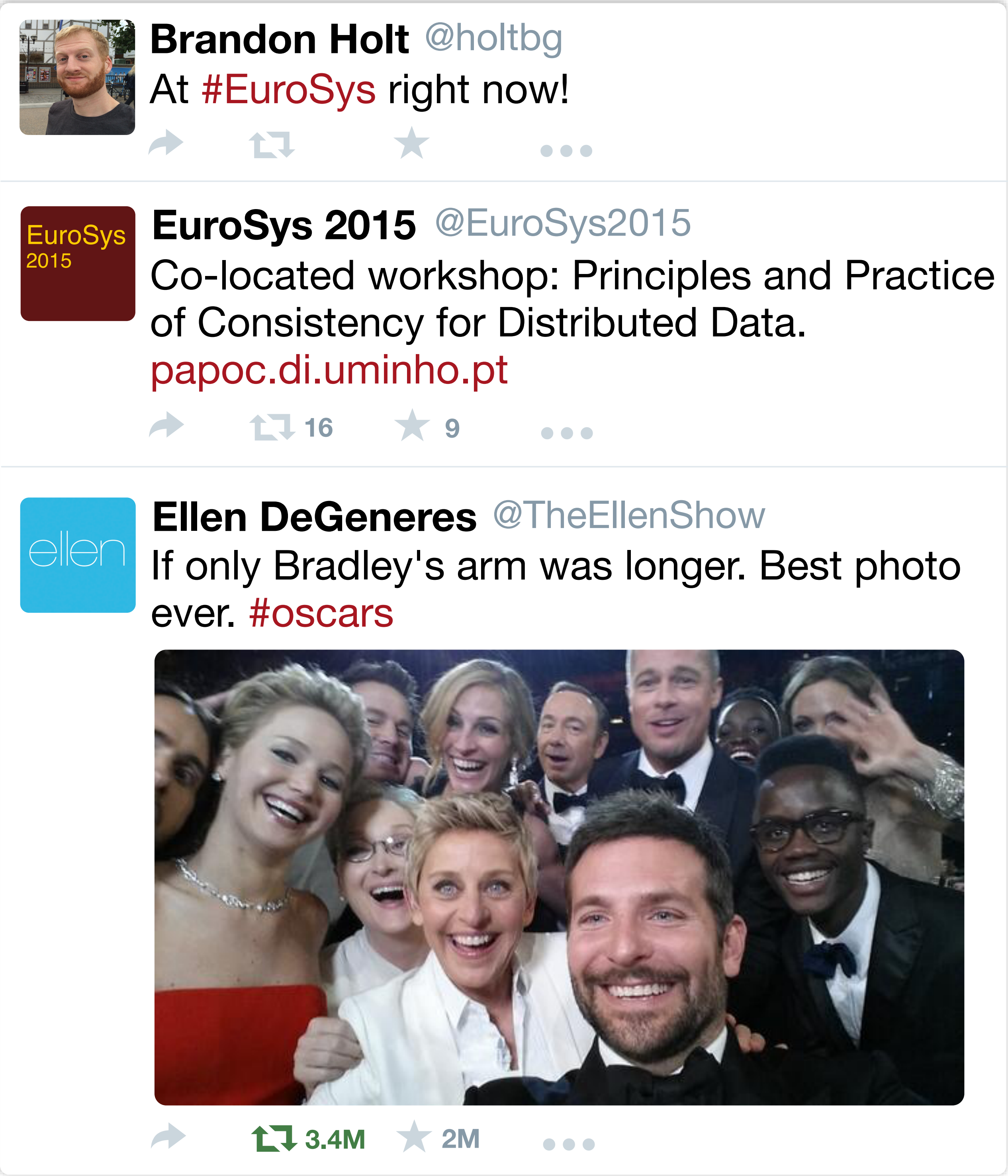
Claret

Using Data Types for High Contention Distributed Transactions

Brandon Holt, Irene Zhang, Dan Ports, Mark Oskin, Luis Ceze

UNIVERSITY *of* WASHINGTON

PaPoC'15 @ EuroSys



Brandon Holt @holtbg

At #EuroSys right now!

EuroSys 2015 @EuroSys2015

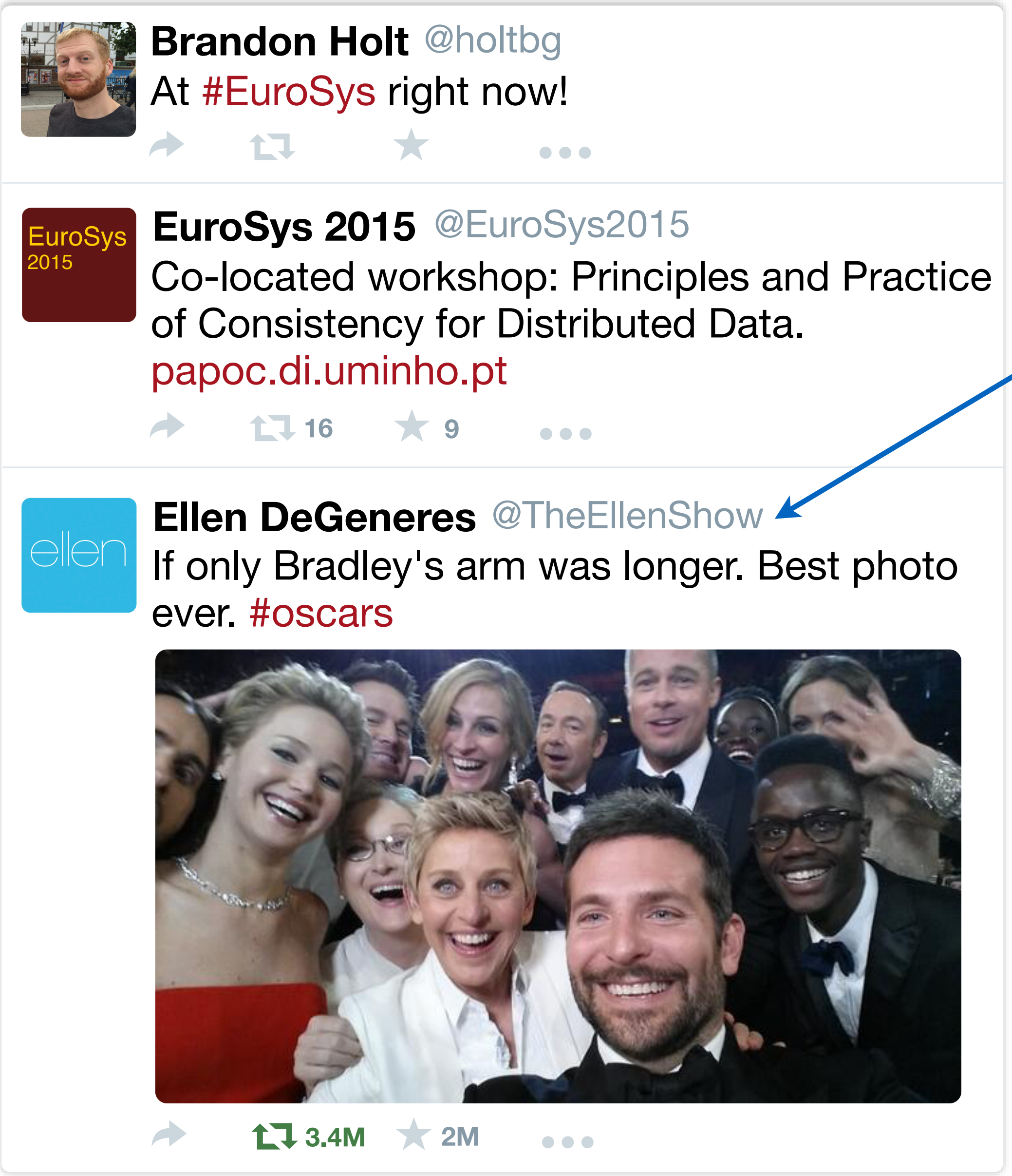
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt

16 ★ 9

Ellen DeGeneres @TheEllenShow

If only Bradley's arm was longer. Best photo ever. #oscars

3.4M ★ 2M



Brandon Holt @holtbg

At #EuroSys right now!

EuroSys 2015 @EuroSys2015

Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt

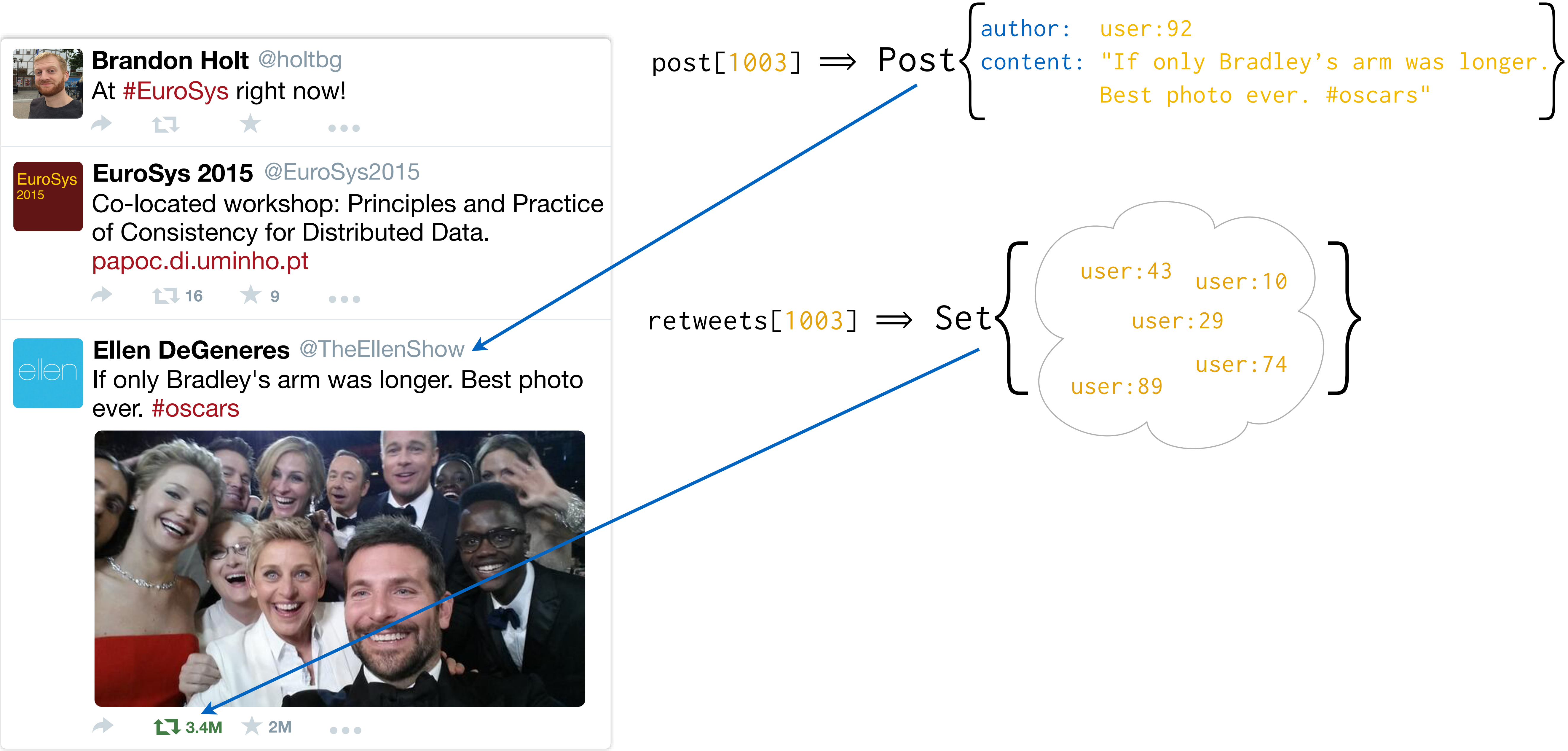
↻ 16 ★ 9

Ellen DeGeneres @TheEllenShow

If only Bradley's arm was longer. Best photo ever. #oscars

↻ 3.4M ★ 2M

```
post[1003] ⟹ Post{ author:  user:92
                    content: "If only Bradley’s arm was longer.
                              Best photo ever. #oscars"       }
```



Brandon Holt @holtbg

At #EuroSys right now!

EuroSys 2015 @EuroSys2015

Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt

16 9

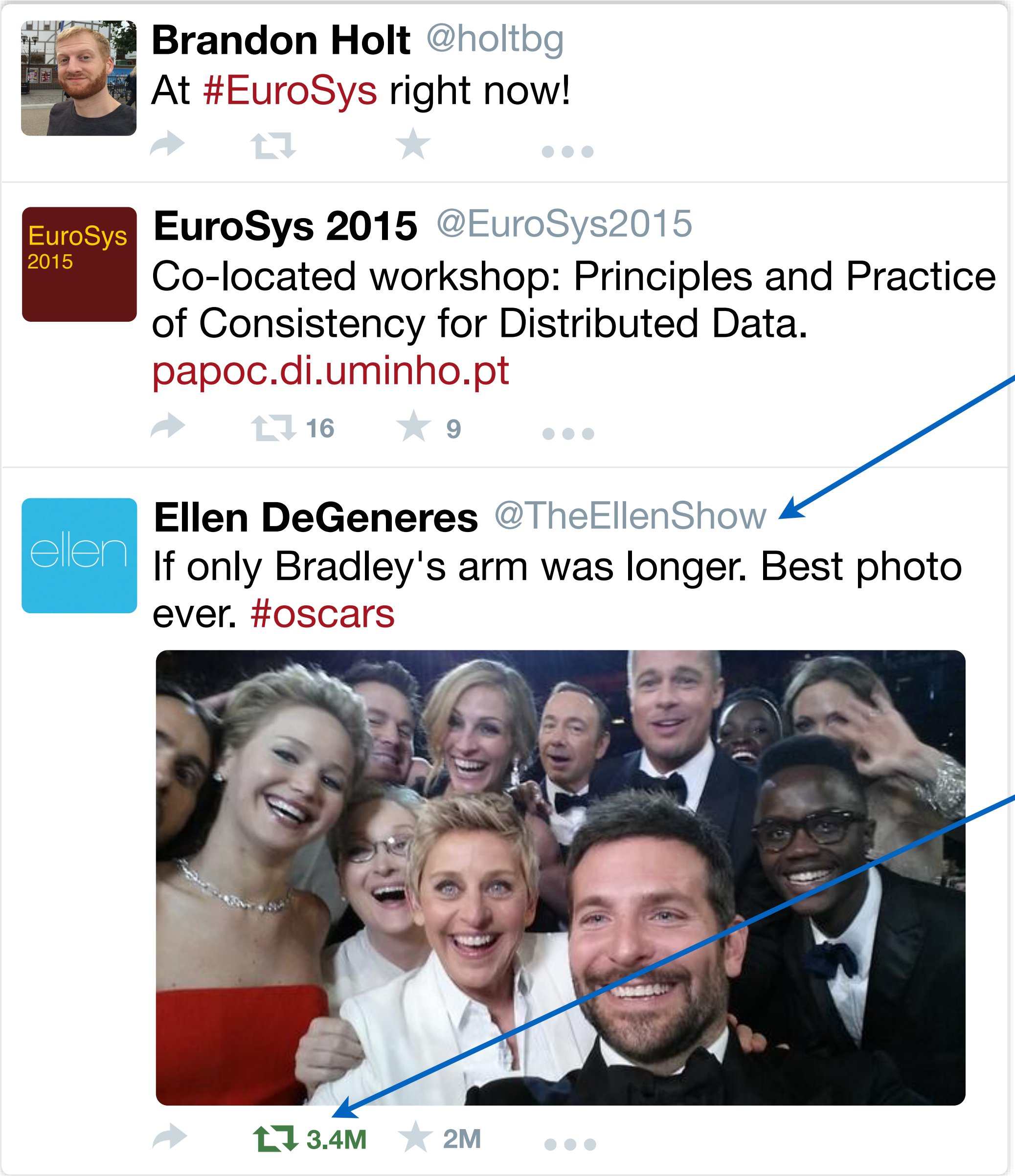
Ellen DeGeneres @TheEllenShow

If only Bradley's arm was longer. Best photo ever. #oscars

3.4M 2M

```
post[1003] ⟹ Post{ author:  user:92
                   content: "If only Bradley's arm was longer.
                             Best photo ever. #oscars" }

retweets[1003] ⟹ Set{ user:43  user:10
                       user:29
                       user:74
                       user:89 }
```



Brandon Holt @holtbg

At #EuroSys right now!

