

Coherence Implementation Patterns

Ben Stopford

The Royal Bank of Scotland

Some Ideas

Nothing More

Why do we use Coherence?

Fast?

Scalable?

Application layer?

Simplifying the Contract

- We don't want ACID all of the time
- We want to pick the bits we need when we need them
- We want to use the context of our business requirement to work our way around the ones we don't need.

Version your Objects

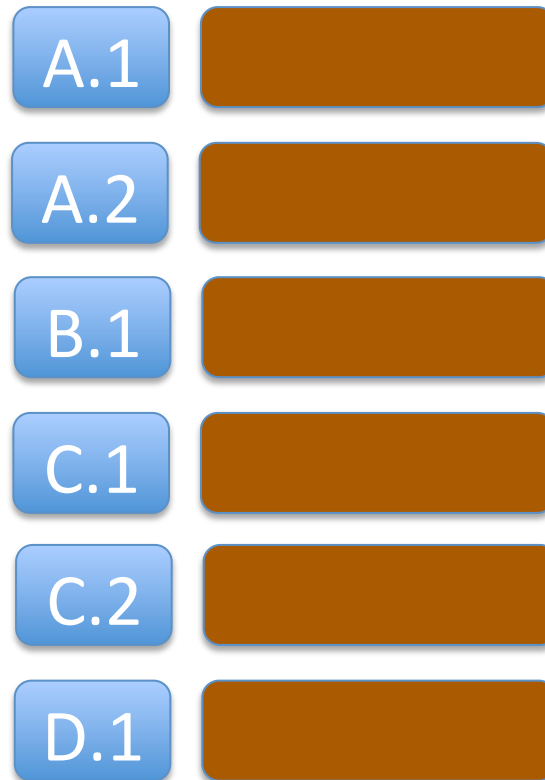
Why do we care?

Without versioning it's a free-for-all.

- What changed?
- Was something overwritten?
- How can you prevent concurrent updates?
- What did the system look like 10 seconds ago.
- How can I provide a consistent view?
- How to I ensure ordering of updates in an asynchronous system?

