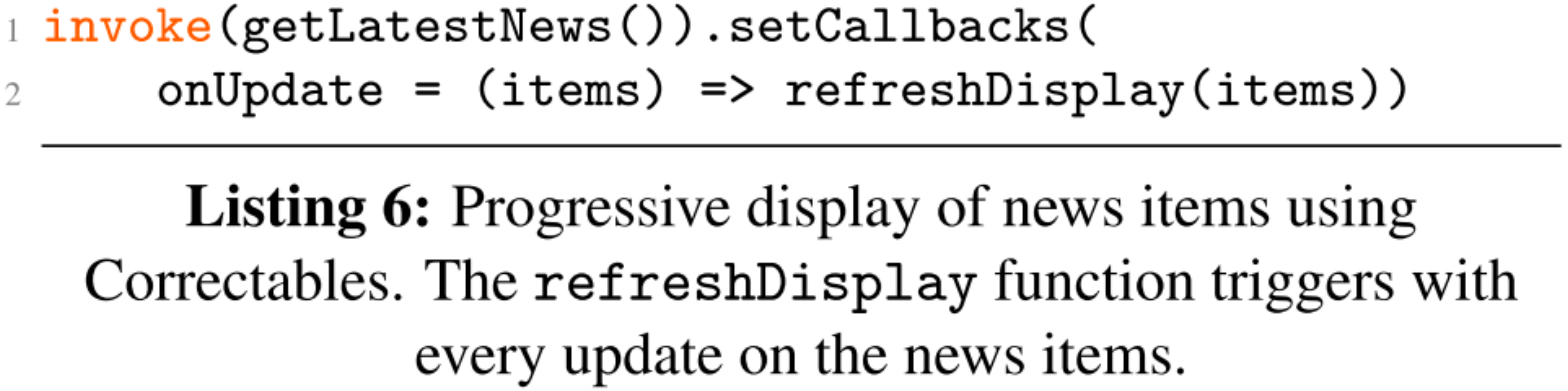
Listing 2: Reddit code rewritten using Correctables.