EuroSys 2015 @EuroSys2015

Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt

16 9

Ellen DeGeneres @TheEllenShow

If only Bradley's arm was longer. Best photo ever. #oscars

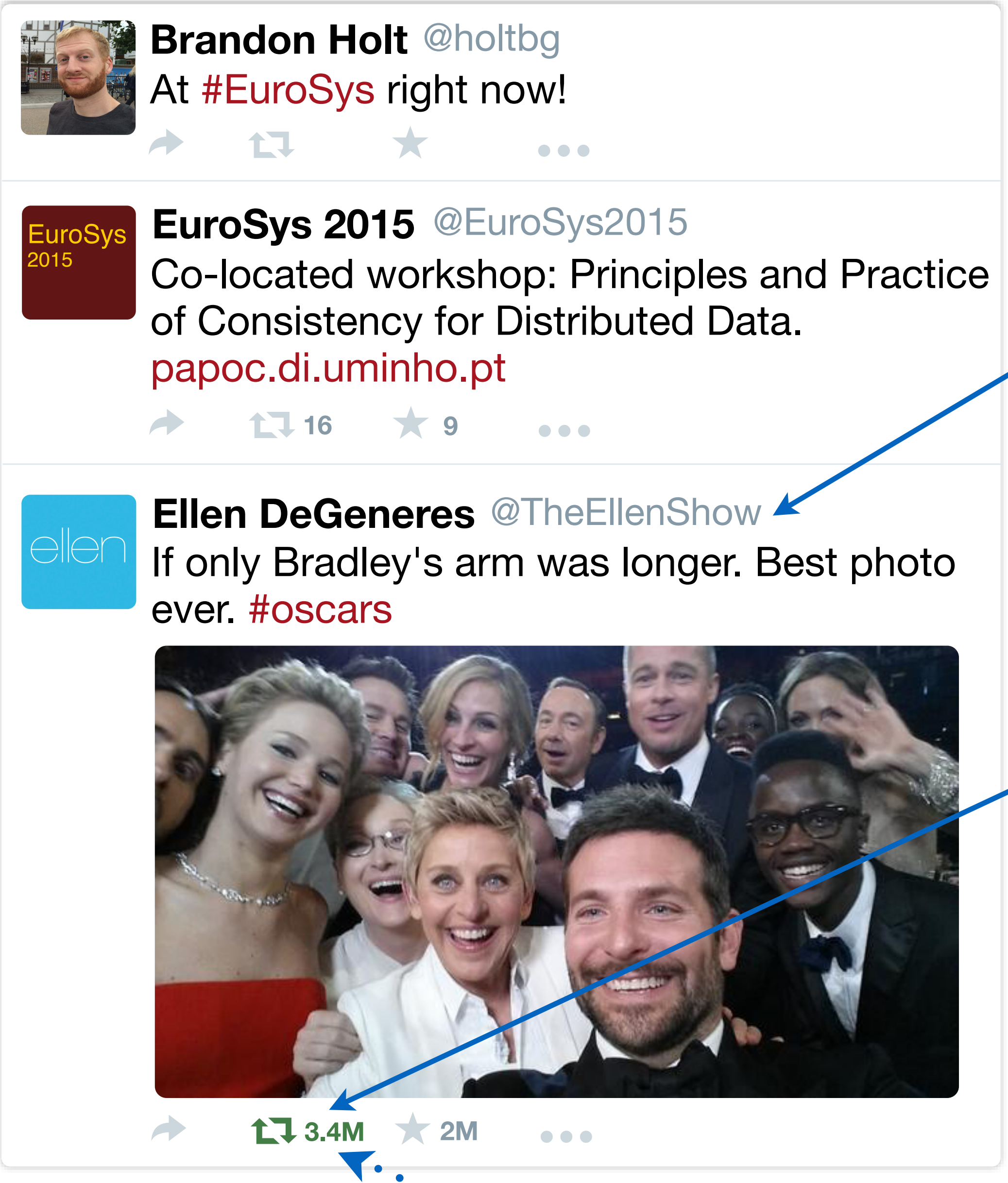
3.4M 2M

```
post[1003] ⟹ Post{author:  user:92
                  content: "If only Bradley's arm was longer.
                            Best photo ever. #oscars"}
```

```
retweets[1003] ⟹ Set{ user:43  user:10  user:29  user:74  user:89 }
```

Retweet

```
retweets[1003].add("user:53")
```



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

```
post[1003] ⟹ Post{author:  user:92
                  content: "If only Bradley's arm was longer.
                           Best photo ever. #oscars"}
```

```
retweets[1003] ⟹ Set{ user:43  user:10
                        user:29
                      user:89   user:74 }
```

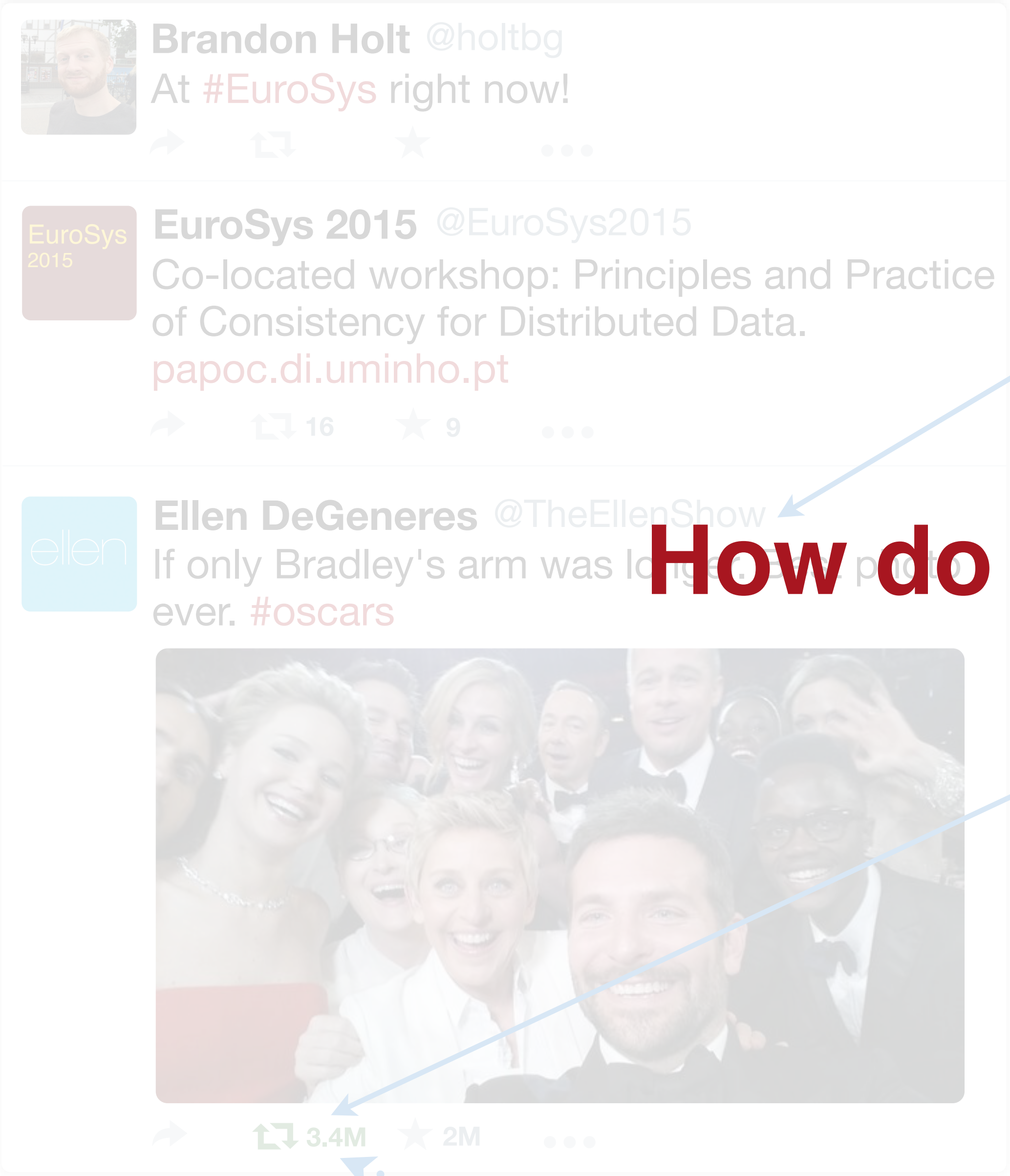
Retweet

```
retweets[1003].add("user:53")
```

View post

```
retweet_count = retweets[1003].size()
# ...
```

How do we make this scale?



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data.
papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

```
post[1003] ⟹ Post{ author:  user:92
                   content: "If only Bradley's arm was longer.
                             Best photo ever. #oscars" }
```

```
retweets[1003] ⟹ Set{ user:43  user:10
                        user:29
                        user:74 }
```

Retweet

```
retweets[1003].add("user:53")
```

View post

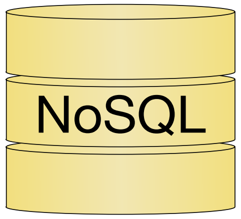
```
retweet_count = retweets[1003].size()
# ...
```

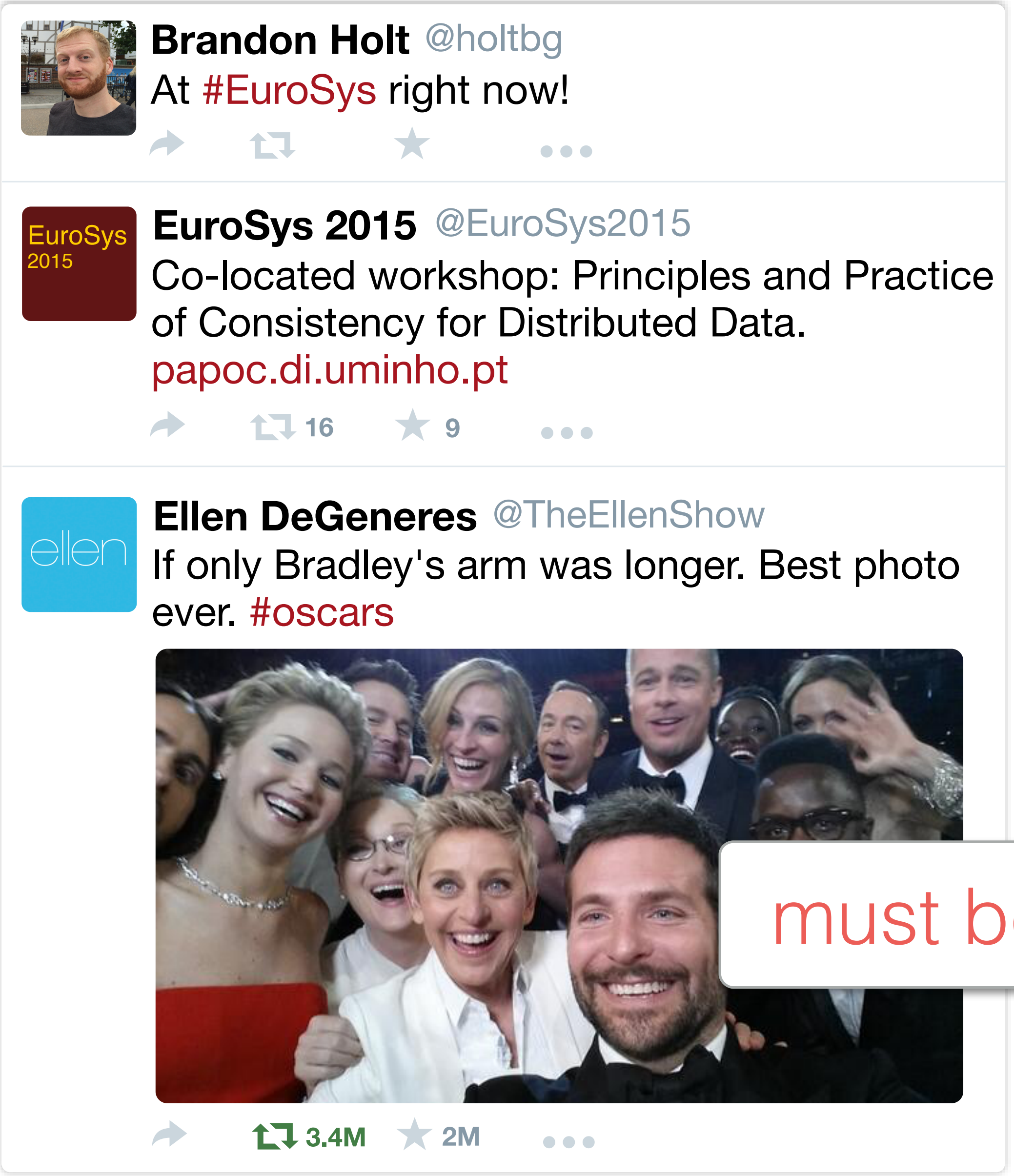
Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2
Co-located workshop: Pri
of Consistency for Distribu
papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEll
If only Bradley's arm was
ever. #oscars
3.4M 2M

```
post[1003] ⟹ Post{ author:  user:92
                   content: "If only Bradley's arm was longer.
                             to ever. #oscars" }
ser:10
9
ser:74
53")
003].size()
```





Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

```
post:1003:author   ⟹ 92
post:1003:content  ⟹ "If only Bradley’s arm was longer.
                      Best photo ever. #oscars"

retweeters:1003    ⟹ "user29,user:89,user:74,
                      user:10,user:43"
```

Retweet

```
s = get("retweeters:1003")
if "user:43" not in s:
  s += "user:43"
  put("retweeters:1003", s)
```

must be atomic

View post

```
retweets = get("retweeters:1003")
# ...
```

which retweets will this contain?

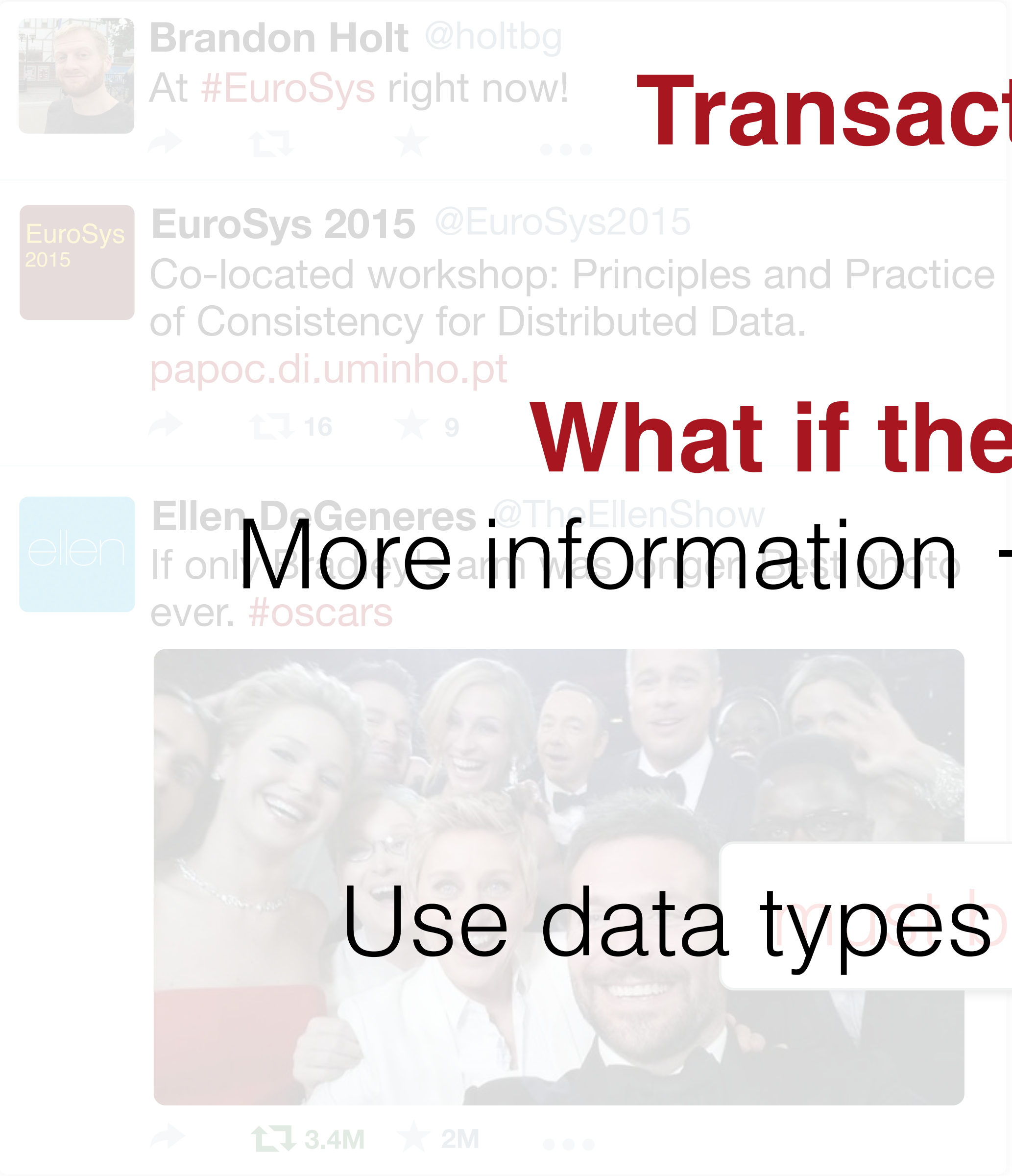
Transactions? *"Too expensive."*
"Don’t scale."

What if the datastore knew more?

More information → more chance for optimization

Opportunity:

Use data types provided by the programmer


Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data.
papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

```
post:1003:author   ⟹ 92
post:1003:content  ⟹ "If only Bradley’s arm was longer.
                      Best photo ever. #oscars"
retweeters:1003    ⟹ "user:29,user:89,user:74,
                      user:10,user:43"
```

Retweet

```
s = get("retweeters:1003")
if "user:43" not in s:
  s += "user:43"
  put("retweeters:1003", s)
```

must be atomic

View post

```
retweets = get("retweeters:1003")
# ...
```

which retweets will this contain?

NoSQL

Abstract Data Types in **NoSQL**

- programmers express *intent* through types
- *flexible* data model, no fixed schema
- *leverage ADT properties* for transaction performance
- *sanely* trade off consistency for scalability

```
s = get("retweeters:1003")
if "user:43" not not in s:
  s += "user:43"
  put("retweeters:1003", s)
```

3.4M 2M

mutate atomic

redis

riak

View post

```
retweets = get("retweeters:1003")
# ...
```

which retweets will this contain?