Versioning your Objects

Versioned Cache



Versioning your Objects

Cache

A.1



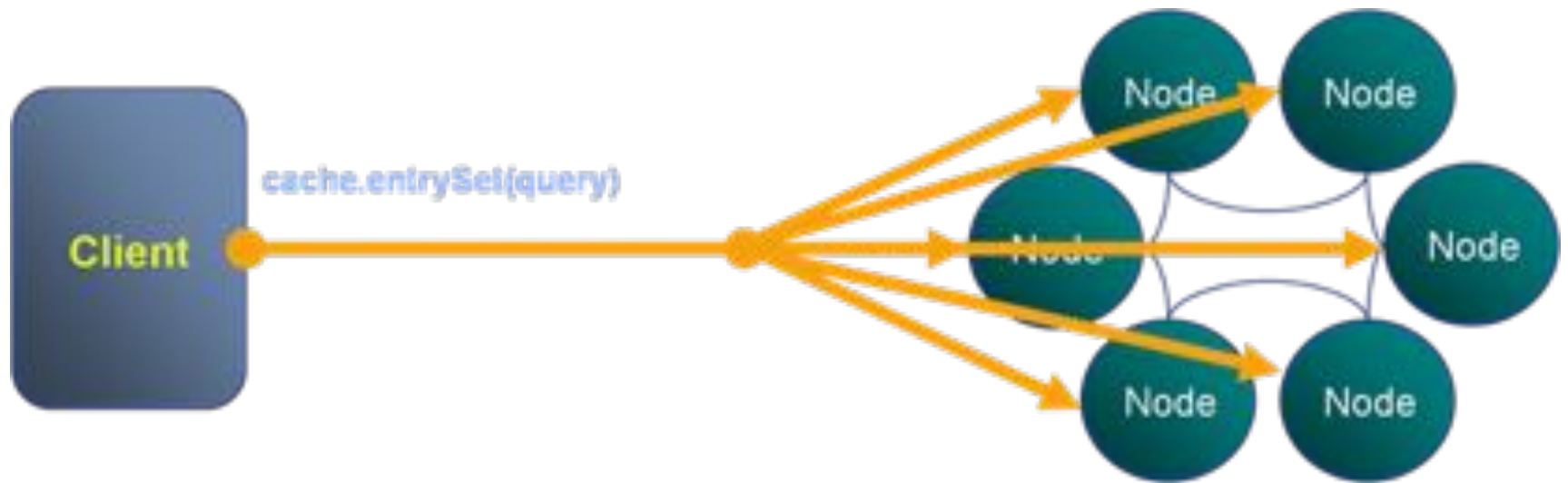
A.2



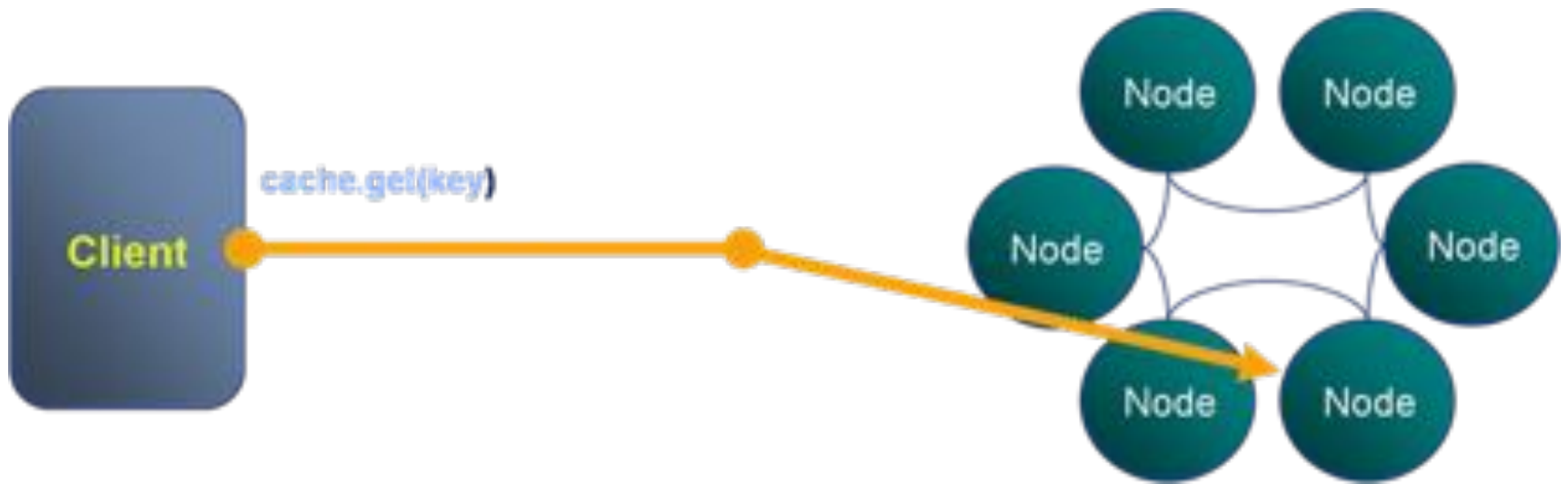
Coherence Trigger

New Version = Old Version + 1 ??