Correctable

semantics

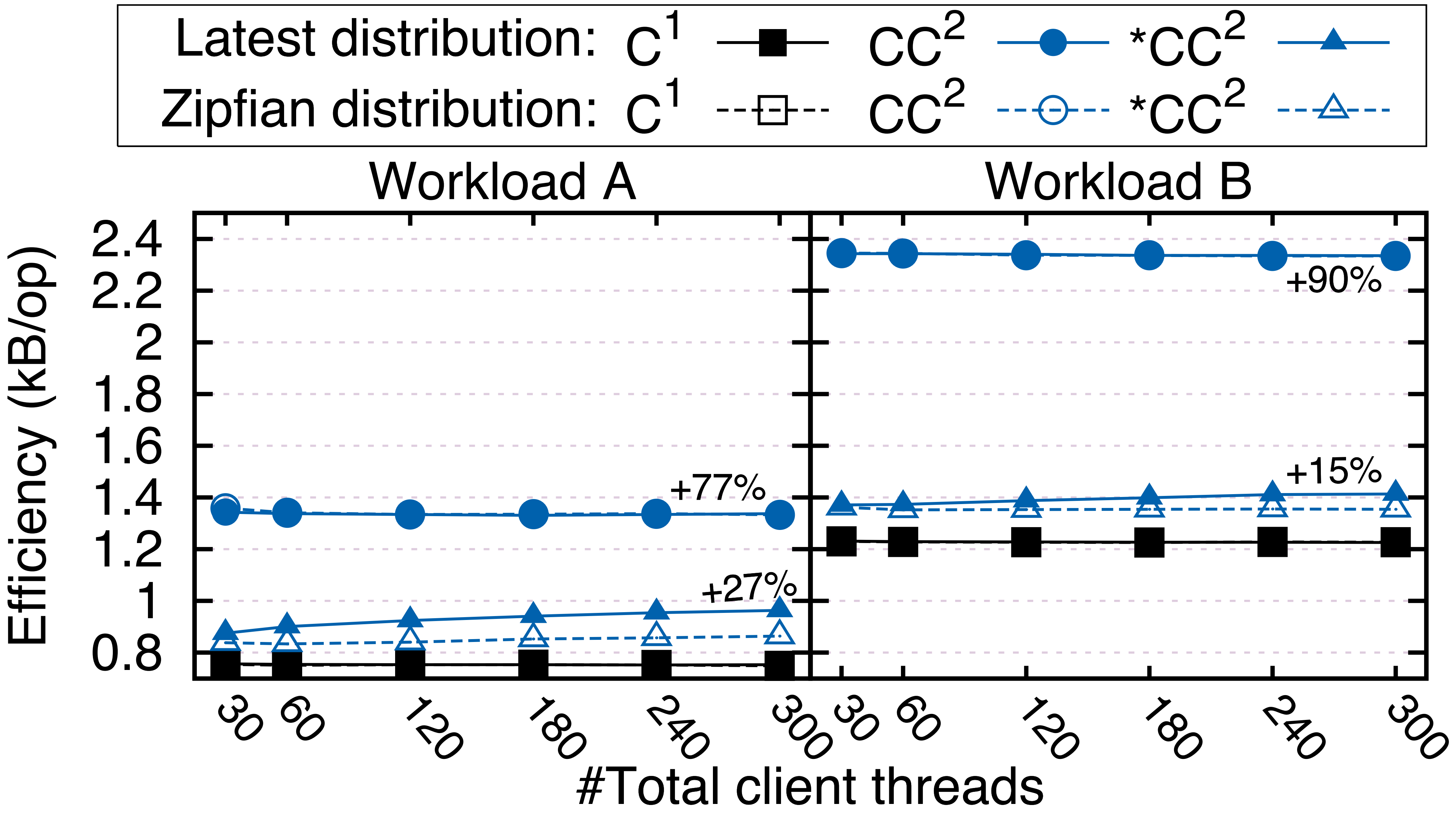
execution details



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

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

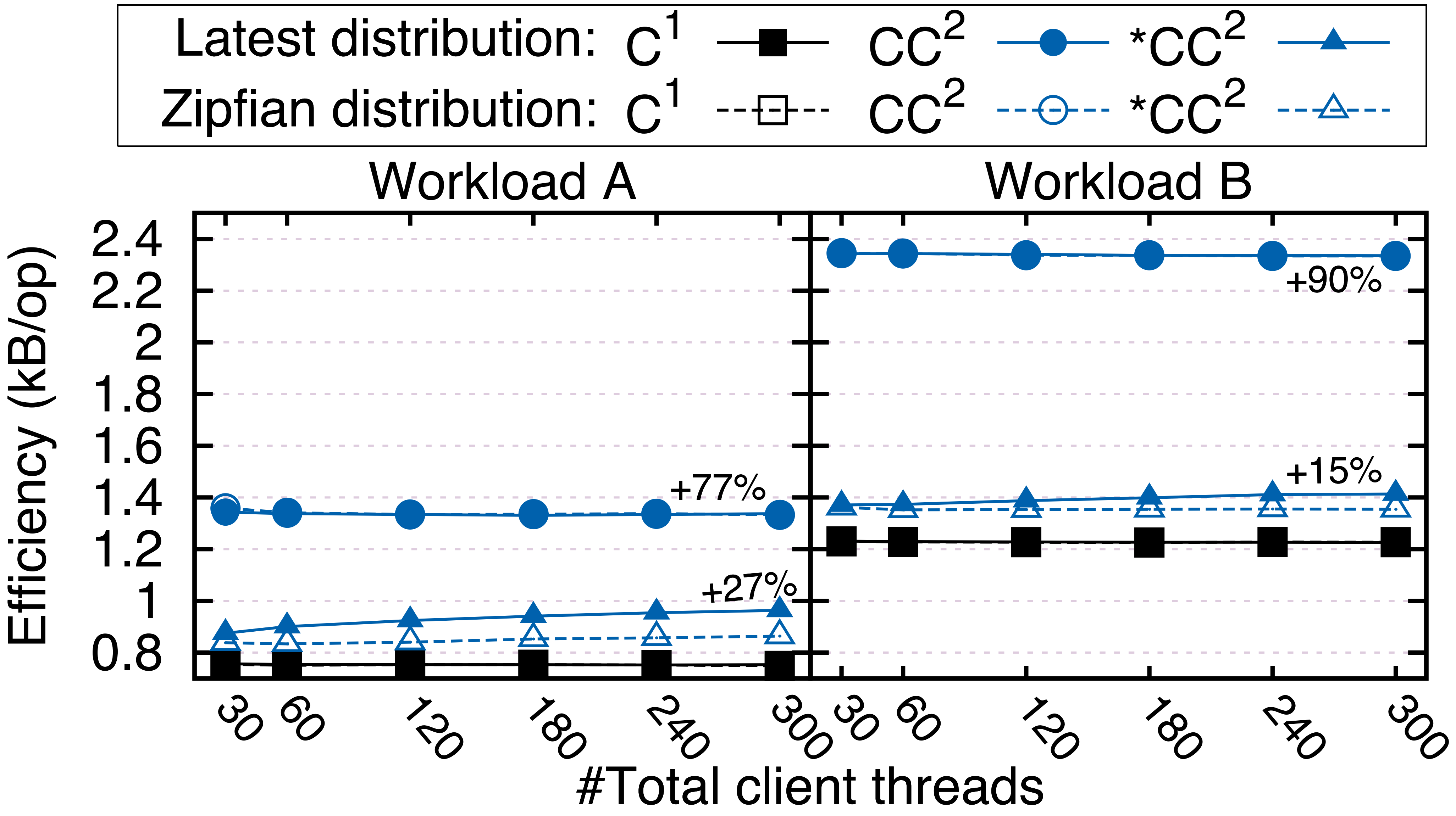
Overheads



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

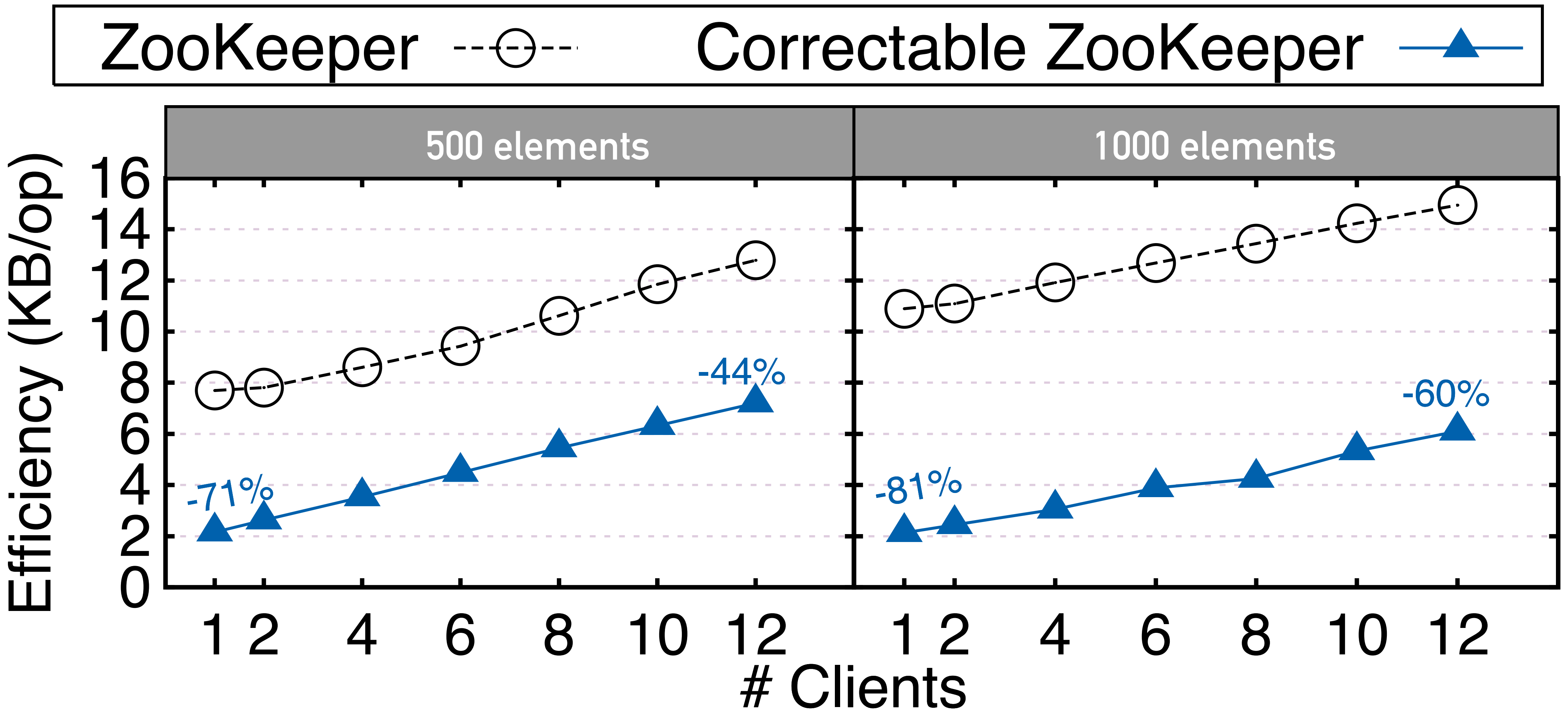
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Overheads



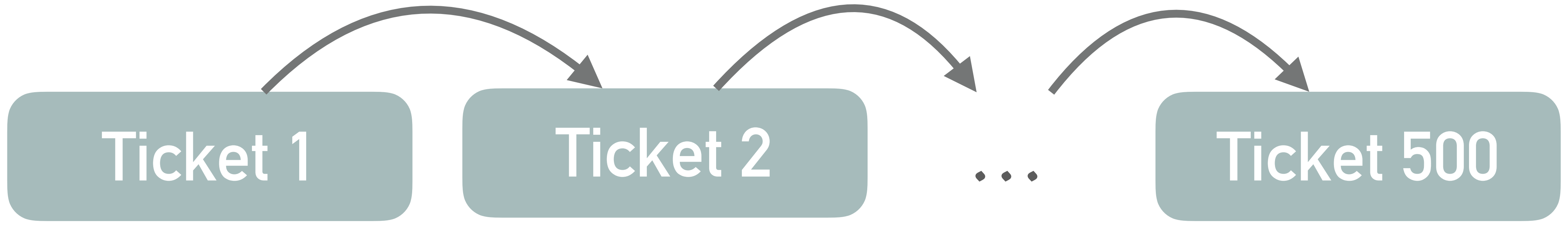
- ➤ Cassandra
- ➤ YCSB workload, various configurations
- ➤ Client in Ireland
- ➤ Replicas in Virginia, Frankfurt, and Ireland