Leveraging **Abstract Data Types** in **NoSQL**

Commutativity

- Transactional boosting
- Combining

Approximate data types

- Bounded inconsistency
- Isolated eventual consistency (CRDTs)
- Probabilistic data types

Evaluation: *Claret* prototype

Leveraging **Abstract Data Types** in **NoSQL**

Commutativity

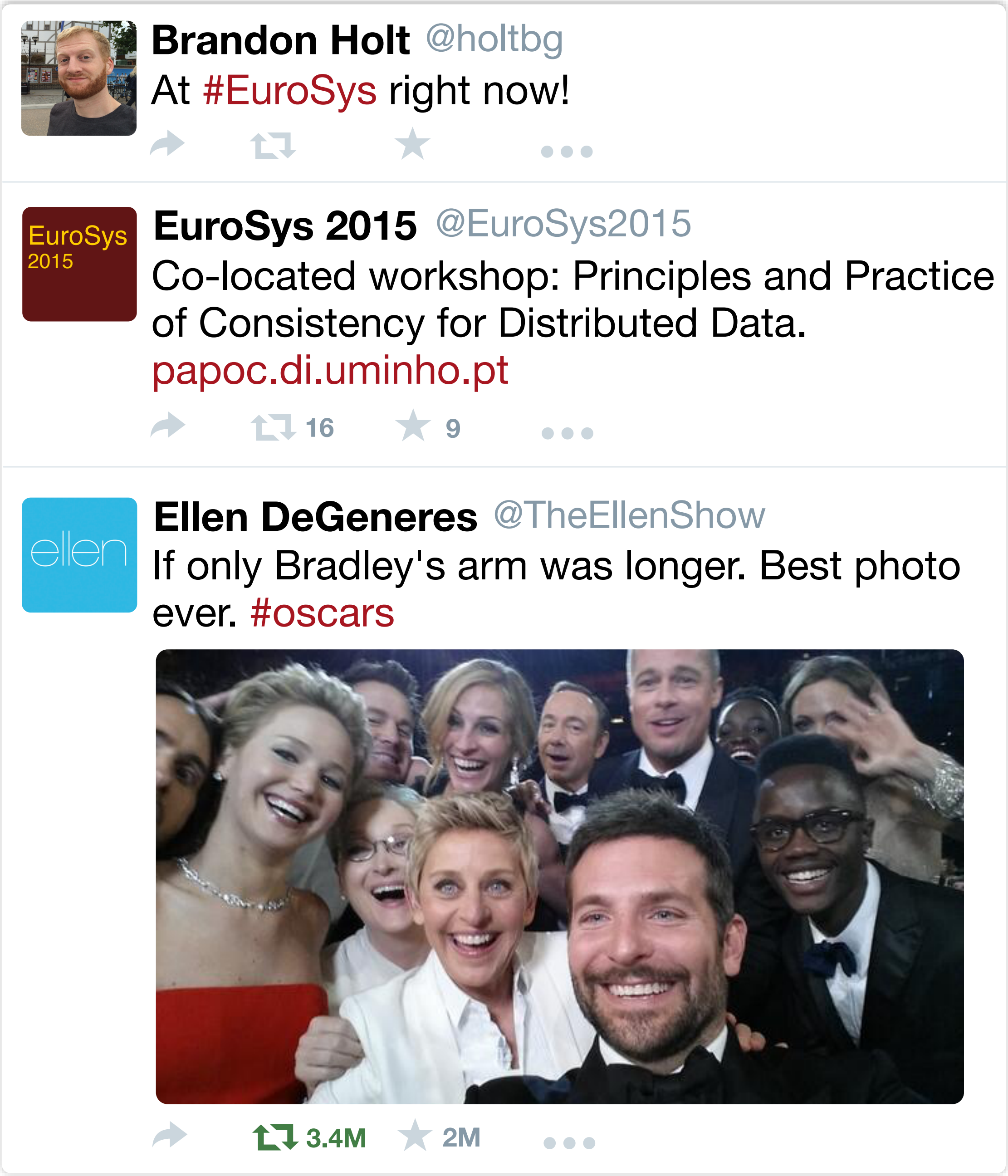
- Transactional boosting
- Combining

Approximate data types

- Bounded inconsistency
- Isolated eventual consistency (CRDTs)
- Probabilistic data types

Evaluation:

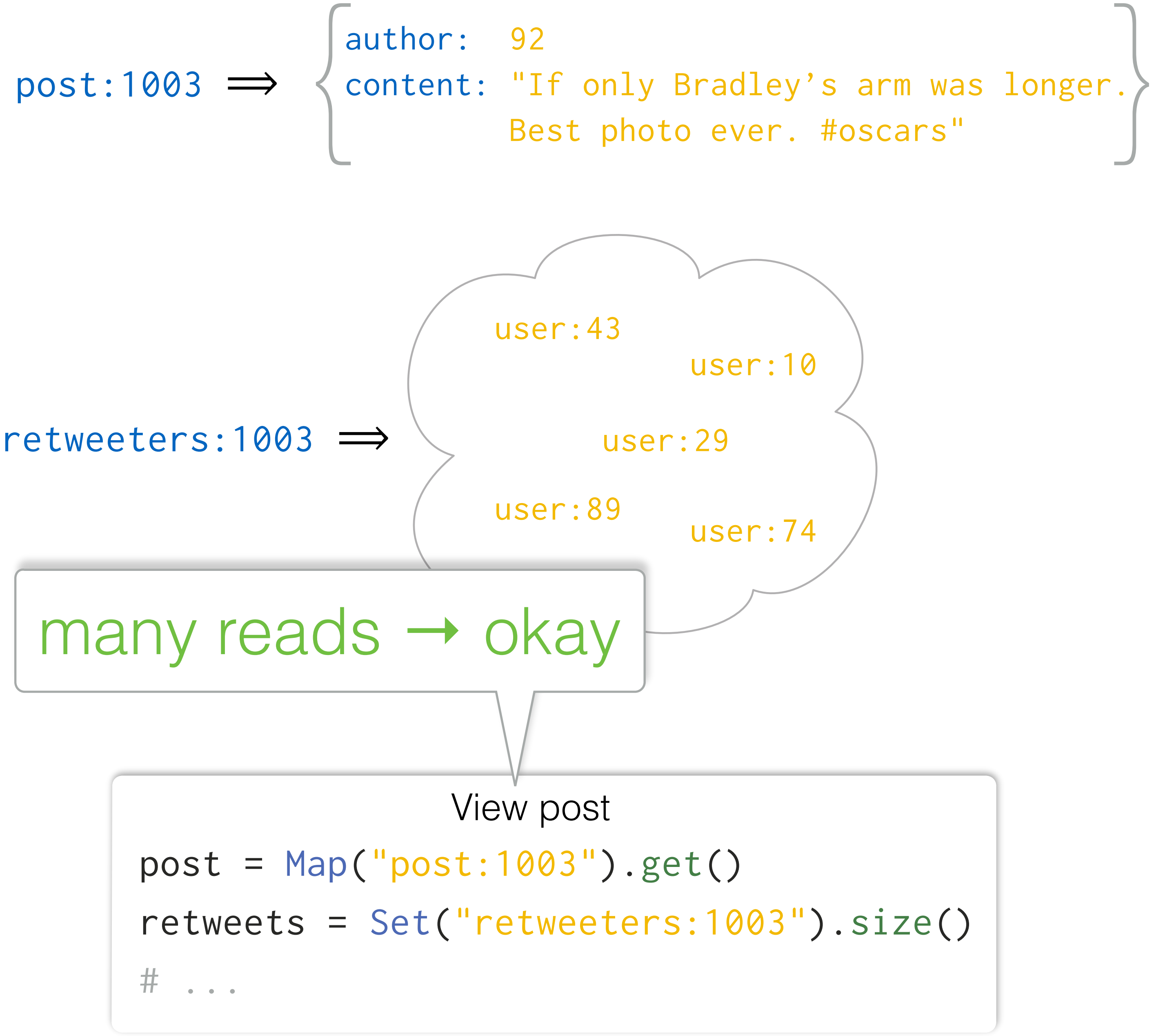
Commutativity



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data.
papoc.di.uminho.pt
16 ★ 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M ★ 2M



```
post:1003 ⟹ { author:  92
              content: "If only Bradley's arm was longer.
                        Best photo ever. #oscars" }
```

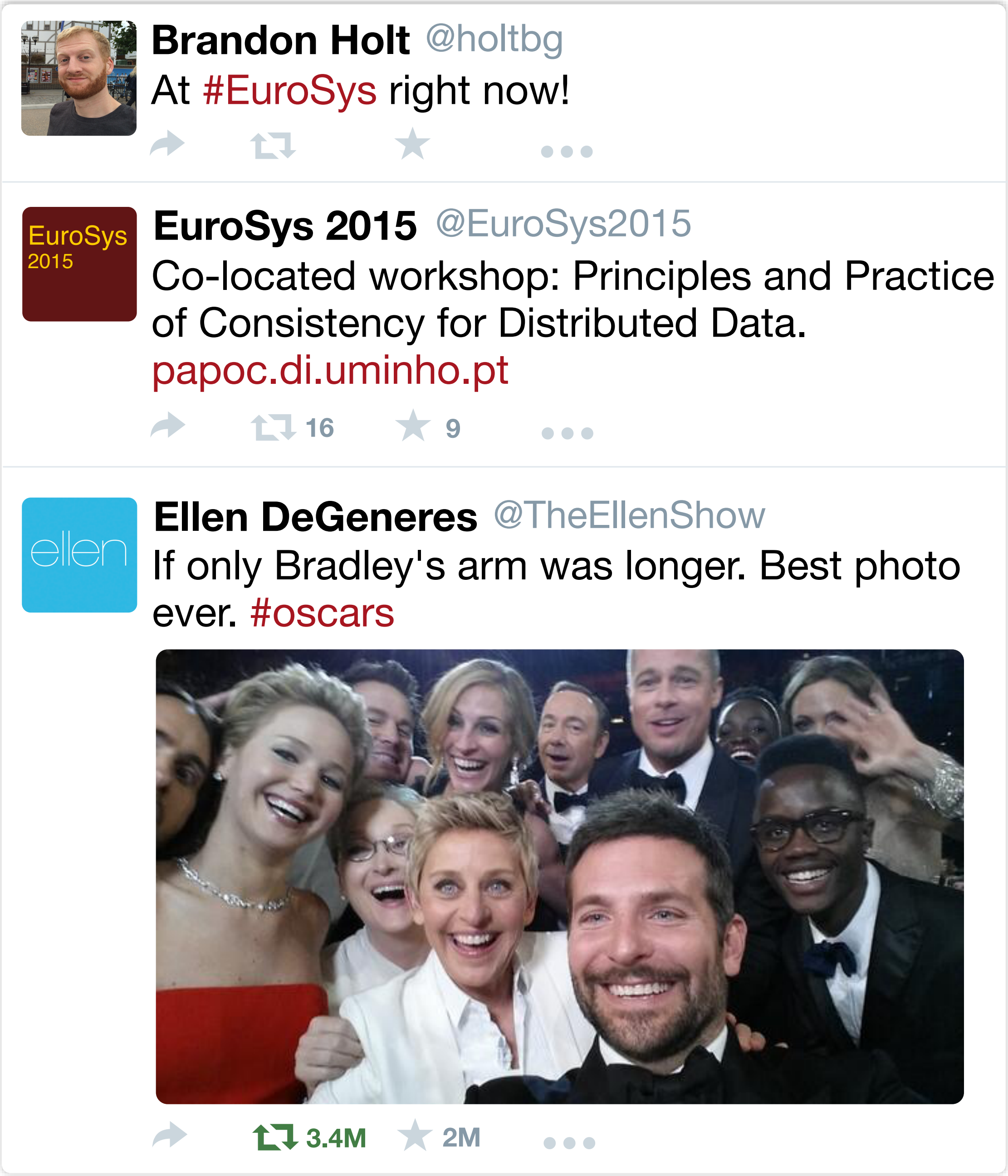
```
retweeters:1003 ⟹ user:43  user:10  user:29  user:89  user:74
```

many reads → okay

View post

```
post = Map("post:1003").get()
retweets = Set("retweeters:1003").size()
# ...
```

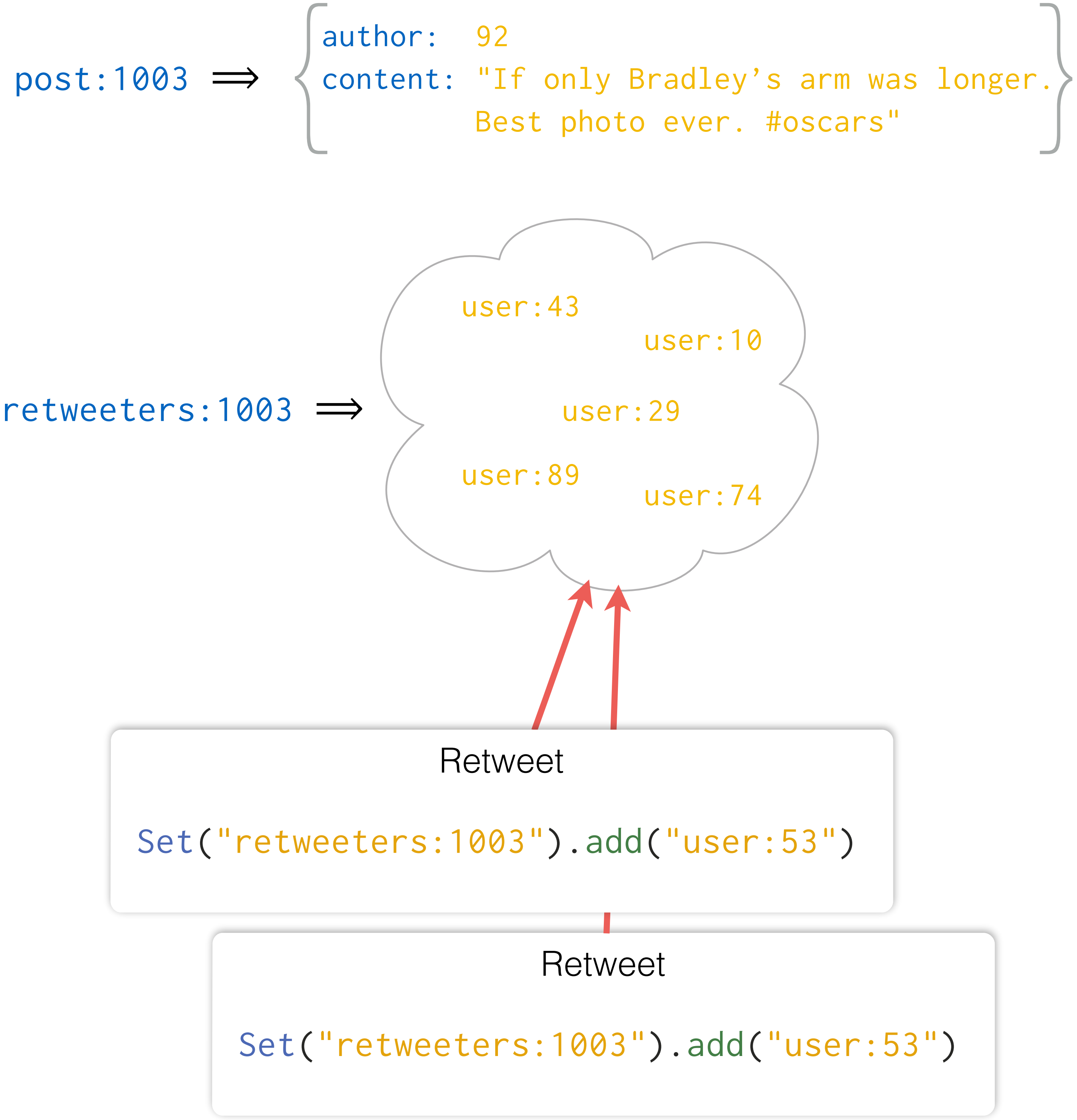
Commutativity



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M



```
post:1003 ⟹ { author:  92
              content: "If only Bradley's arm was longer.
                        Best photo ever. #oscars" }
```