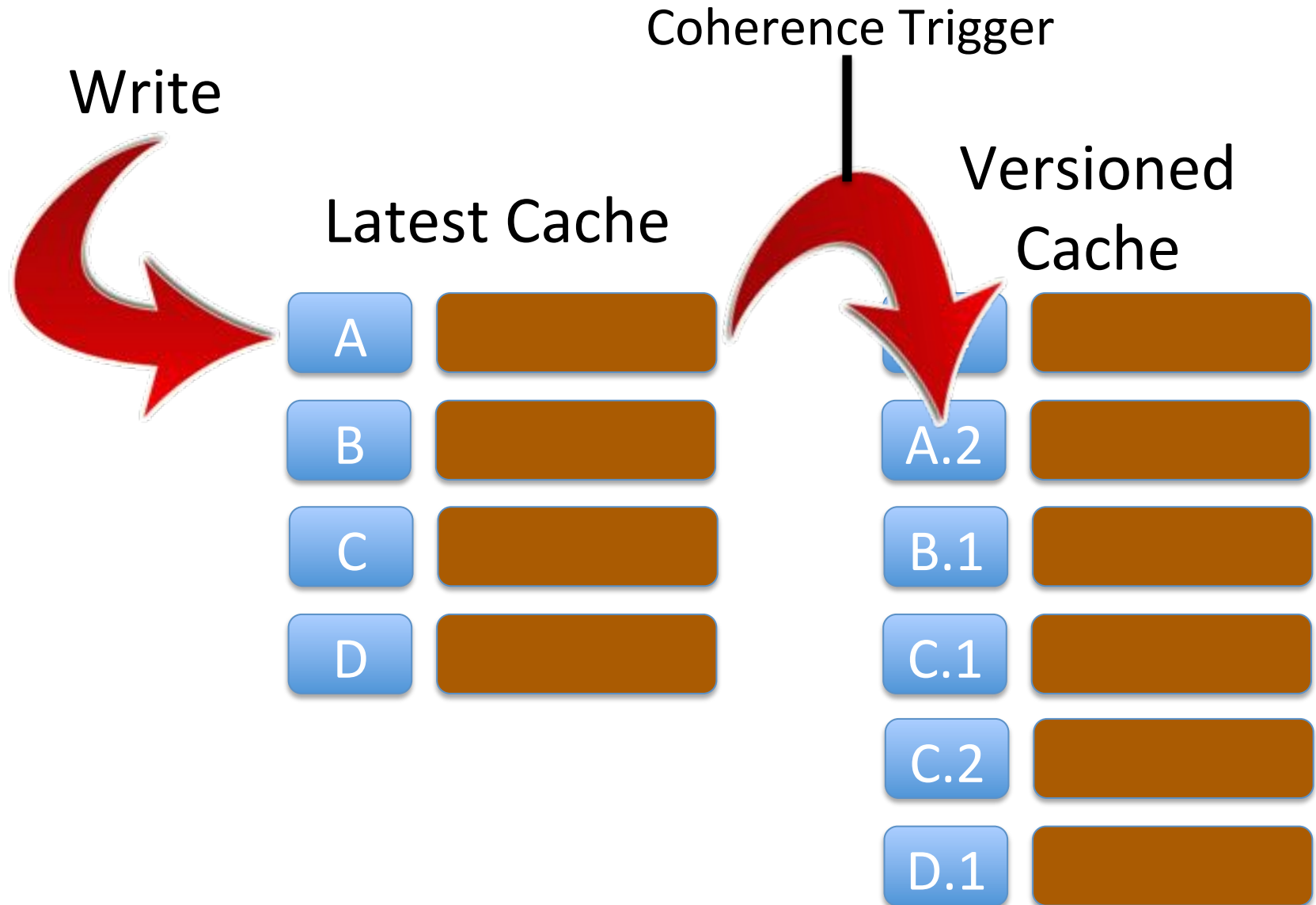
Running a Coherence Filter



Using Key-Based Access



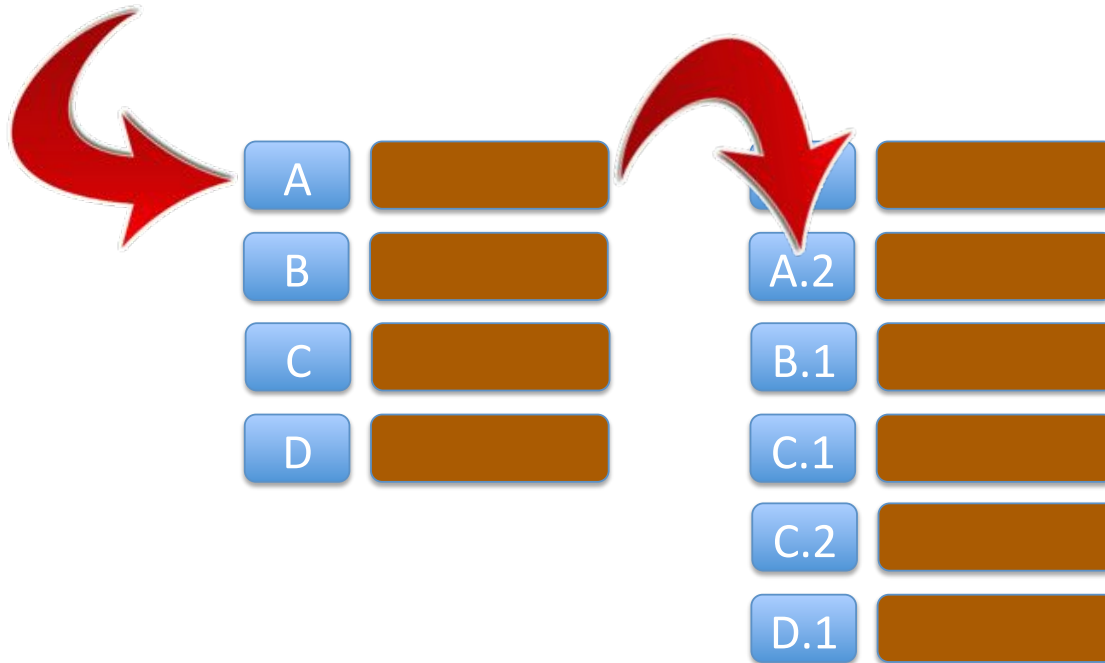