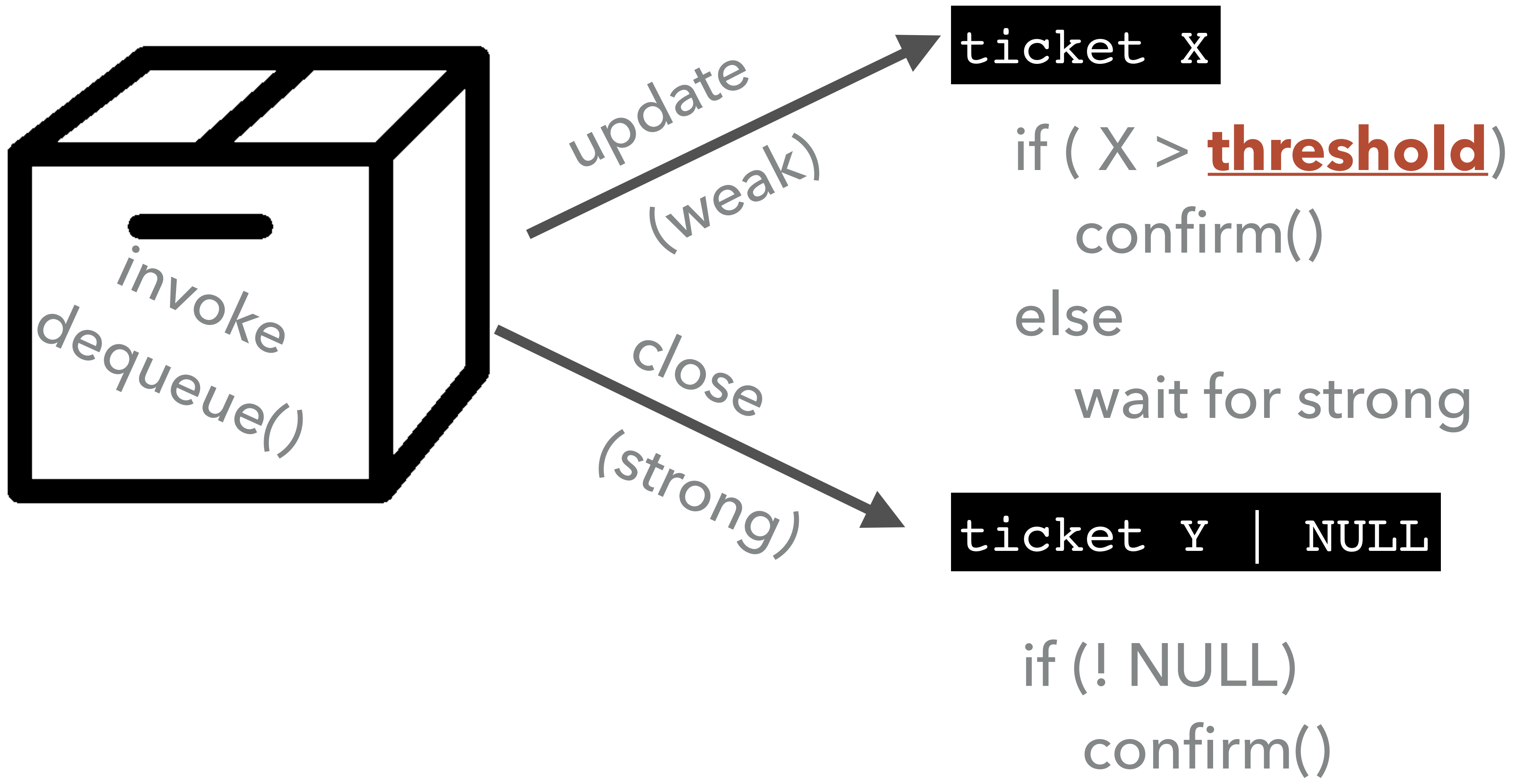
- ➤ ZooKeeper queue implementation
- ➤ Wasteful implementation (by default)
- ➤ We were able to improve — negative overhead



Exploiting application semantics

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



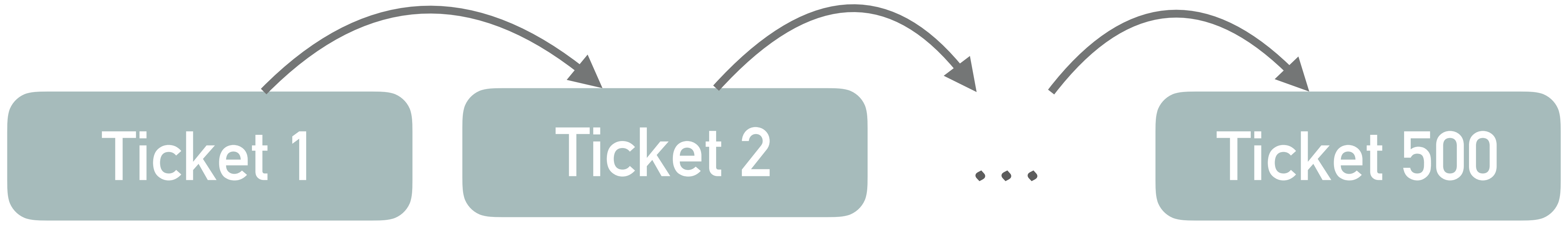


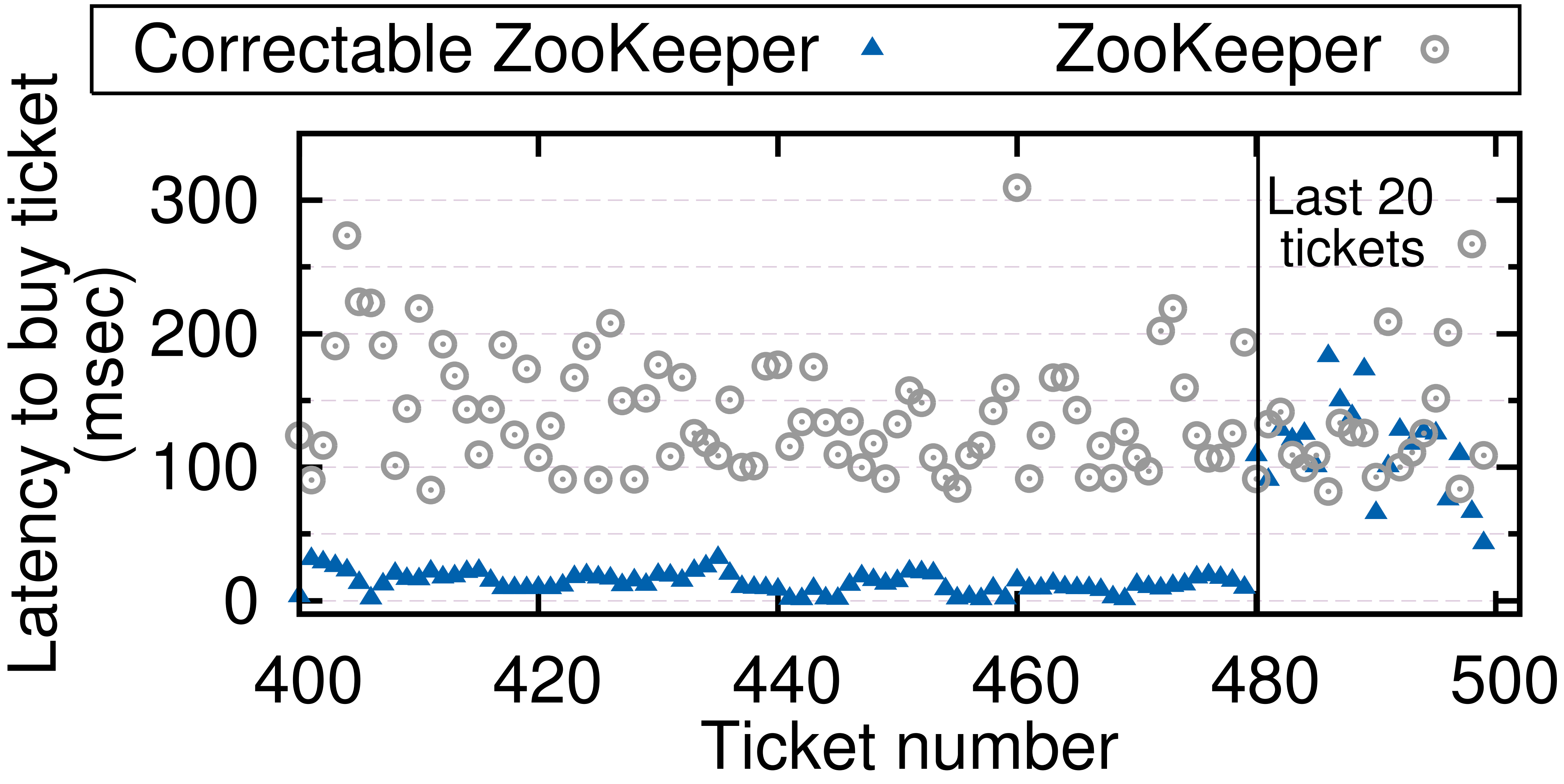
```
if ( X > threshold)
    confirm()
else
    wait for strong
```