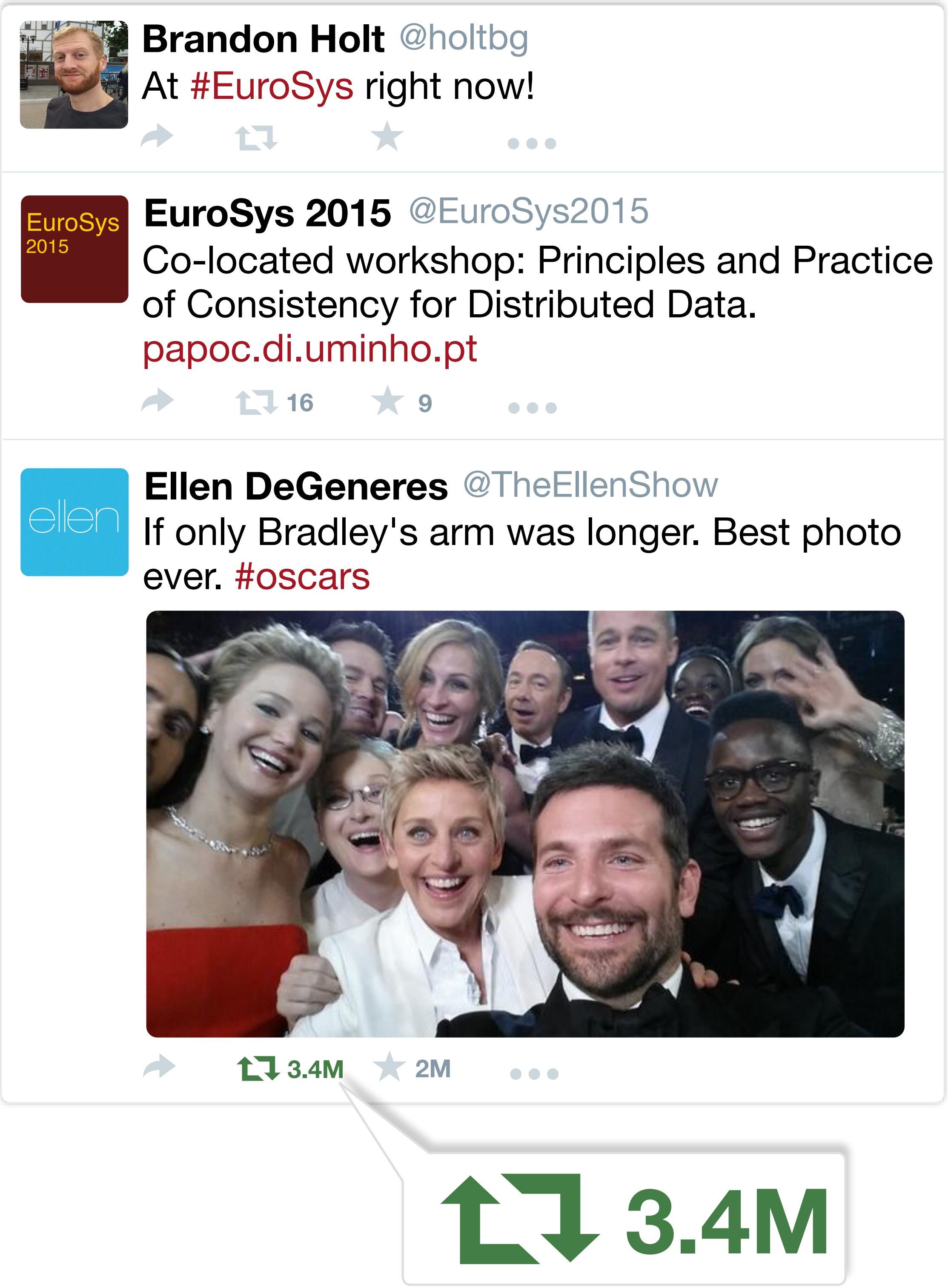
Retweet

```
Set("retweeters:1003").add("user:53")
```

Retweet

```
Set("retweeters:1003").add("user:53")
```

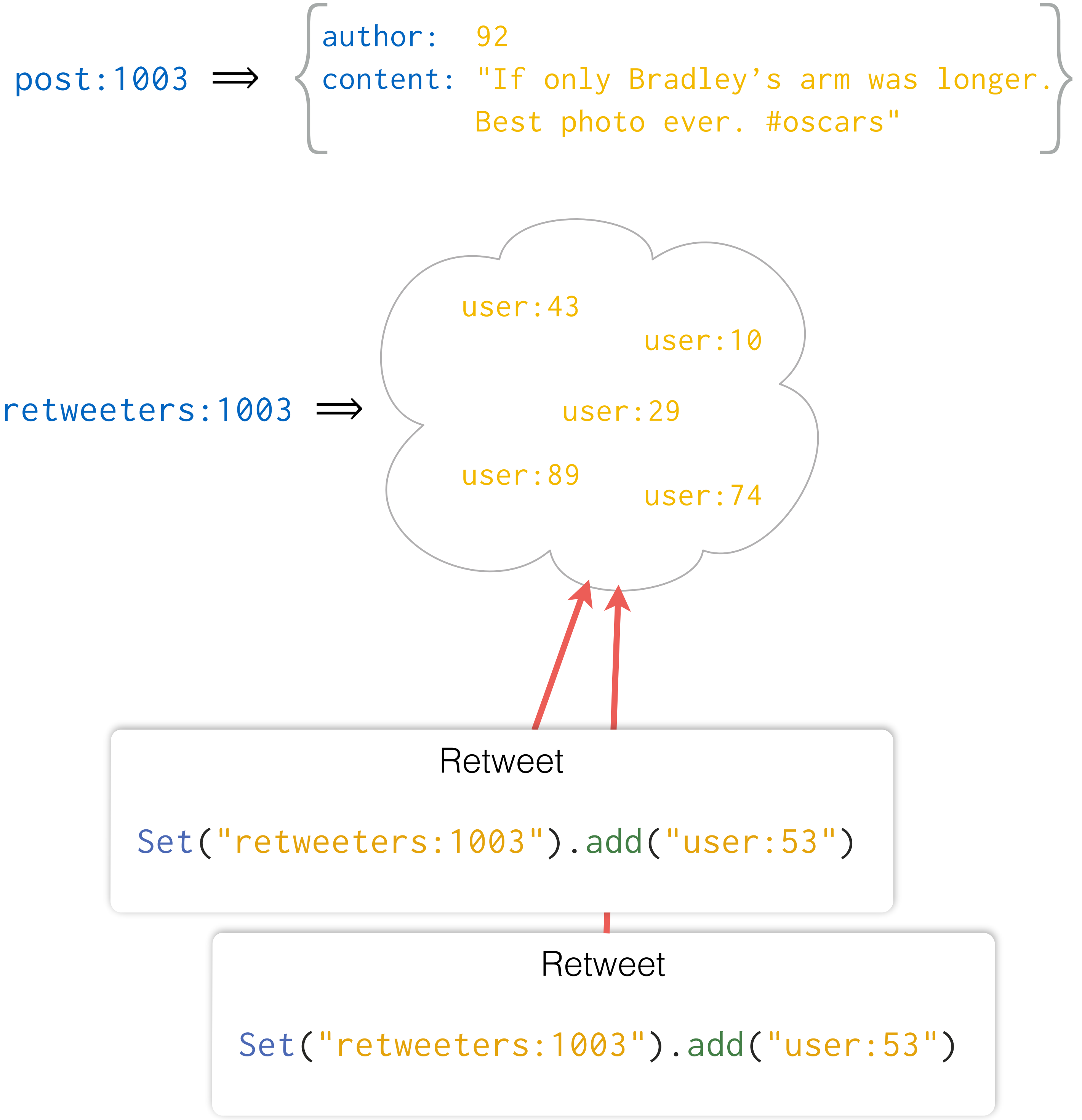
Commutativity



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M



```
post:1003 ⟹ { author:  92
              content: "If only Bradley’s arm was longer.
                        Best photo ever. #oscars" }

retweeters:1003 ⟹ user:43 user:10 user:29 user:89 user:74
```

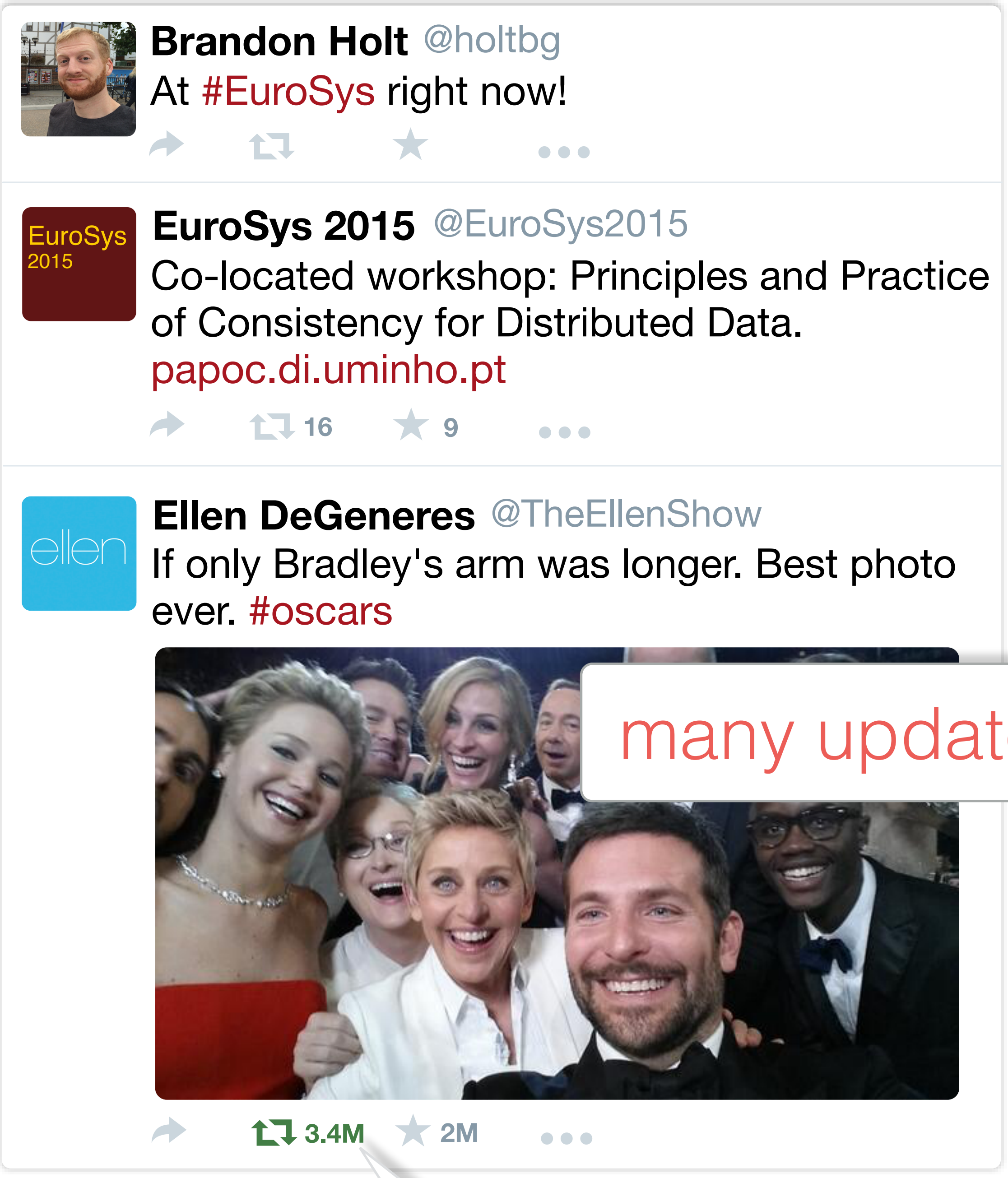
Retweet

```
Set("retweeters:1003").add("user:53")
```

Retweet

```
Set("retweeters:1003").add("user:53")
```

Commutativity



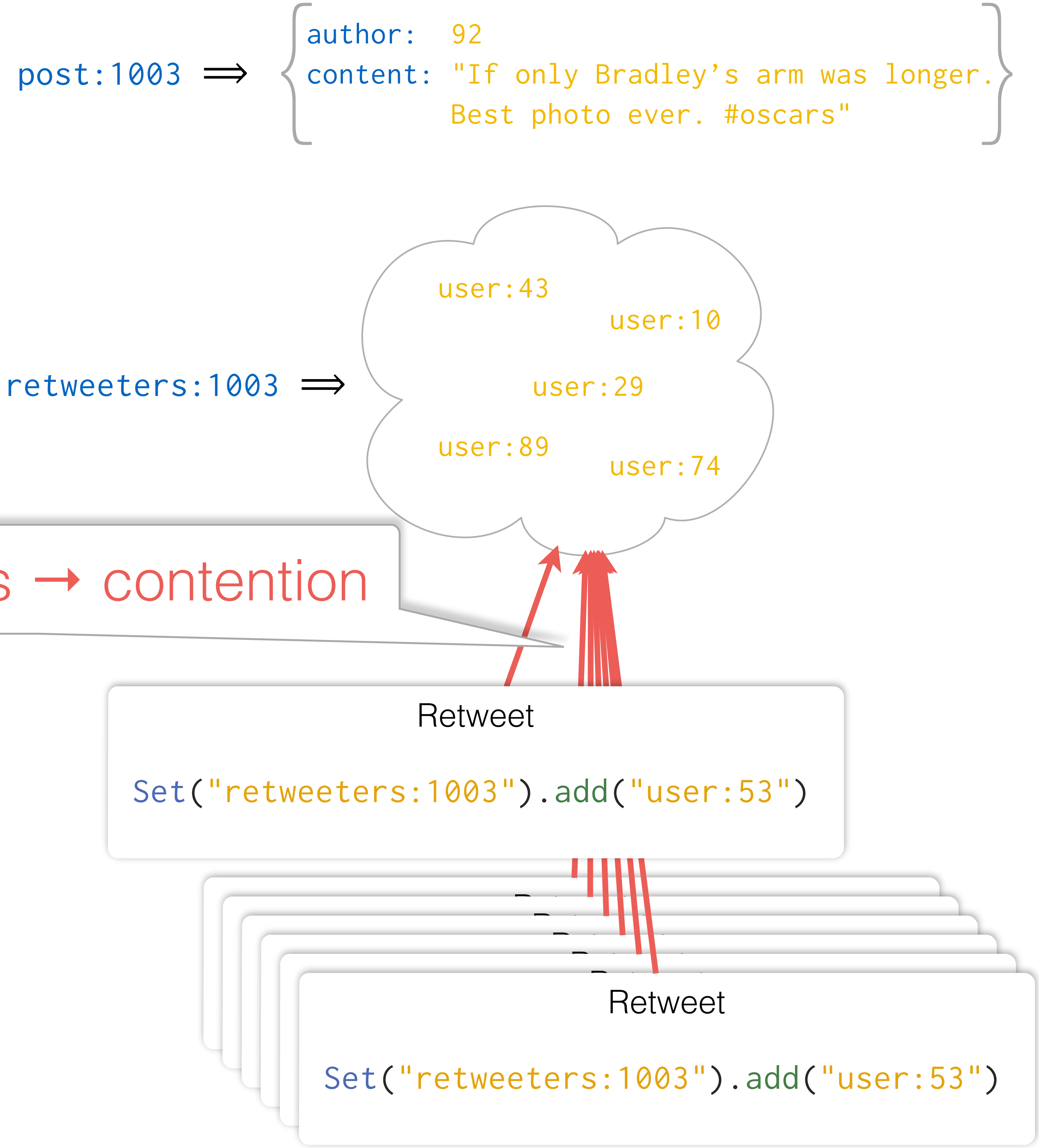
Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

3.4M

many updates → contention



```
post:1003 ⟹ { author:  92
              content: "If only Bradley's arm was longer.
                        Best photo ever. #oscars" }
```

```
retweeters:1003 ⟹ user:43 user:10 user:29 user:89 user:74
```

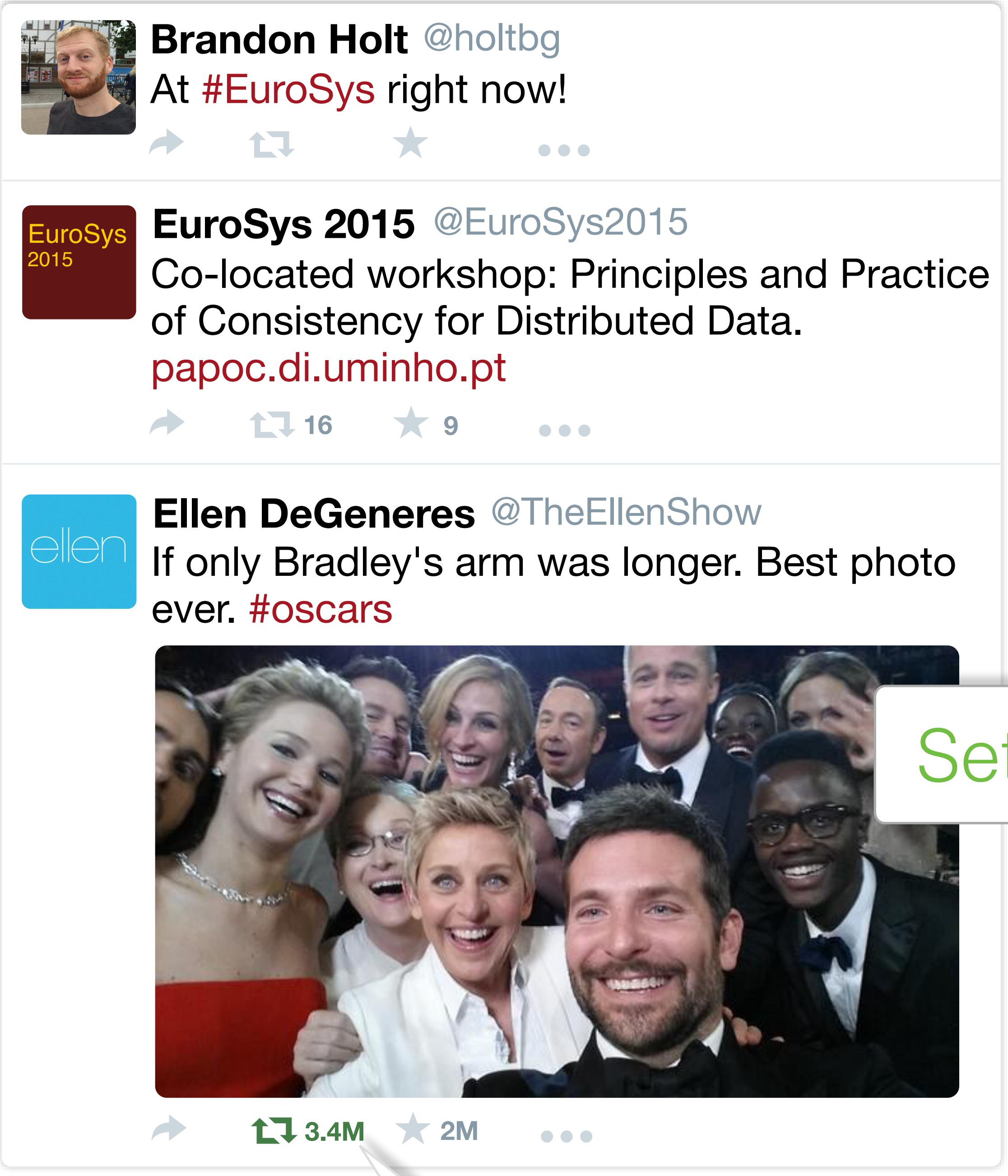
Retweet

```
Set("retweeters:1003").add("user:53")
```