Latest / Versioned Pattern



Latest / Versioned Pattern

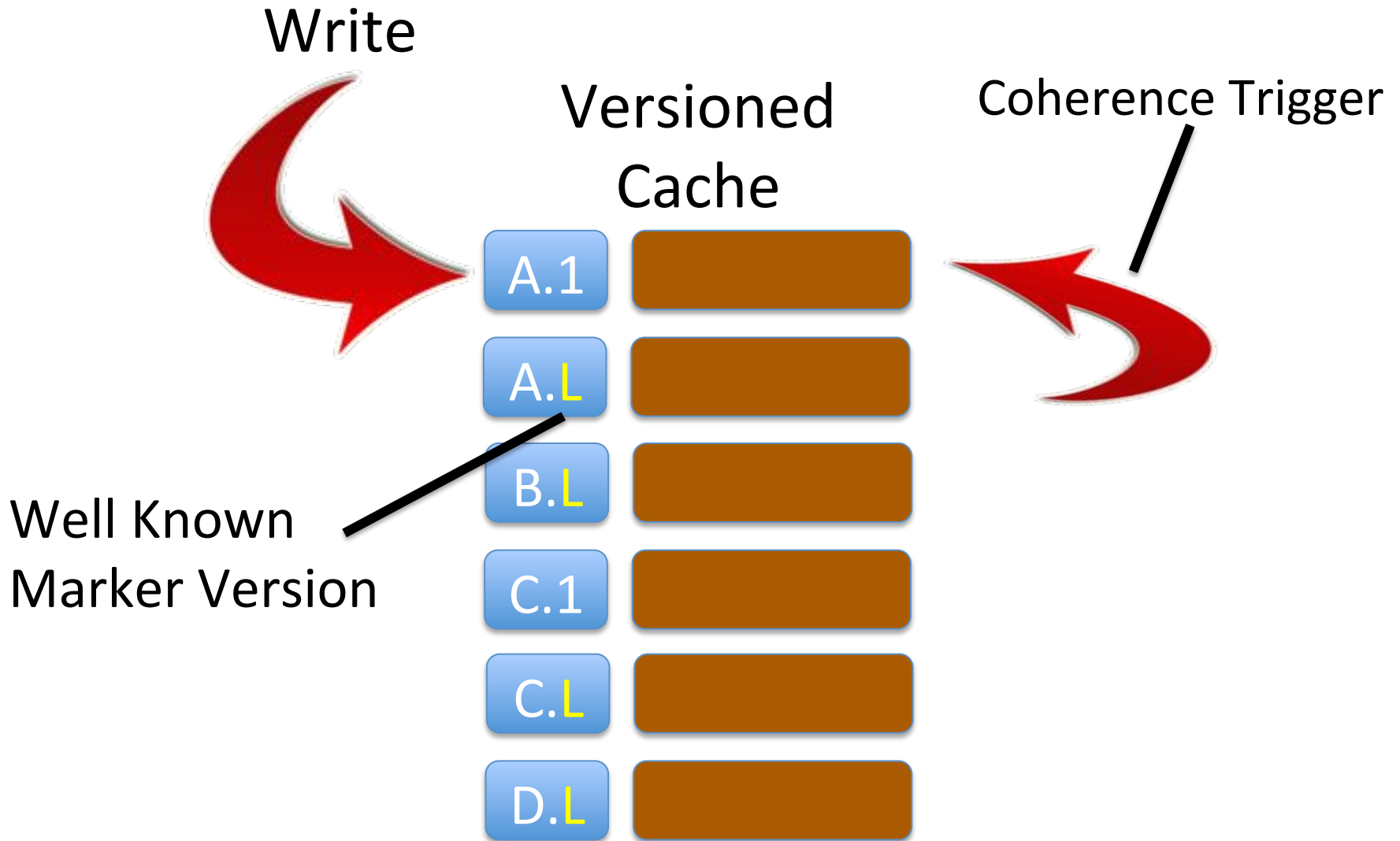
Latest Cache Key = [Business Key]