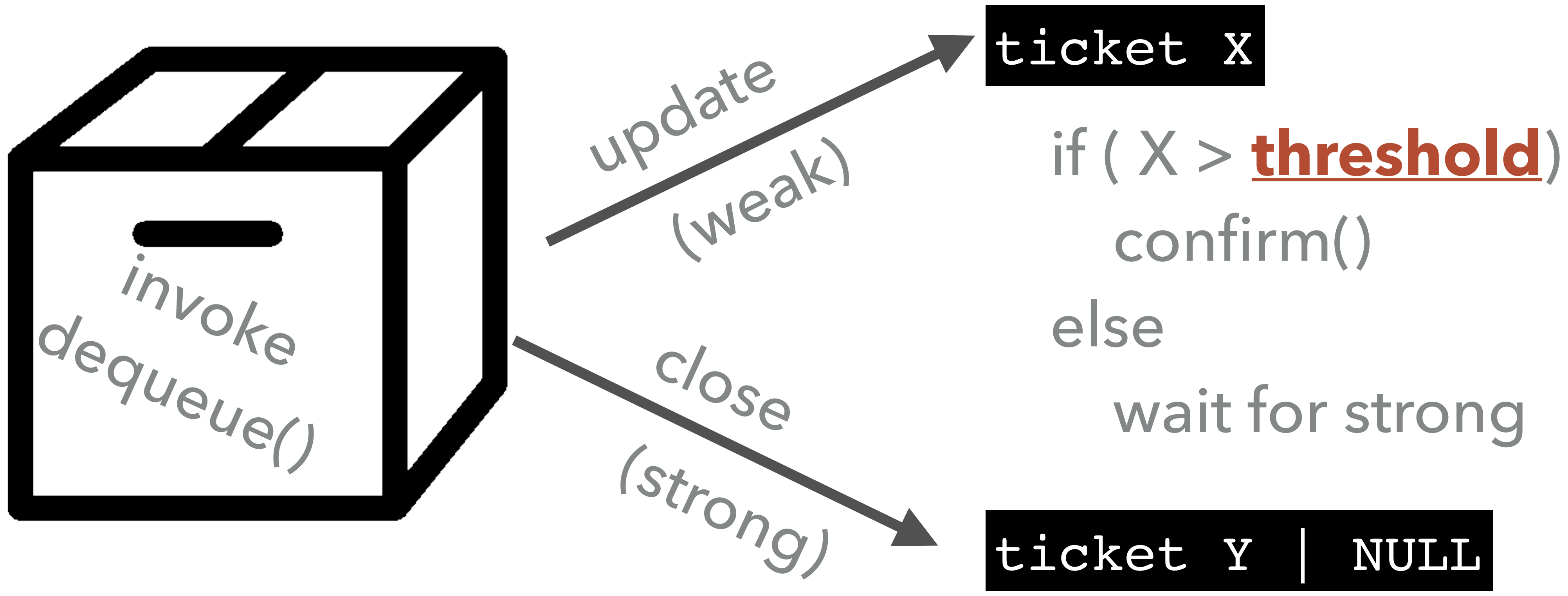
```
if (! NULL)
    confirm()
```

Exploiting application semantics

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



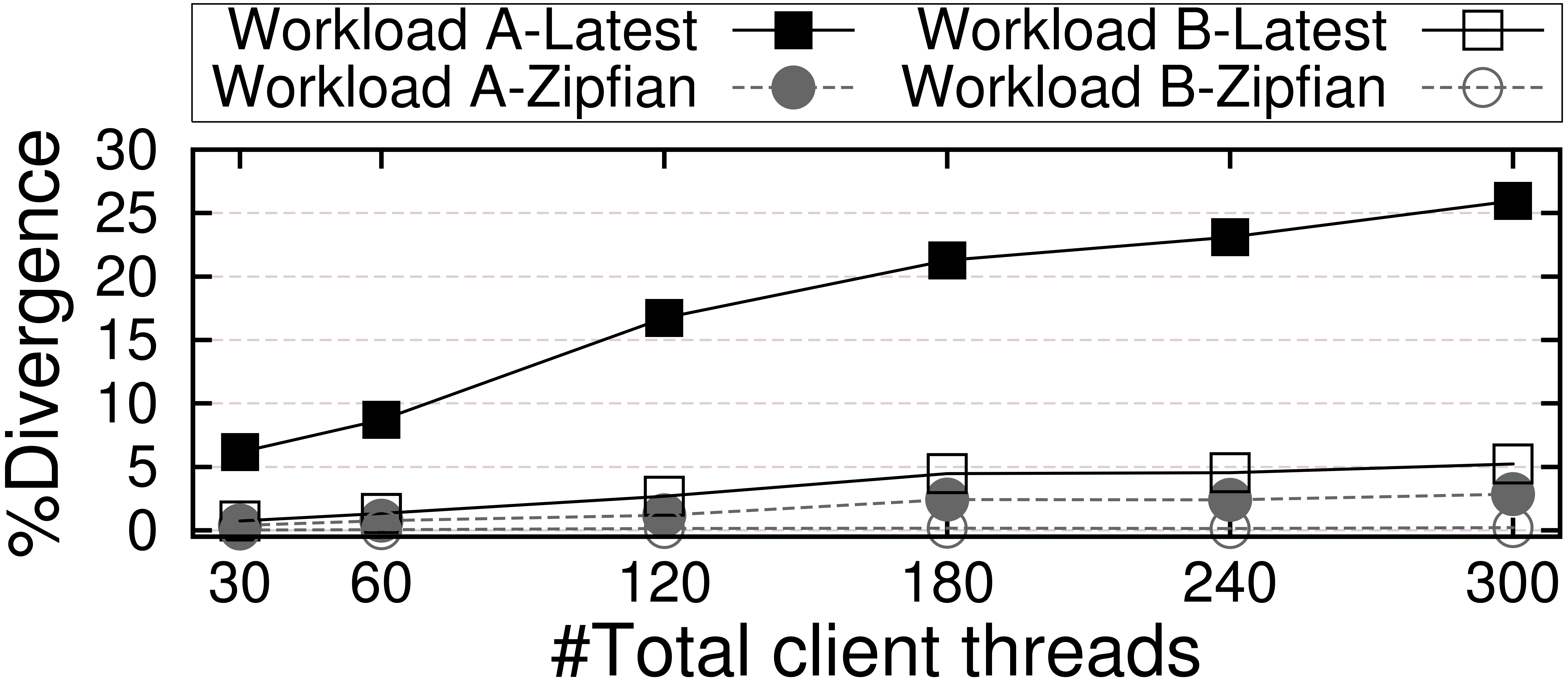




```
if ( X > threshold)
    confirm()
else
    wait for strong
```

```
if (! NULL)
    confirm()
```