Retweet

```
Set("retweeters:1003").add("user:53")
```

Commutativity



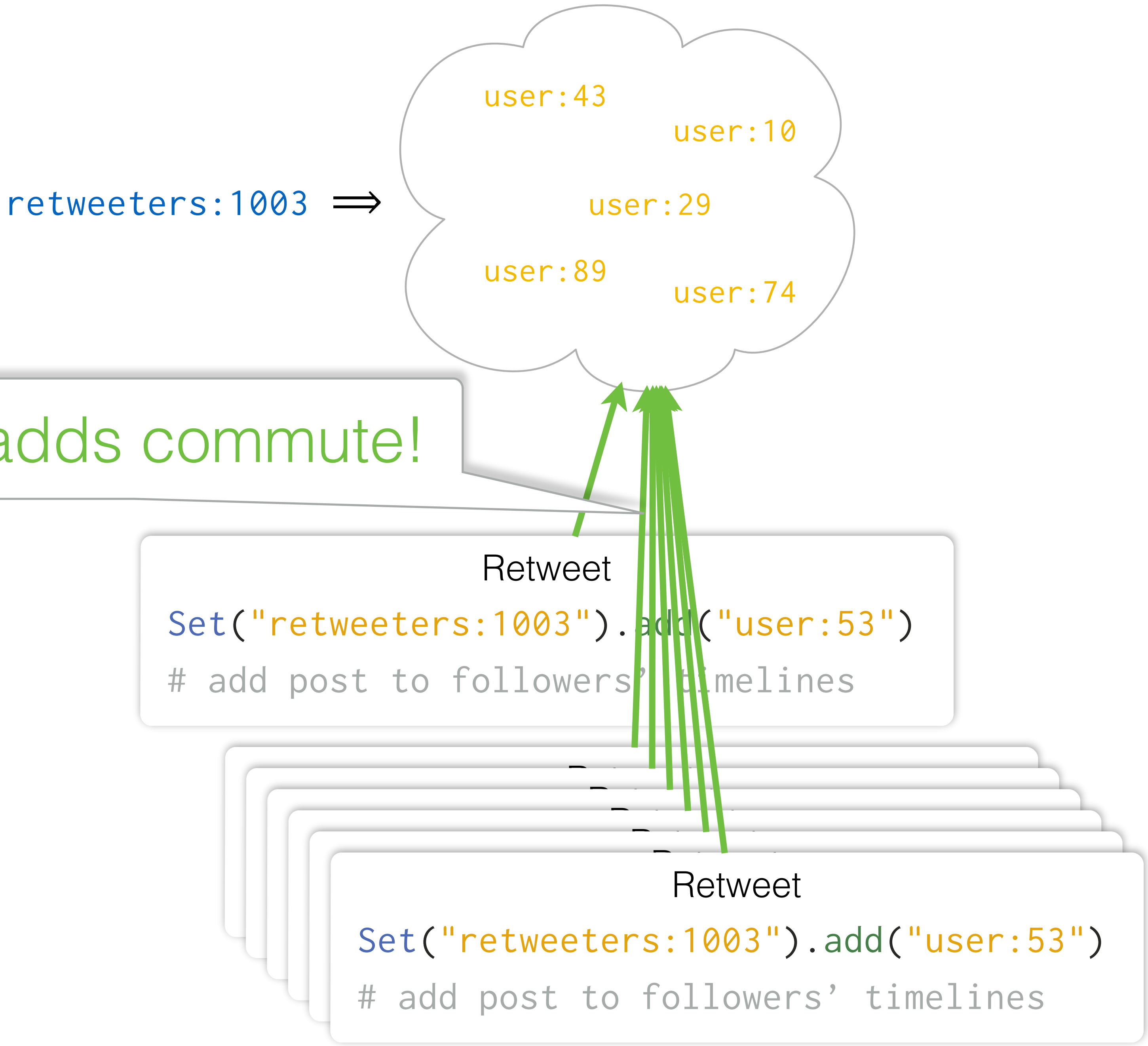
Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

3.4M

```
post:1003 ⟹ { author:  92
              content: "If only Bradley’s arm was longer.
                       Best photo ever. #oscars" }
```



Set adds commute!

Retweet

```
Set("retweeters:1003").add("user:53")
# add post to followers’ timelines
```

Retweet

```
Set("retweeters:1003").add("user:53")
# add post to followers’ timelines
```

Commutativity

For a given data type: which pairs of operations commute?

Commutativity Specification* for Set

method:	commutes with:	when:
`add(x): void`	`add(y)`	$\forall x,y$
`remove(x): void`	`remove(y)`	$\forall x,y$
	`add(y)`	$x \neq y$
`size(): int`	`add(x)`	$x \in Set$
	`remove(x)`	$x \notin Set$
`contains(x): bool`	`add(y)`	$x \neq y \vee y \in Set$
	`remove(y)`	$x \neq y \vee y \notin Set$
	`size()`	$\forall x$

* M. Kulkarni, D. Nguyen, D. Prountzos, X. Sui, and K. Pingali. Exploiting the Commutativity Lattice. *PLDI '11.*

Commutativity

For a given data type: which pairs of operations commute?

Commutativity Specification* for Set

method:	commutes with:	when:
`add(x): void`	`add(y)`	$\forall x,y$
`remove(x): void`	`remove(y)`	$\forall x,y$
	`add(y)`	$x \neq y$
`size(): int`	`add(x)`	$x \in Set$
	`remove(x)`	$x \notin Set$
`contains(x): bool`	`add(y)`	$x \neq y \vee y \in Set$
	`remove(y)`	$x \neq y \vee y \notin Set$
	`size()`	$\forall x$

If the key/value store knew this, what could it do?

* M. Kulkarni, D. Nguyen, D. Prountzos, X. Sui, and K. Pingali. Exploiting the Commutativity Lattice. *PLDI '11.*

Commutativity

Problem: contention → many aborts / retries

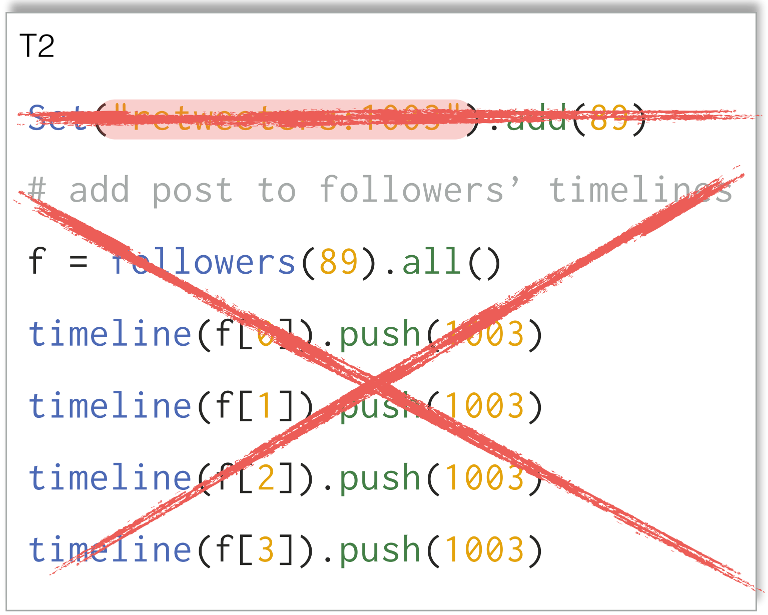
```
T1
Set("retweeters:1003").add(53)
# add post to followers’ timelines
f = followers(53).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

```
T2
Set("retweeters:1003").add(89)
# add post to followers’ timelines
f = followers(89).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

Commutativity

Problem: contention → many aborts / retries

```
T1
Set("retweeters:1003").add(53)
# add post to followers’ timelines
f = followers(53).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```



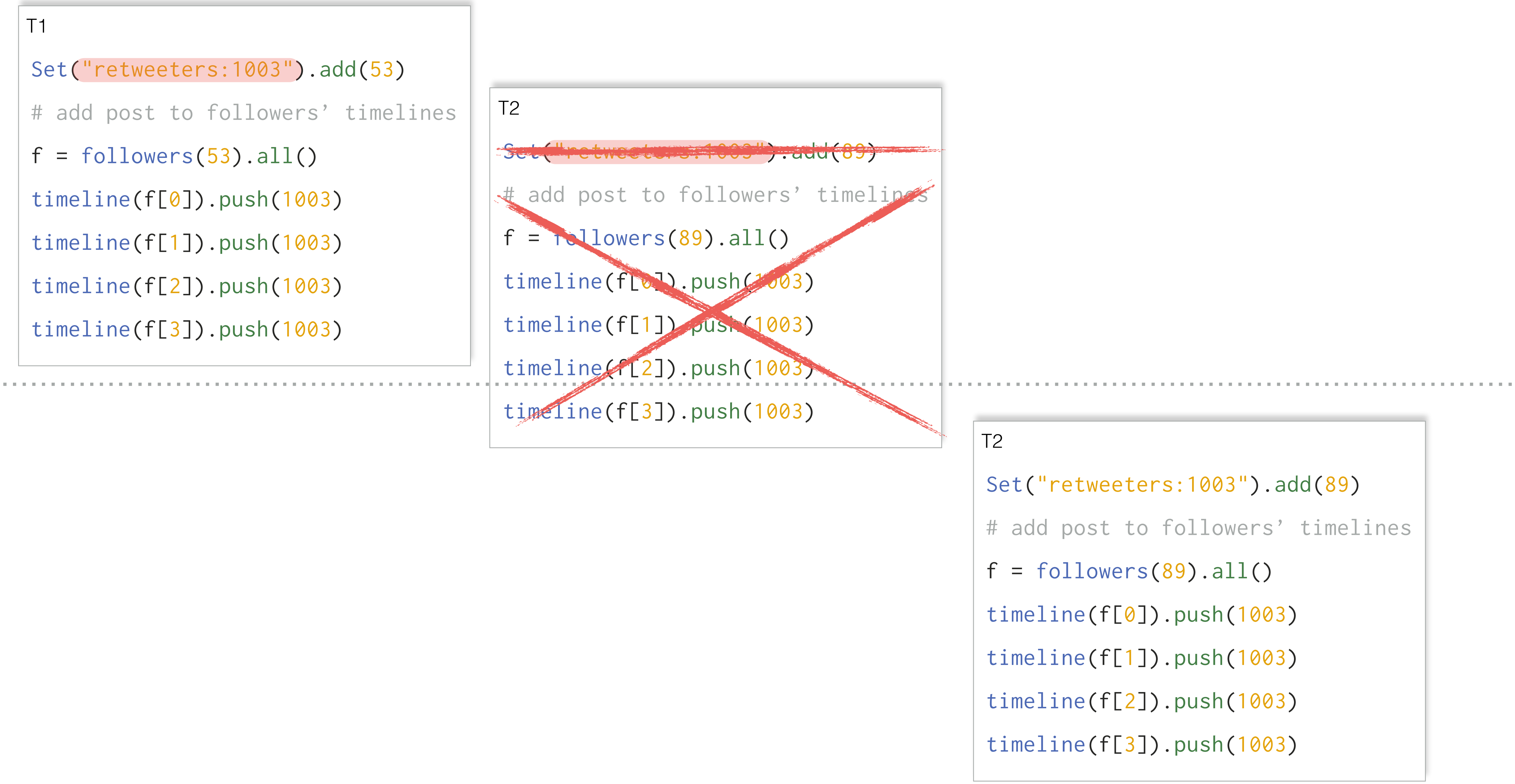
T2

~~`Set("retweeters:1003").add(89)`~~

```
# add post to followers’ timelines
f = followers(89).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

Commutativity

Problem: contention → many aborts / retries



T1

```
Set("retweeters:1003").add(53)
# add post to followers' timelines
f = followers(53).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

T2

~~Set("retweeters:1003").add(89)~~
~~# add post to followers' timelines~~
~~f = followers(89).all()~~
~~timeline(f[0]).push(1003)~~
~~timeline(f[1]).push(1003)~~
~~timeline(f[2]).push(1003)~~
~~timeline(f[3]).push(1003)~~

T2

```
Set("retweeters:1003").add(89)
# add post to followers' timelines
f = followers(89).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

Commutativity

Problem:

Solution: *Transactional boosting*[*]

- when operations commute, no need to abort their transactions

```
T1
Set("retweeters:1003").add(53)
# add post to followers' timelines
f = followers(53).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

```
T2
Set("retweeters:1003").add(89)
# add post to followers' timelines
f = followers(89).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

[*] M. Herlihy and E. Koskinen.
Transactional Boosting: A Methodology for Highly-concurrent Transactional Objects. PPoPP 2008.

Commutativity

Problem:

Solution: *Transactional boosting*[*]

- when operations commute, no need to abort their transactions
- reduce abort rate → increase throughput

```
T1
Set("retweeters:1003").add(53)
# add post to followers’ timelines
f = followers(53).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

```
T2
Set("retweeters:1003").add(89)
# add post to followers’ timelines
f = followers(89).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

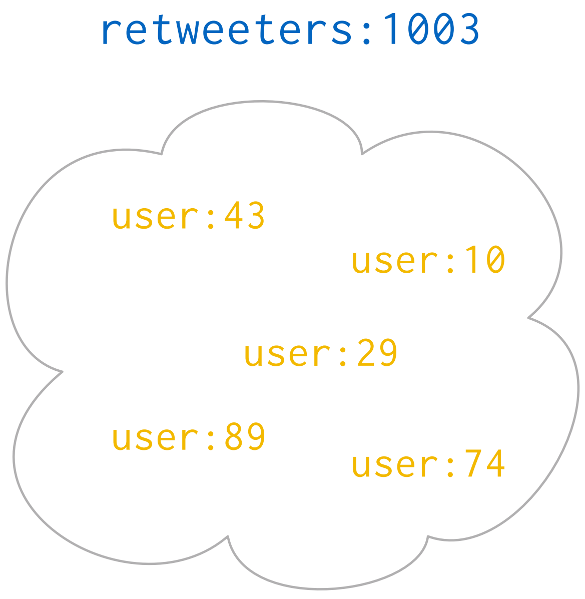
```
T3
Set("retweeters:1003").add(71)
# add post to followers’ timelines
f = followers(71).all()
timeline(f[0]).push(1003)
timeline(f[1]).push(1003)
timeline(f[2]).push(1003)
timeline(f[3]).push(1003)
```

[*] M. Herlihy and E. Koskinen.
Transactional Boosting: A Methodology for Highly-concurrent Transactional Objects. PPoPP 2008.