Versioned Cache Key = [Business Key][Version]



Suffers from data duplication

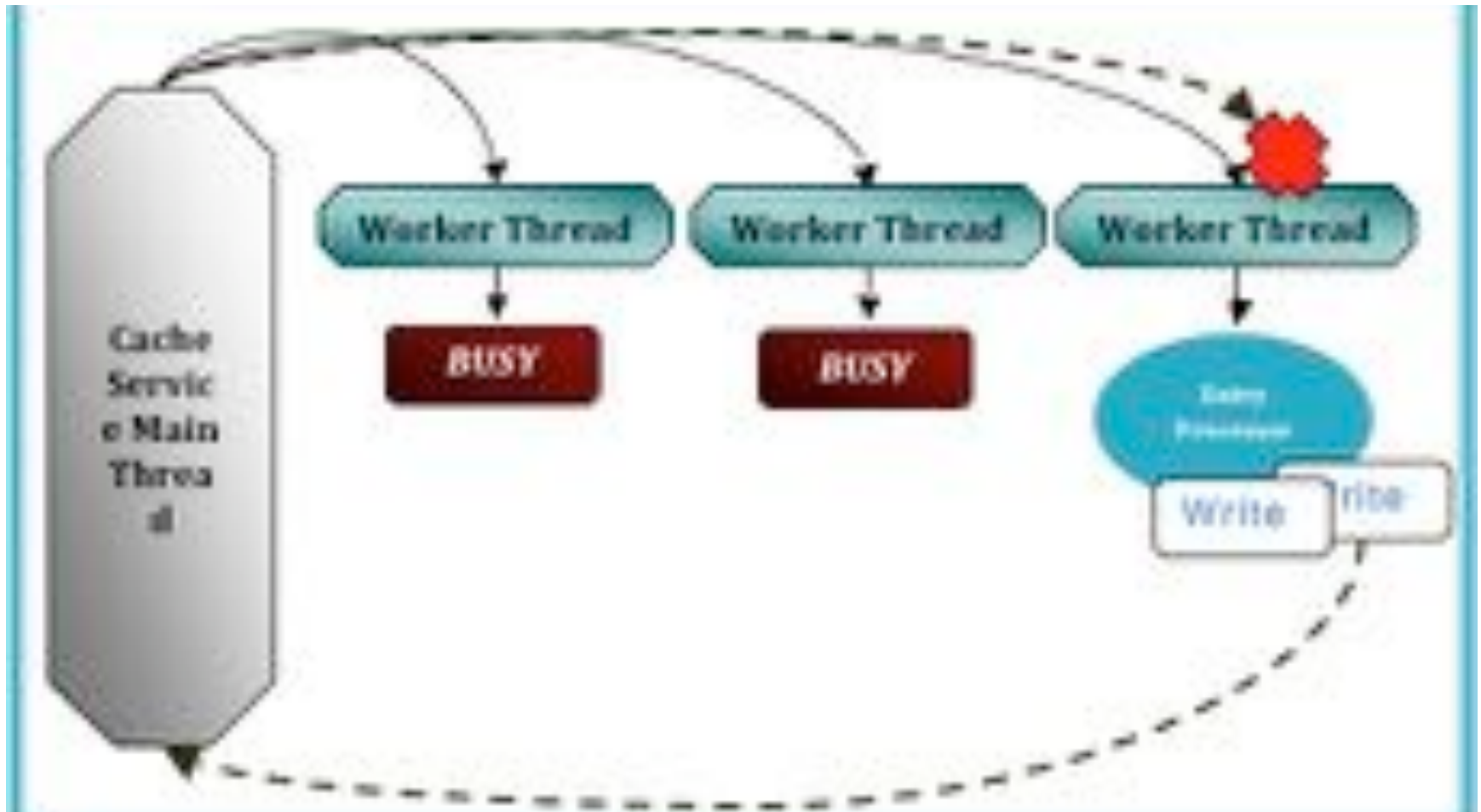
Latest Marker Pattern



However our trigger can't use
cache.put()

Why?

Need to consider the threading model



So we'll need to use the backing map directly

```
public void copyObjectToVersionedCacheAddingVersion(MapTrigger.Entry entry)
{
    MyValue value = (MyValue)entry.getValue();
    MyKey versionedKey = (MyKey)value.getKey();

    BinaryEntry binary = (BinaryEntry)entry;
    Binary binaryValue = binaryEntry.getBinaryValue();