Divergence between weak and strong consistency

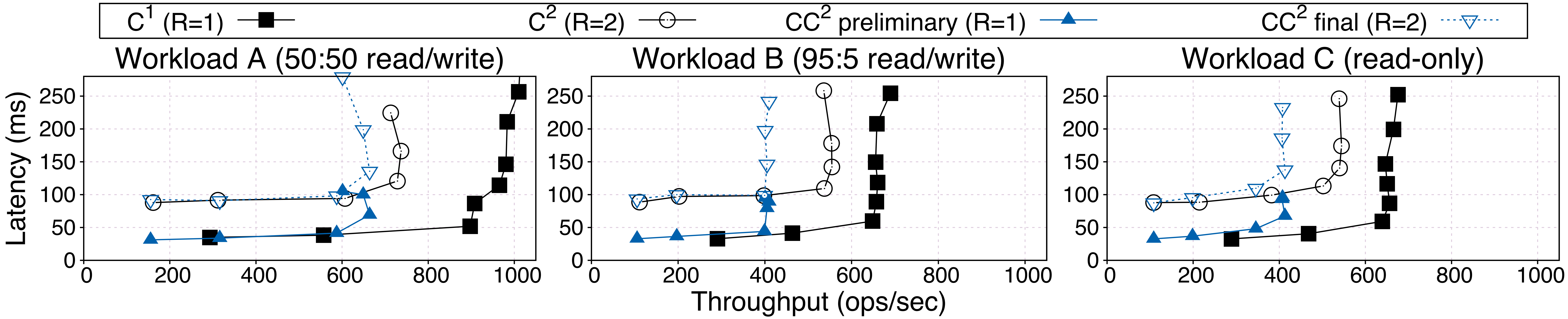


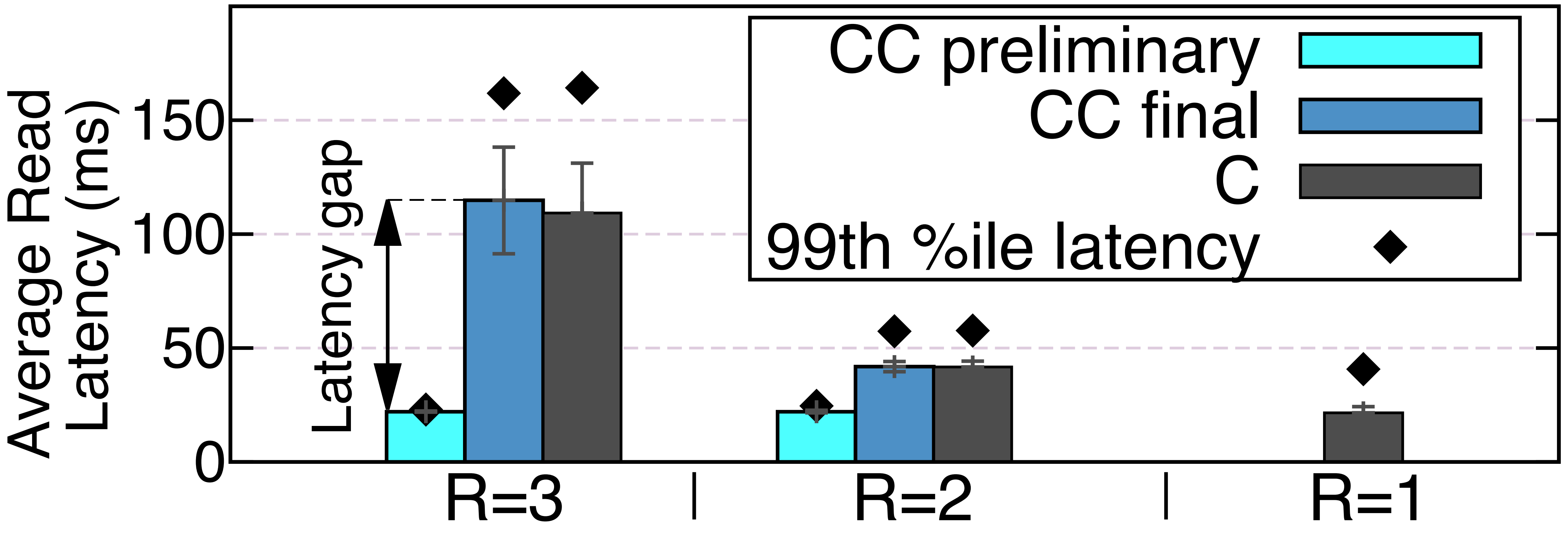
➤ Extremely high

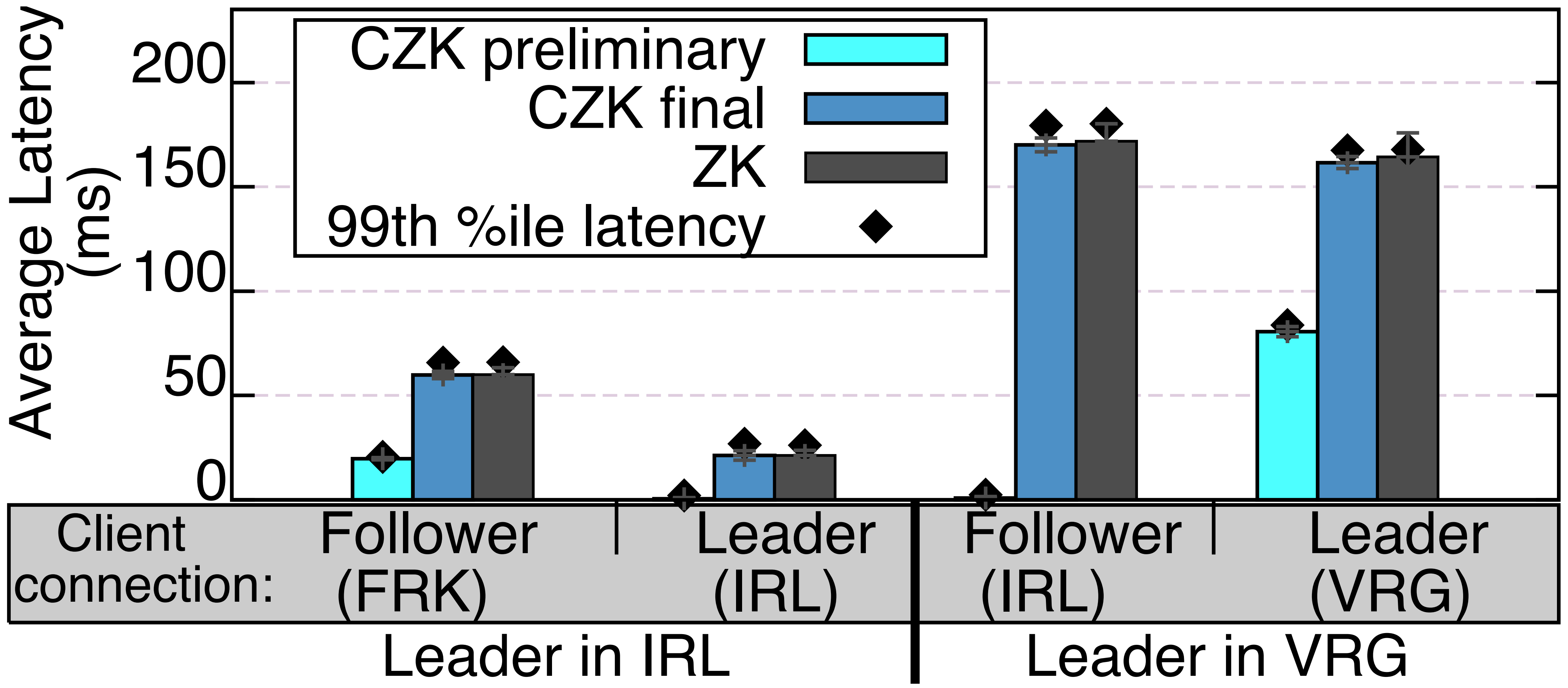
➤ Unusual

} 0.1% — 5% typical range

Latency gaps between consistency models







Efficiency of Multiple Responses

SOCIAL MEDIA APPLICATION