Commutativity

Problem: Serializing operations on **hot** records

```
Set("retweeters:1003").add(53)
Set("retweeters:1003").add(89)
Set("retweeters:1003").add(71)
Set("retweeters:1003").add(22)
Set("retweeters:1003").add(11)
Set("retweeters:1003").add(55)
Set("retweeters:1003").add(42)
Set("retweeters:1003").add(91)
Set("retweeters:1003").add(96)
```



retweeters:1003

user:43

user:10

user:29

user:89

user:74

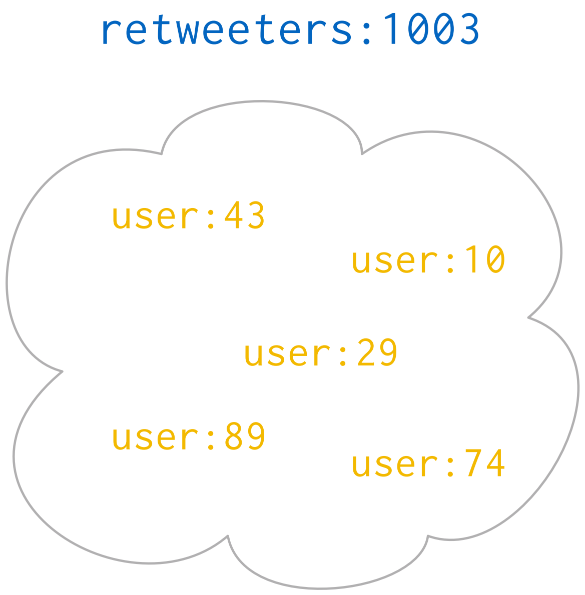
Commutativity

Problem:

```
Set("retweeters:1003").add(53)
Set("retweeters:1003").add(89)
Set("retweeters:1003").add(71)
```

```
Set("retweeters:1003").add(22)
Set("retweeters:1003").add(11)
Set("retweeters:1003").add(55)
```

```
Set("retweeters:1003").add(42)
Set("retweeters:1003").add(91)
Set("retweeters:1003").add(96)
```



retweeters:1003

user:43

user:10

user:29

user:89

user:74

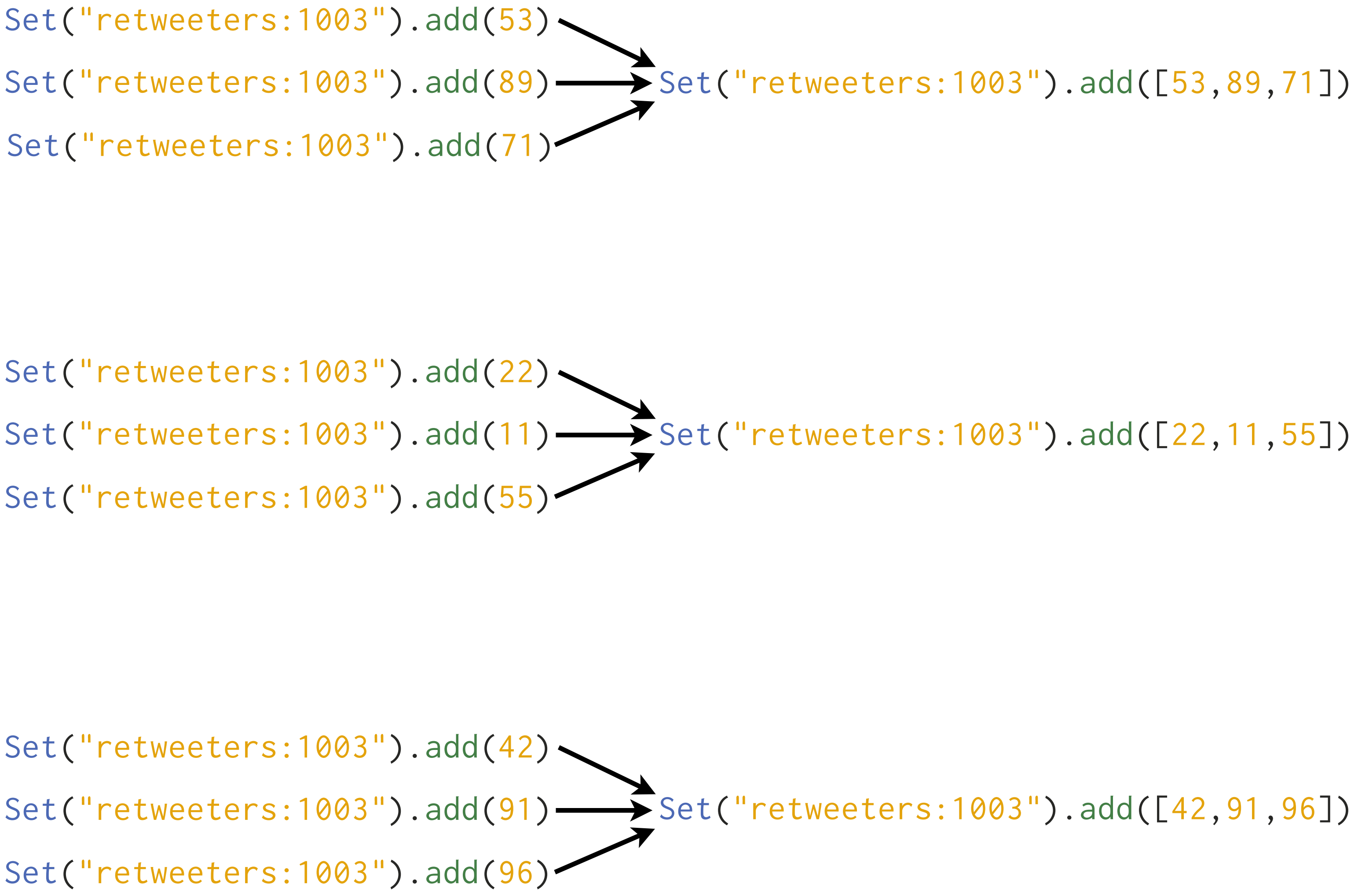
* D. Hendler, I. Incze, N. Shavit, and M. Tzafrir. Flat combining and the synchronization-parallelism tradeoff. ACM Symposium on Parallelism in Algorithms and Architectures, 2010.

Commutativity

Problem:

Solution: *Combining*[*]

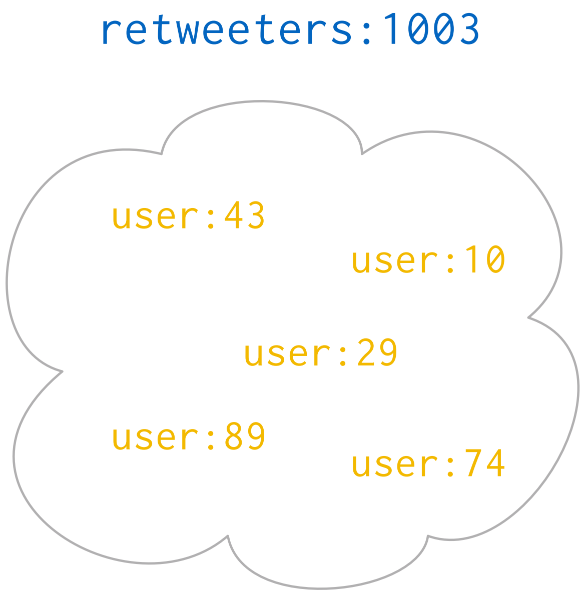
- merge multiple operations together and apply them all at once



```
Set("retweeters:1003").add(53)
Set("retweeters:1003").add(89)  →  Set("retweeters:1003").add([53,89,71])
Set("retweeters:1003").add(71)

Set("retweeters:1003").add(22)
Set("retweeters:1003").add(11)  →  Set("retweeters:1003").add([22,11,55])
Set("retweeters:1003").add(55)

Set("retweeters:1003").add(42)
Set("retweeters:1003").add(91)  →  Set("retweeters:1003").add([42,91,96])
Set("retweeters:1003").add(96)
```



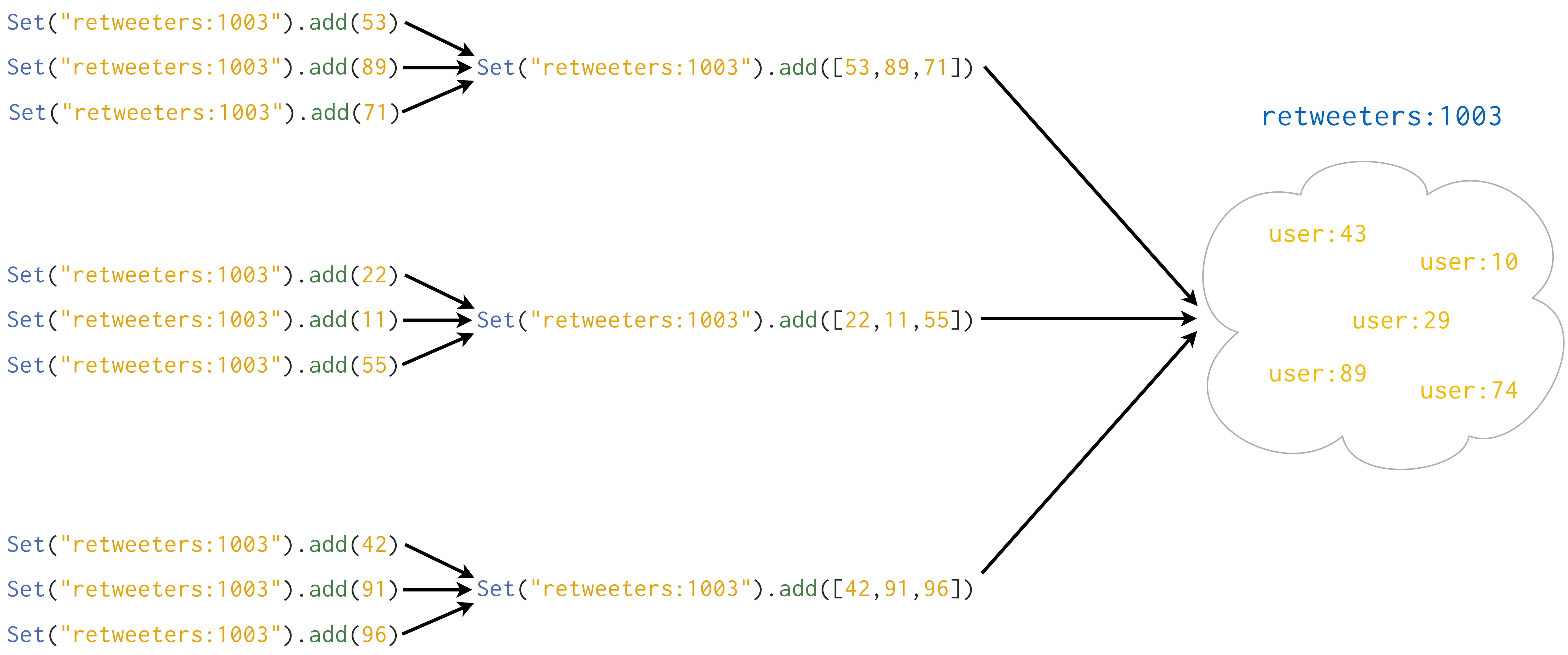
[*] D. Hendler, I. Incze, N. Shavit, and M. Tzafrir. Flat combining and the synchronization-parallelism tradeoff. ACM Symposium on Parallelism in Algorithms and Architectures, 2010.

Commutativity

Problem:

Solution: *Combining*[*]

- merge multiple operations together and apply them all at once
- parallelize and decrease contention



```
Set("retweeters:1003").add(53)
Set("retweeters:1003").add(89)
Set("retweeters:1003").add(71)
Set("retweeters:1003").add([53,89,71])

Set("retweeters:1003").add(22)
Set("retweeters:1003").add(11)
Set("retweeters:1003").add(55)
Set("retweeters:1003").add([22,11,55])

Set("retweeters:1003").add(42)
Set("retweeters:1003").add(91)
Set("retweeters:1003").add(96)
Set("retweeters:1003").add([42,91,96])
```

[*] D. Hendler, I. Incze, N. Shavit, and M. Tzafrir. Flat combining and the synchronization-parallelism tradeoff. ACM Symposium on Parallelism in Algorithms and Architectures, 2010.

Leveraging **Abstract Data Types** in **NoSQL**

Commutativity

- Transactional boosting
- Combining

Approximate data types

- Bounded inconsistency
- Isolated eventual consistency (CRDTs)
- Probabilistic data types

Evaluation: *Claret* prototype

Approximate data types

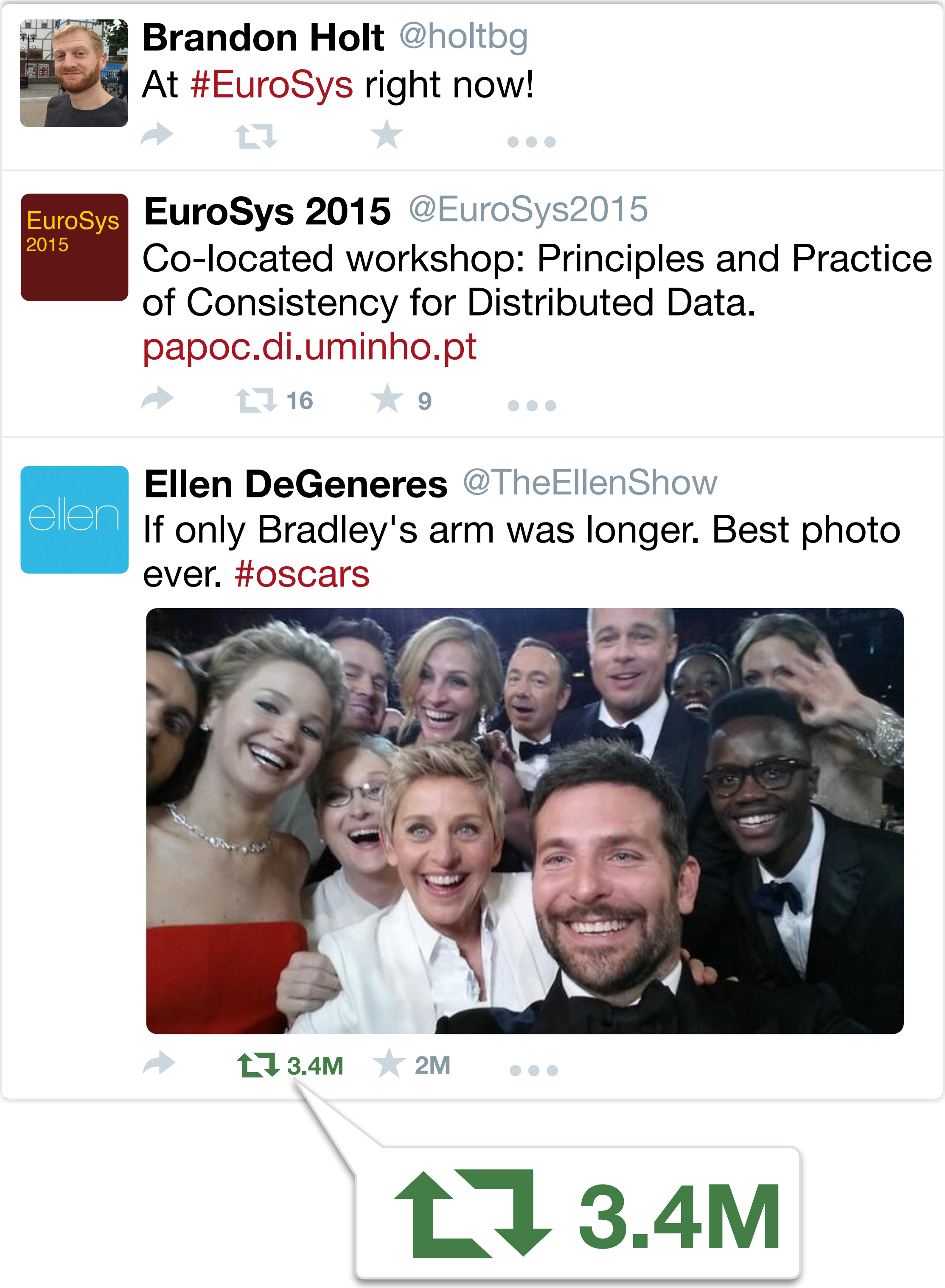
Problem: Reads don’t commute with **updates**

Retweet

```
Set("retweeters:1003").add("user:53")
# ...
```

View post

```
# ...
retweets = Set("retweeters:1003").size()
# ...
```



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

3.4M

Approximate data types

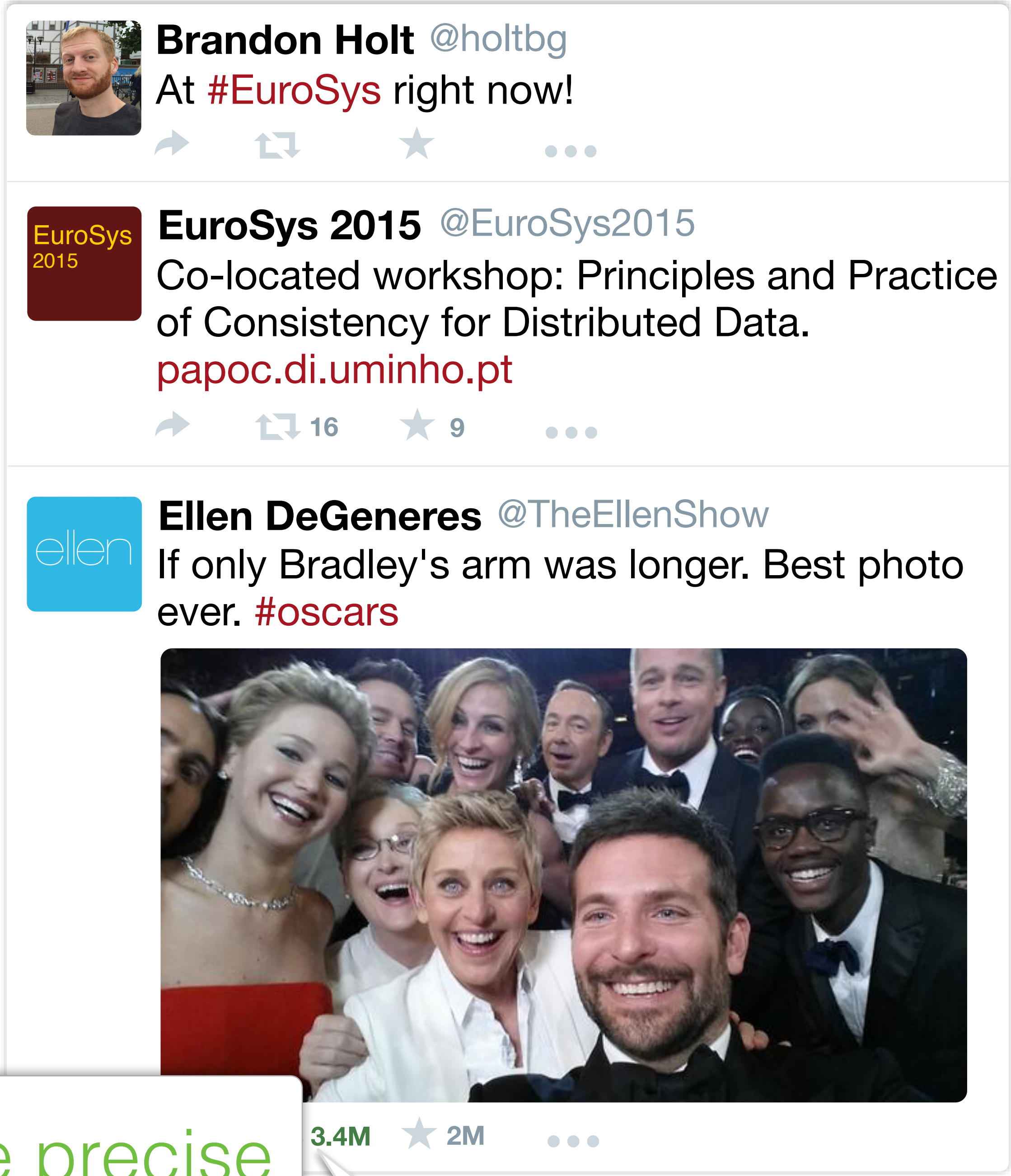
Problem: Reads don’t commute with **updates**