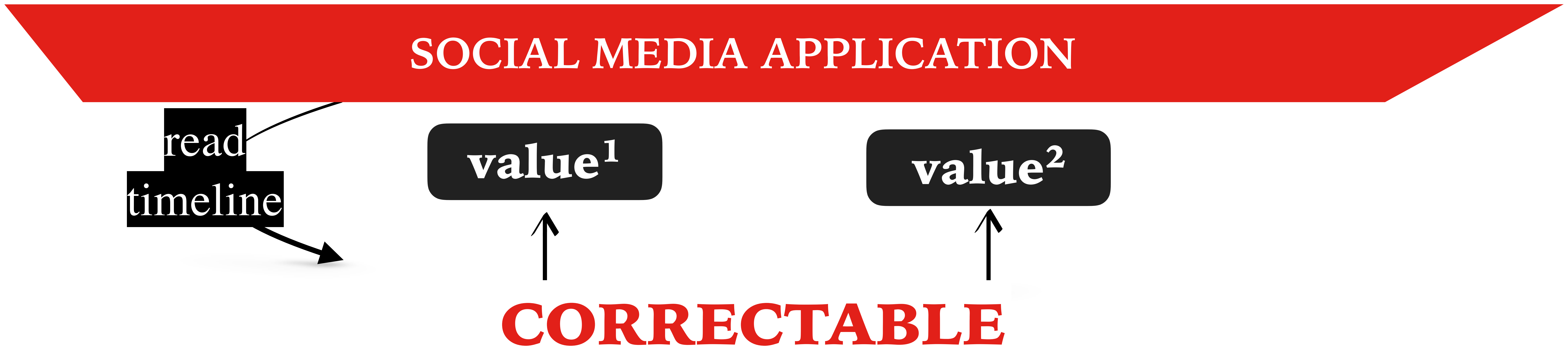
read timeline

value^1

value^2

CORRECTABLE

Efficiency of Multiple Responses



SOCIAL MEDIA APPLICATION

read timeline

value^1

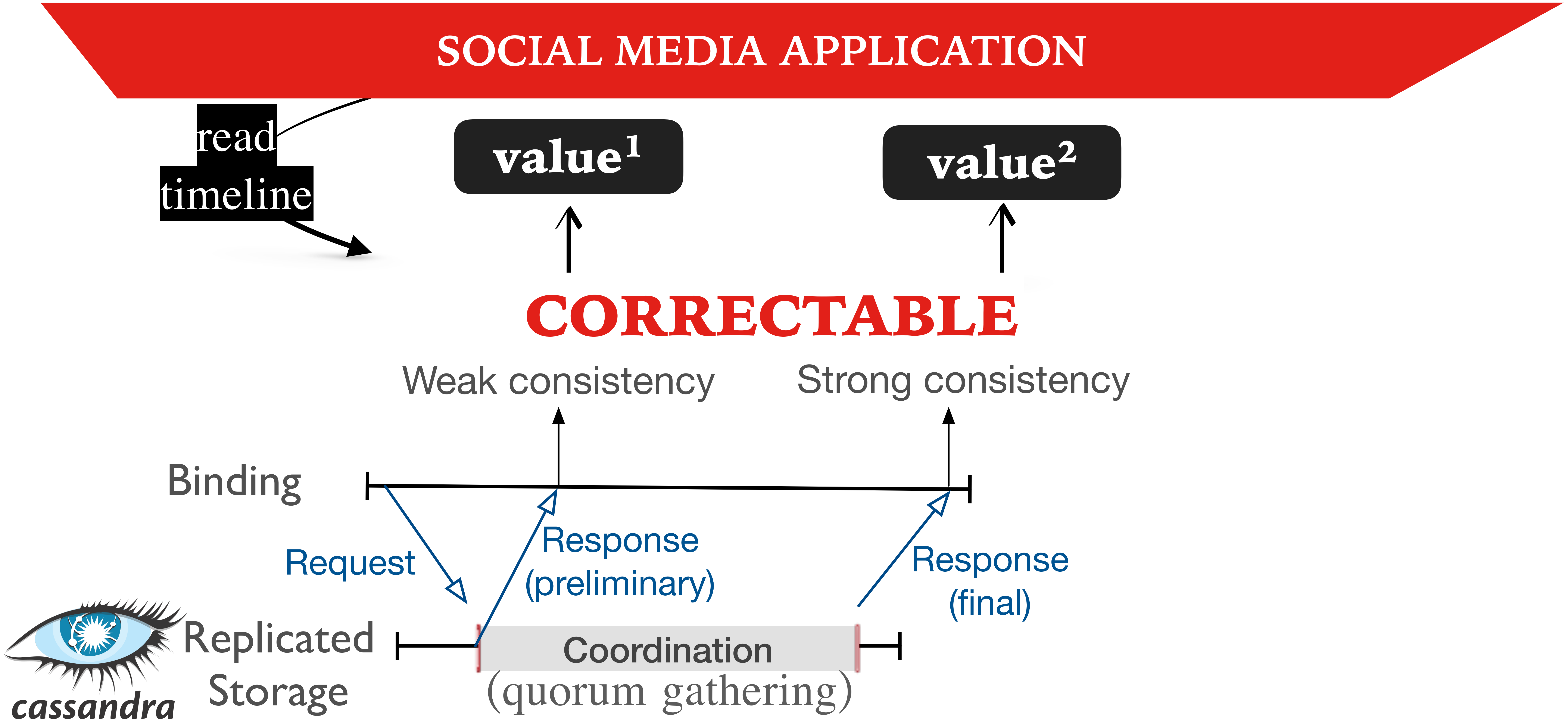
value^2

CORRECTABLE

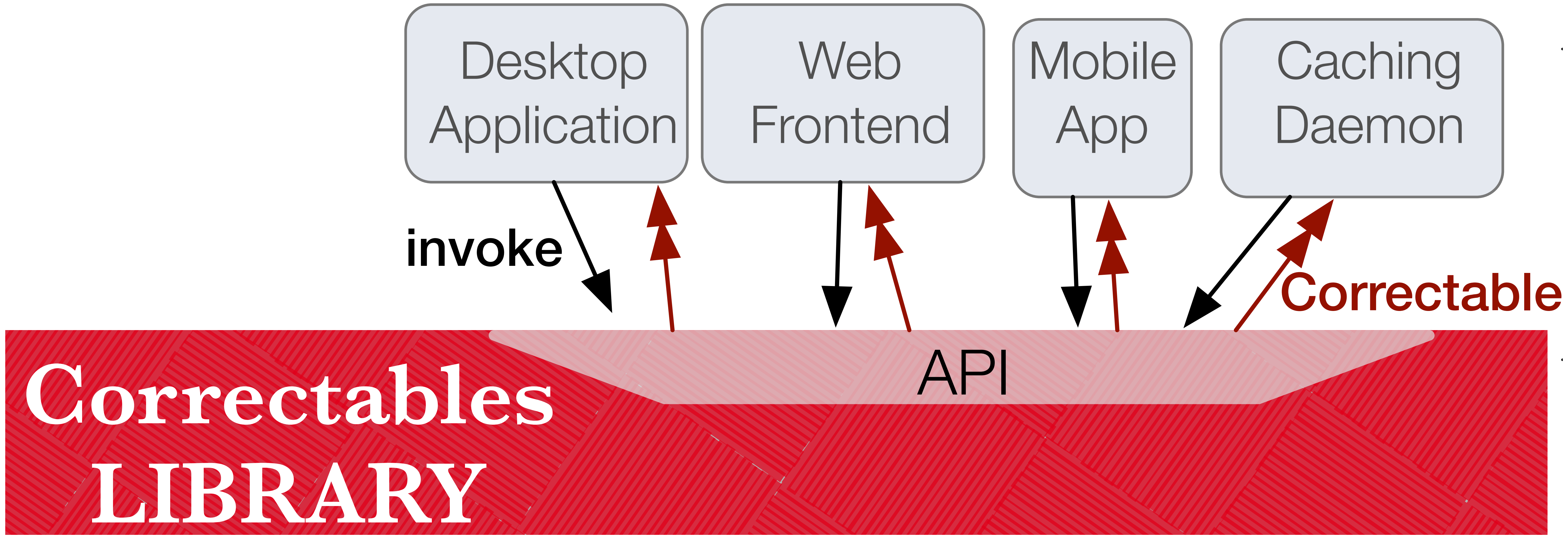
Binding

Replicated Storage

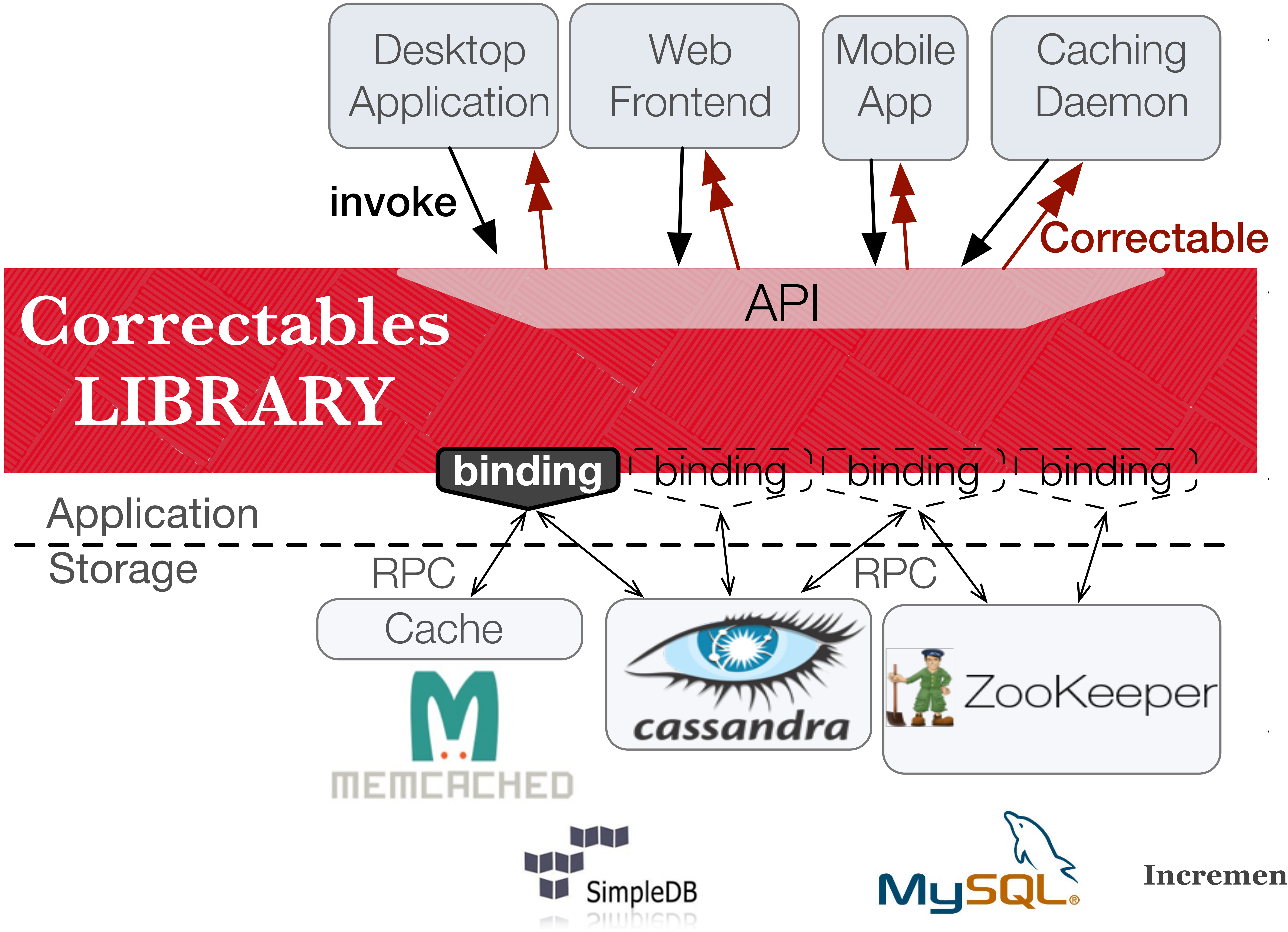