Retweet

```
Set("retweeters:1003").add("user:53")
# ...
```

View post

```
# ...
retweets = Set("retweeters:1003").size()
# ...
```



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data.
papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

doesn’t need to be precise

3.4M

Approximate data types

Problem:

Solution: *Bounded inconsistency*

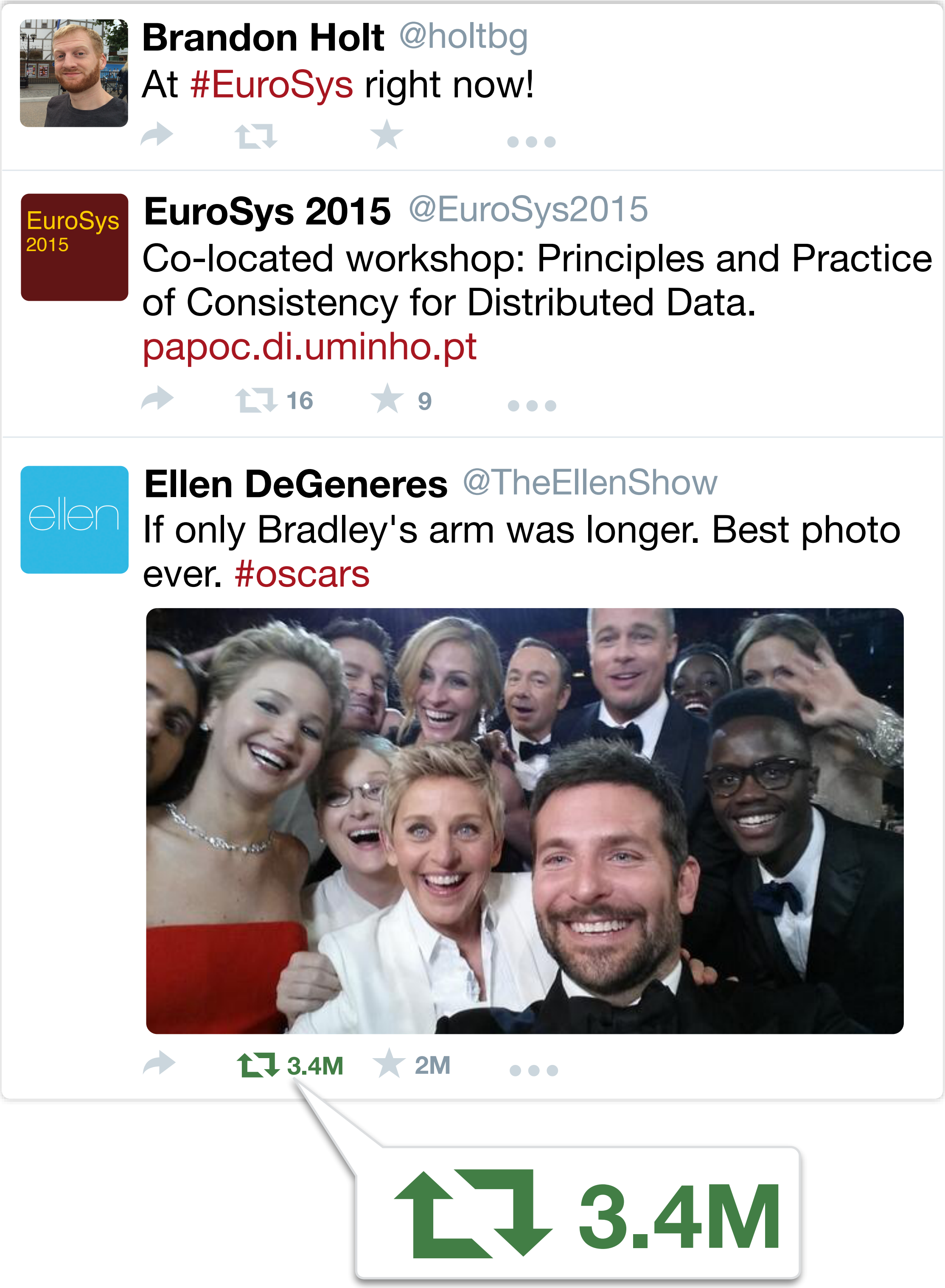
- allow *some* updates concurrently with reads
- exposes additional "commutativity"

Retweet

```
Set("retweeters:1003").add("user:53")
# ...
```

View post

```
# ...
retweets = Set("retweeters:1003").approxSize<0.05>()
# ...
```



Brandon Holt @holtbg
At #EuroSys right now!

EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data. papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

3.4M

Approximate data types

Problem:

Solution: *Bounded inconsistency*

- allow *some* updates concurrently with reads
- exposes additional "commutativity"

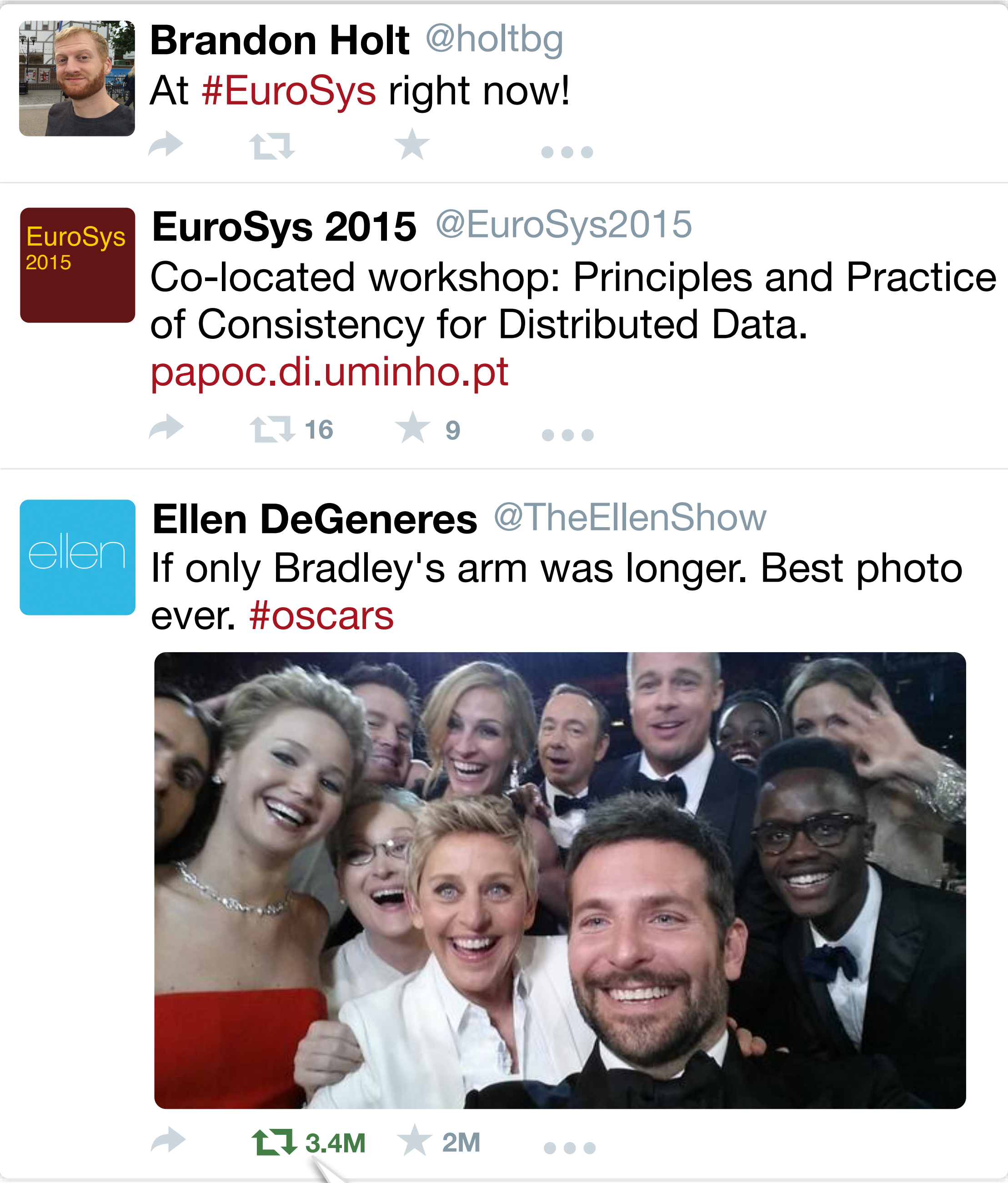
Retweet

```
Set("retweeters:1003").add("user:53")
# ...
```

View post

```
# ...
retweets = Set("retweeters:1003").approxSize<0.05>()
# ...
```

5% error → 170,000 adds



Brandon Holt @holtbg
At #EuroSys right now!

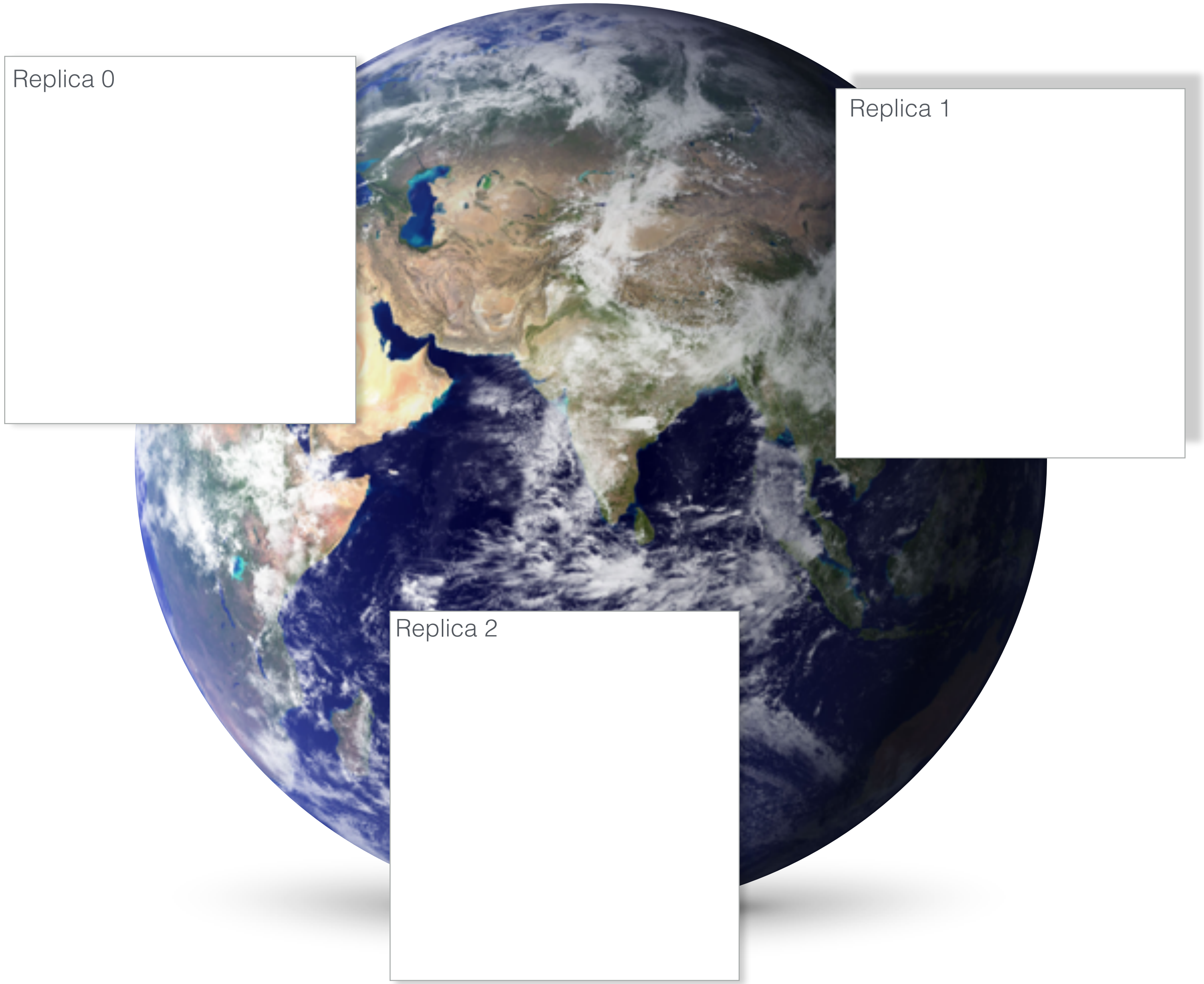
EuroSys 2015 @EuroSys2015
Co-located workshop: Principles and Practice of Consistency for Distributed Data.
papoc.di.uminho.pt
16 9

Ellen DeGeneres @TheEllenShow
If only Bradley's arm was longer. Best photo ever. #oscars
3.4M 2M