Efficiency of Multiple Responses



Correctables / **Library**

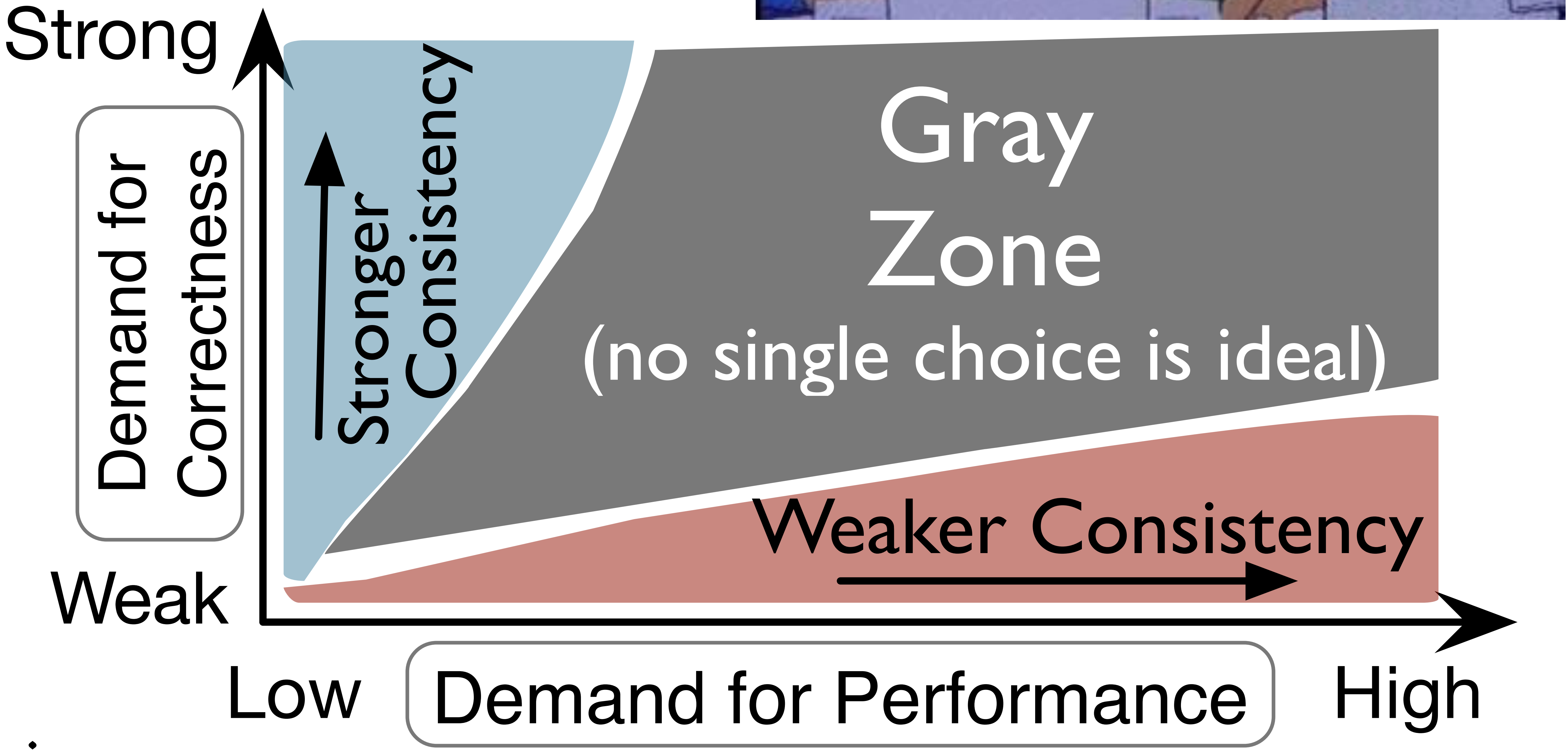


Correctables / **Library**



- Infrastructure services
- Stock tickers
- Trading applications

- E-mail
- Calendar
- Social networks
- Online shopping
- Ad serving
- News reading
- Collaborative editing