3.4M

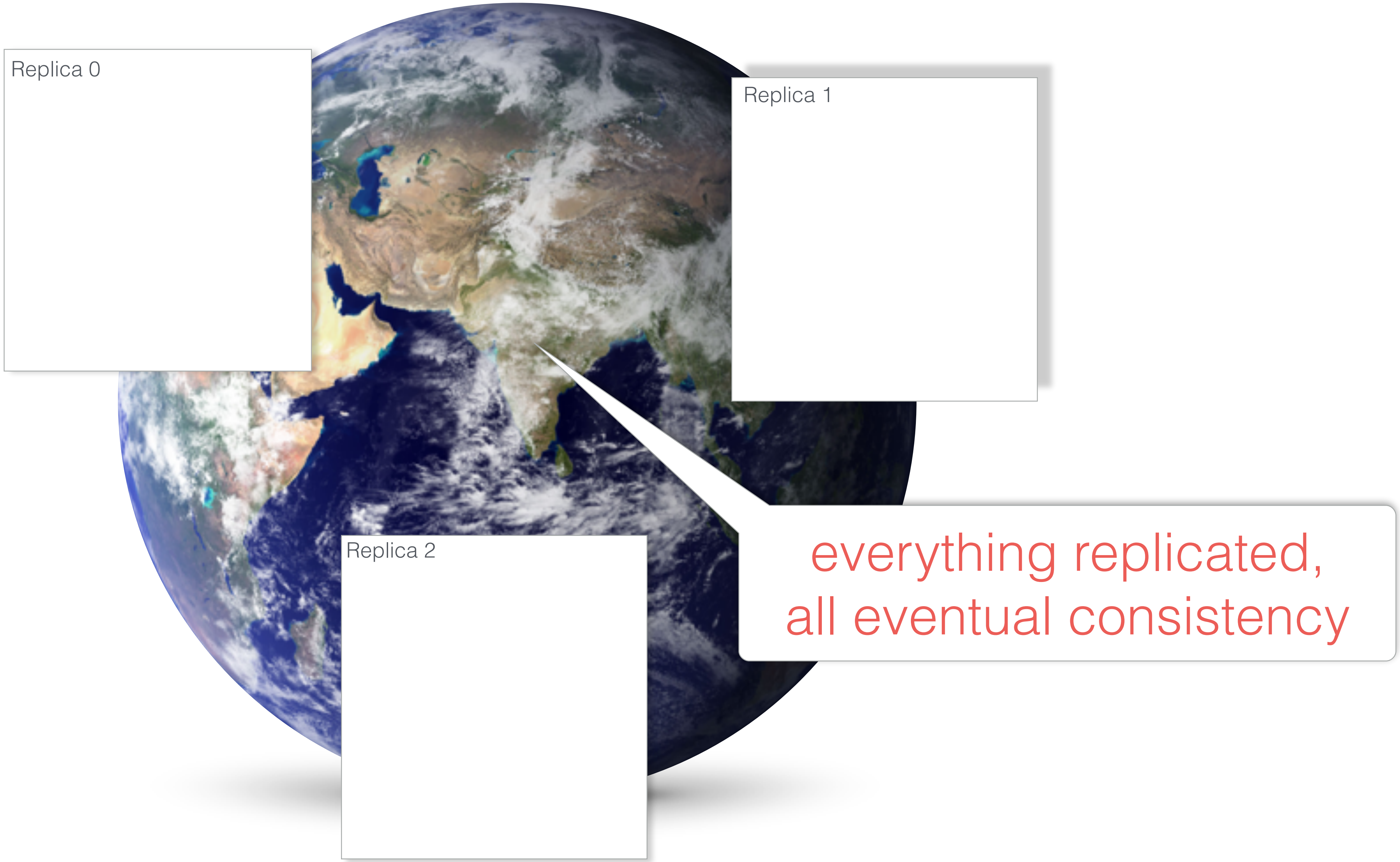
Approximate data types

Problem: Scaling → high latencies, low availability



Approximate data types

Problem: Scaling → high latencies, low availability



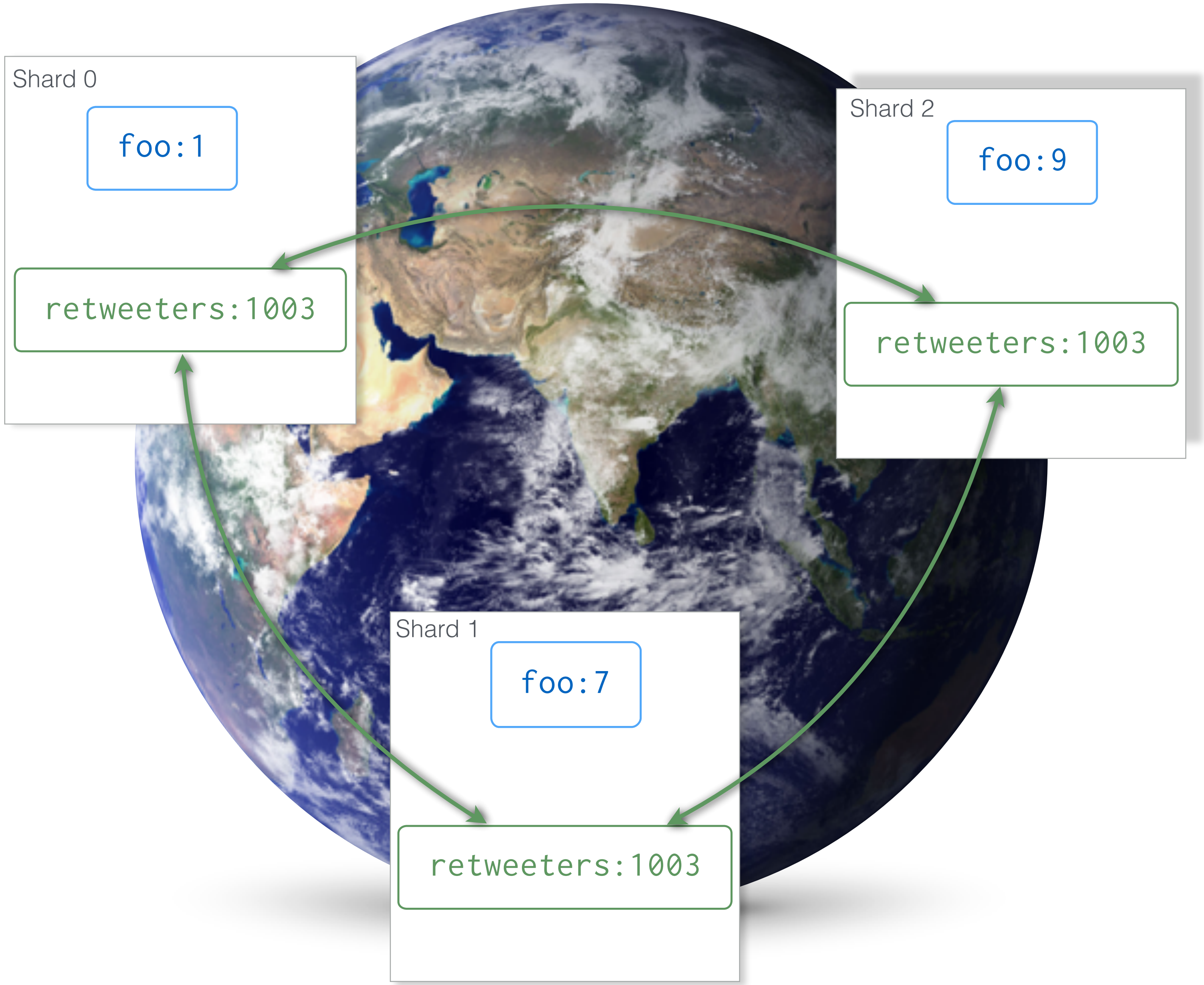
everything replicated,
all eventual consistency

Approximate data types

Problem:

Solution: *Isolated eventual consistency* via *CRDTs*

- use CRDT data type only where needed for scaling or low-latency
- programmers choose what can be approximate



Approximate data types

Problem: Can’t (or don’t want to) store all the data

Tweets per second

Support for #France and #England during their #Euro2012 match on June 11, 2012.

#ENGFRA

Approximate data types

Problem:

Tweets per second

Approximate data types

Problem:

Solution: *Probabilistic data types*

- e.g. HyperLogLog, Bloom filter, Count-min sketch, T-digest
- useful for tracking statistics, summary of high-volume data, or partially-materialized views

Tweets per second

Leveraging **Abstract Data Types** in **NoSQL**

Commutativity

- Transactional boosting
- Combining

Approximate data types

- Bounded inconsistency
- Isolated eventual consistency (CRDTs)
- Probabilistic data types

Evaluation: *Claret* prototype

- Transactional boosting
- Bounded inconsistency

Evaluation

Claret: Key-value store with data types

- simple two-phase commit protocol with locking (**+transactional boosting**)
- experiments run with 4 shards,
  standard local ethernet network,
  8-core 2GHz Intel Xeon processor per node

Evaluation

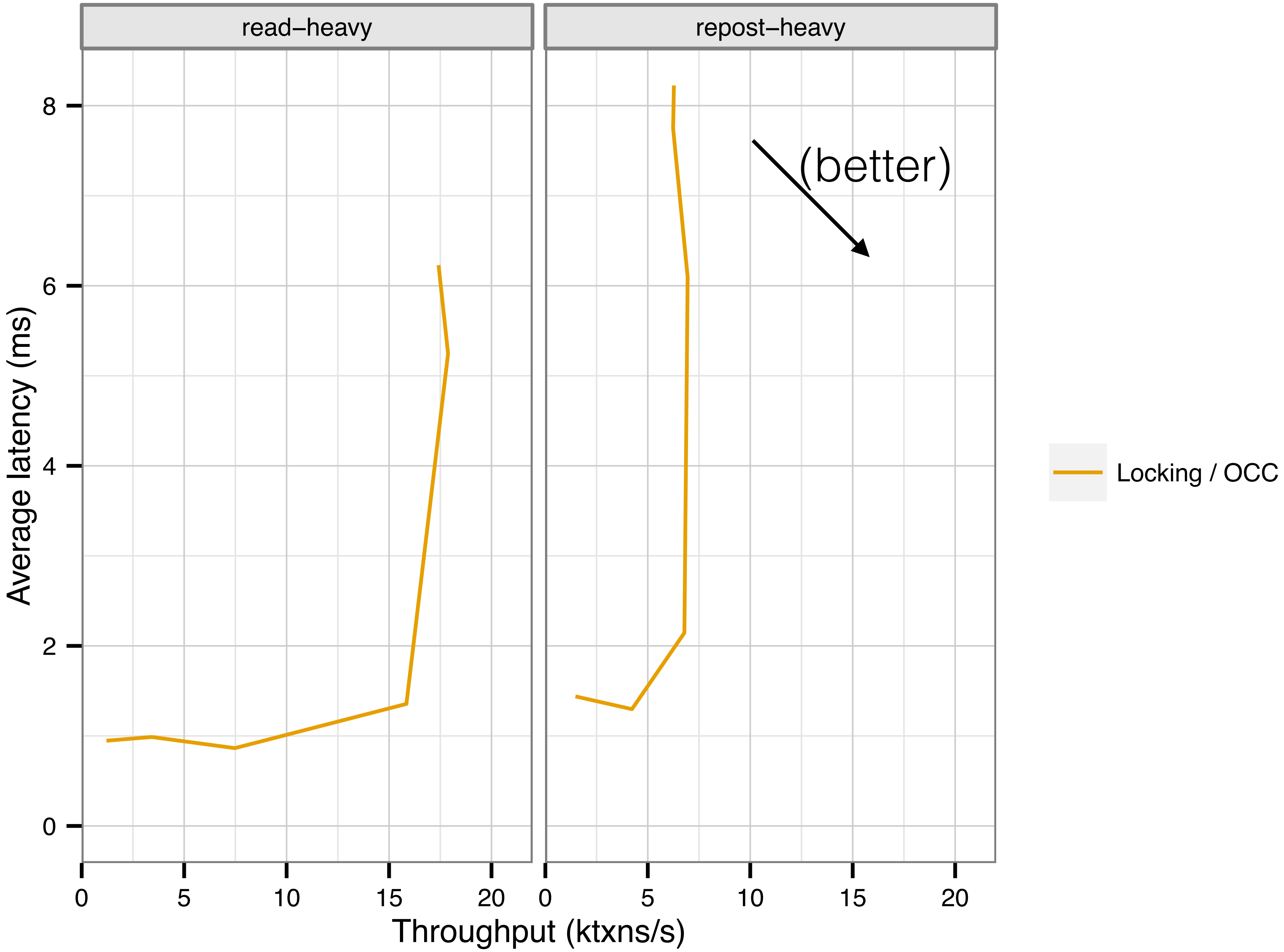
Case study: Twitter clone

- realistic synthetic graph (Kronecker, scale 14)
- simple random user model, retweet more popular posts (*viral* effect)

Evaluation

Case study: Twitter clone

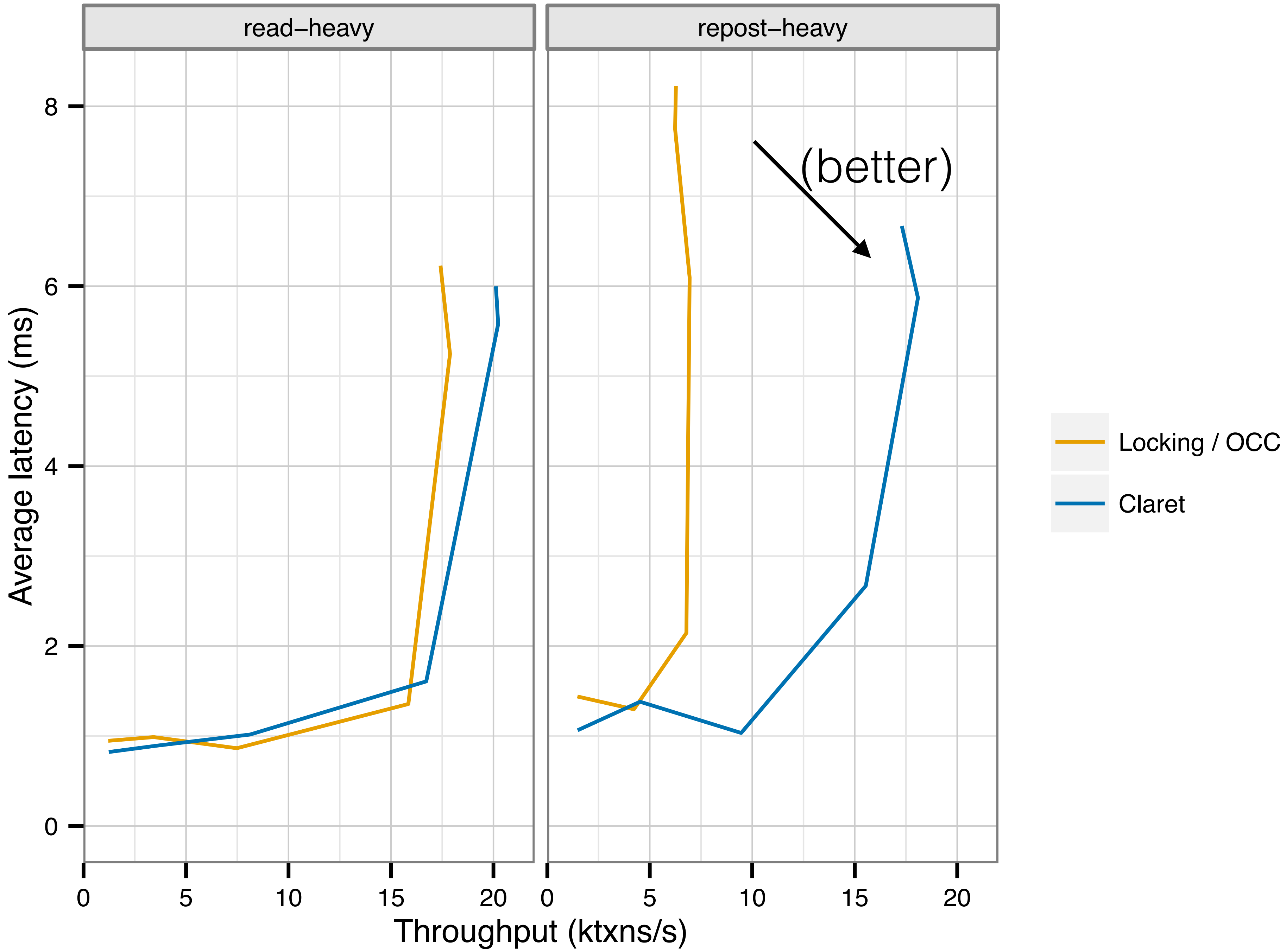
- realistic synthetic graph (Kronecker, scale 16)
- simple random user model, retweet more popular posts (*viral* effect)



Evaluation

Case study: Twitter clone

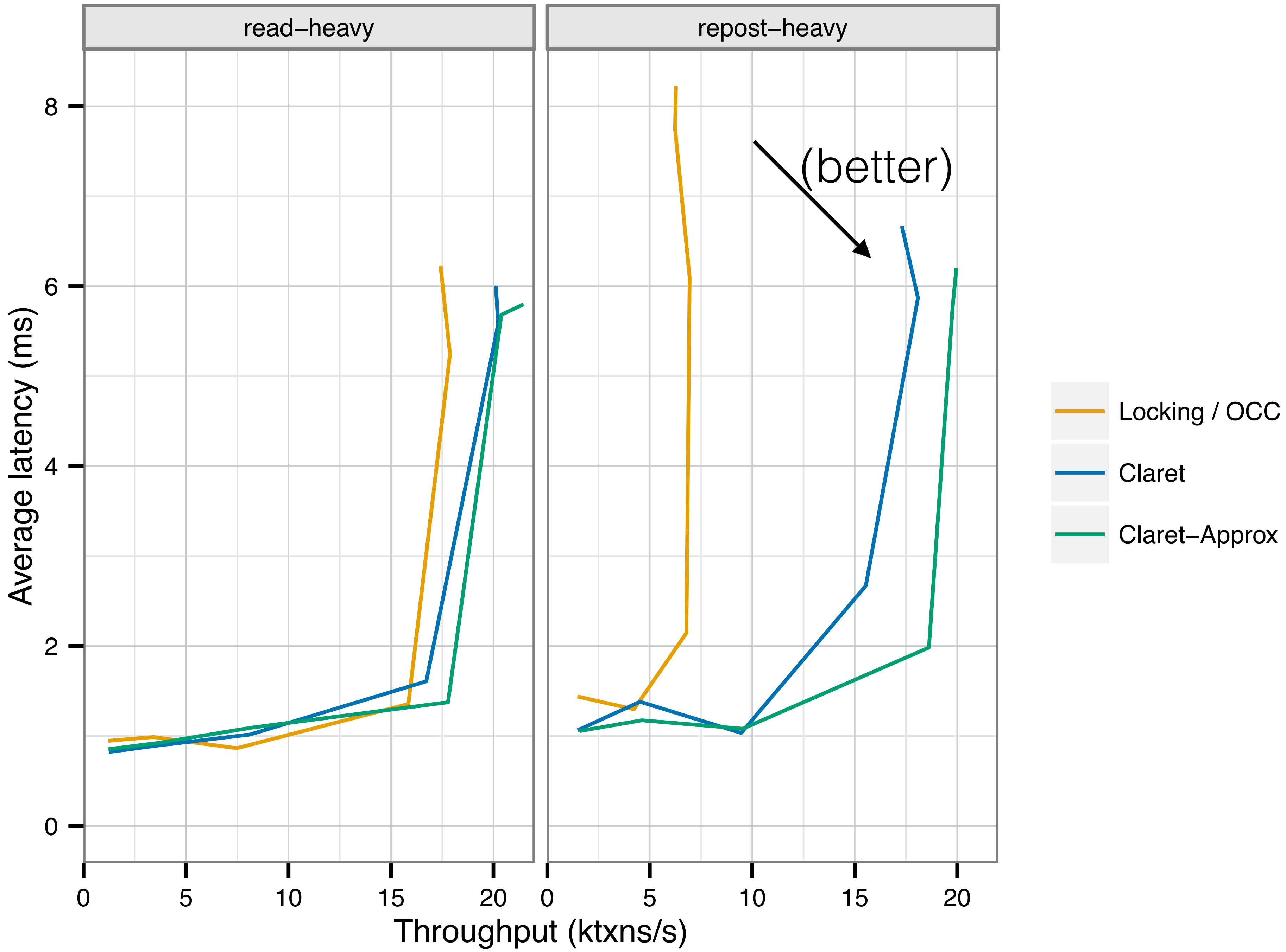
- realistic synthetic graph (Kronecker, scale 16)
- simple random user model, retweet more popular posts (*viral* effect)



Evaluation

Case study: Twitter clone

- realistic synthetic graph (Kronecker, scale 16)
- simple random user model, retweet more popular posts (*viral* effect)



Claret

Abstract Data Types for **NoSQL**

Flexible data model lets programmers express *intent*

Commutativity

Leverage type info for transaction performance

Approximate data types

Sanely trade off consistency for scalability

Claret

Abstract Data Types for NoSQL

Brandon Holt, Irene Zhang, Dan Ports, Mark Oskin, Luis Ceze
UNIVERSITY *of* WASHINGTON