Strong

Weak

Demand for Correctness

Stronger Consistency

Gray Zone

(no single choice is ideal)

Weaker Consistency

Low

Demand for Performance

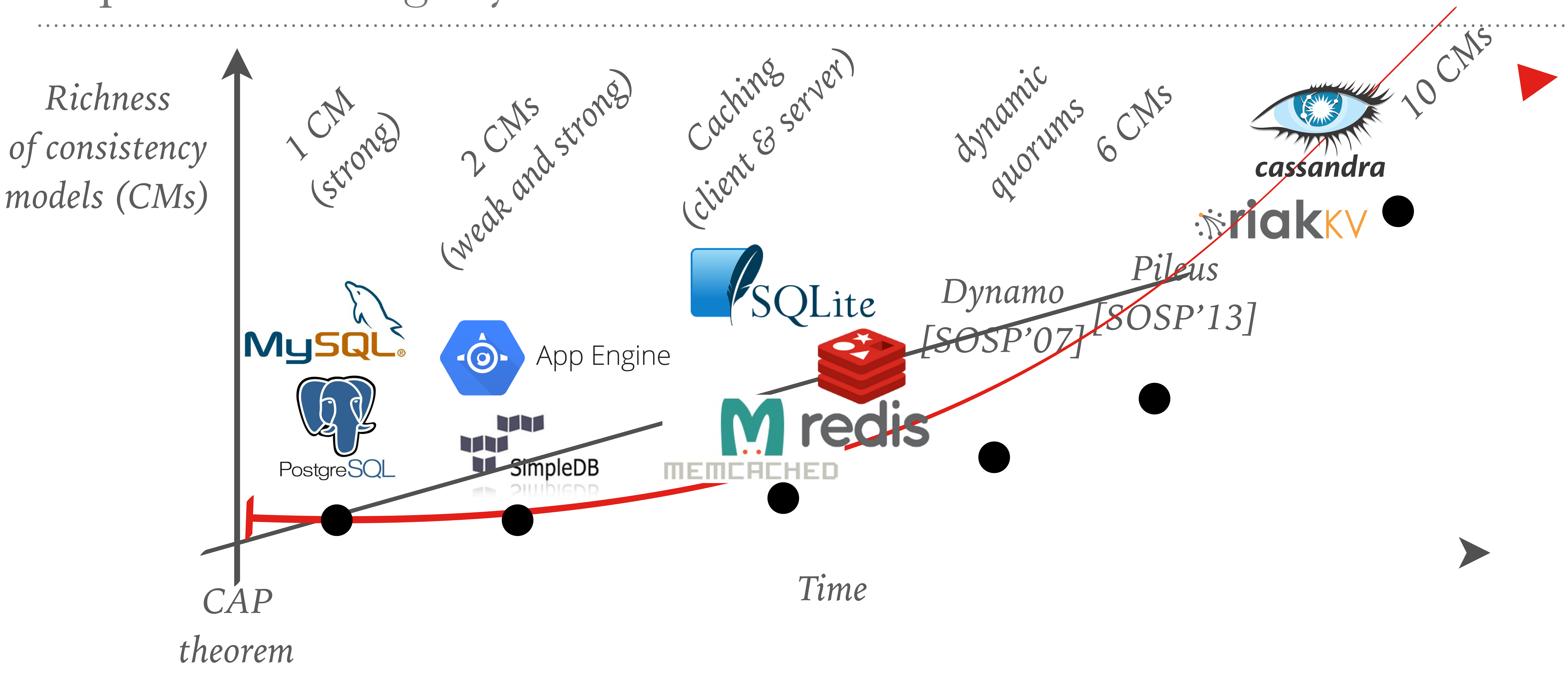
High

Performance is a second-order concern

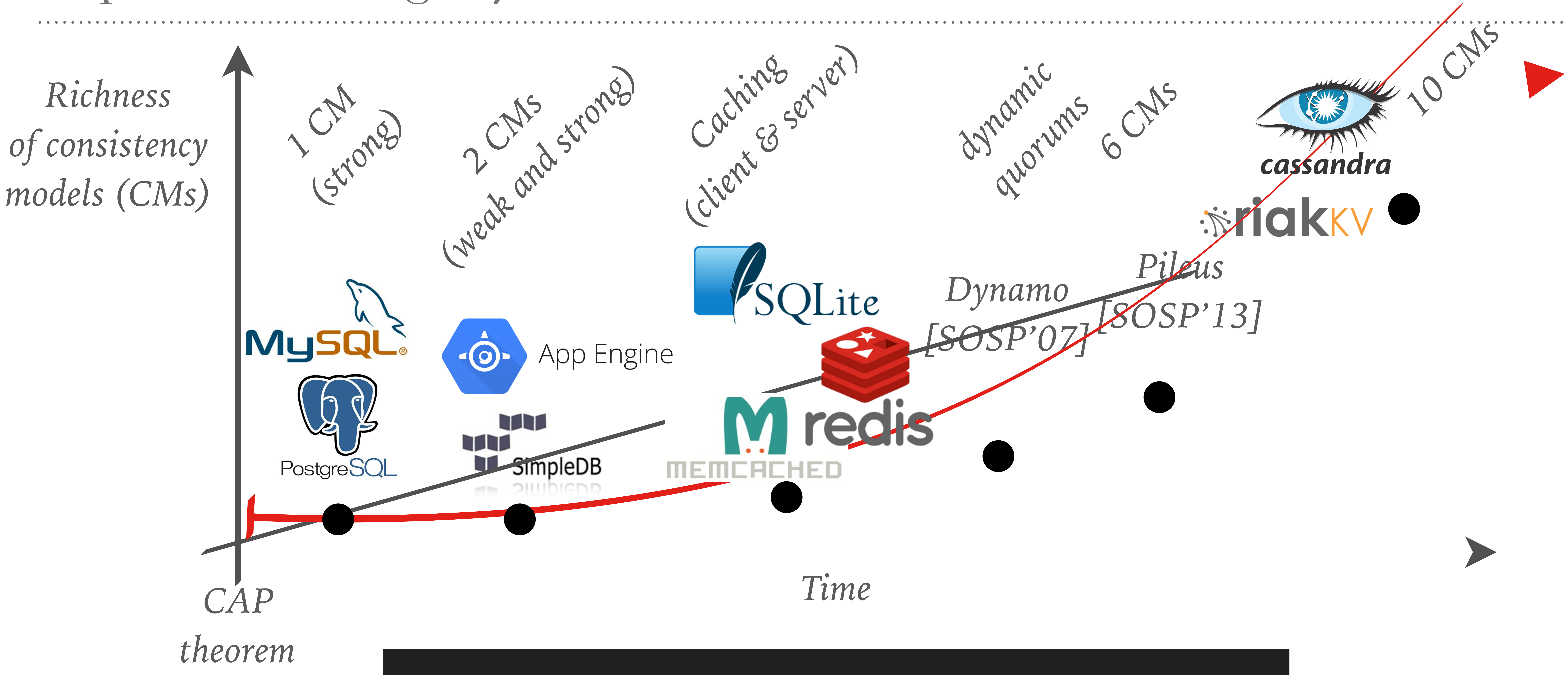
- Computation on static content
- Cold data analysis
- Disconnected operations in mobile applications

Semantics allow to bypass the need for strong consistency

Replicated storage systems



Replicated storage systems



No single consistency model is ideal
→ expose multiple choices