    Map map = binary.getContext().getBackingMap("VersionedCacheName");
    map.put(toBinary(versionedKey), binaryValue);
}
```

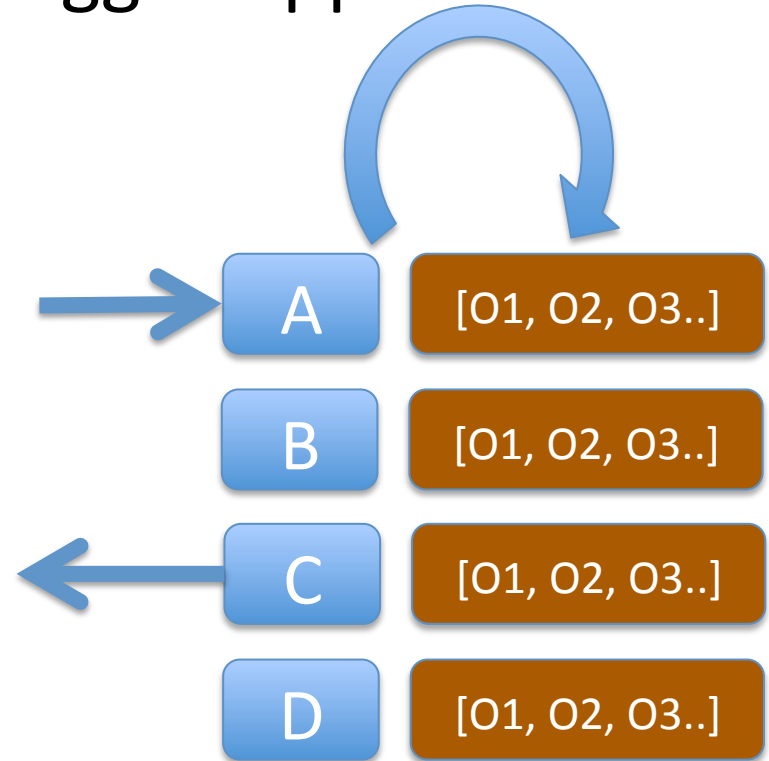
A third approach

The Collections Cache

Trigger Appends to Collection

`collectionsCache.put(key, val);`

`collectionsCache.invoke(key,
new LastValueGetter());`
...or override backing store



CollectionsCache

So we have 3 patterns for managing
versioning whilst retaining key
based access

Using versioning to manage concurrent changes

Multi Version Concurrency Control
(MVCC)

Cache

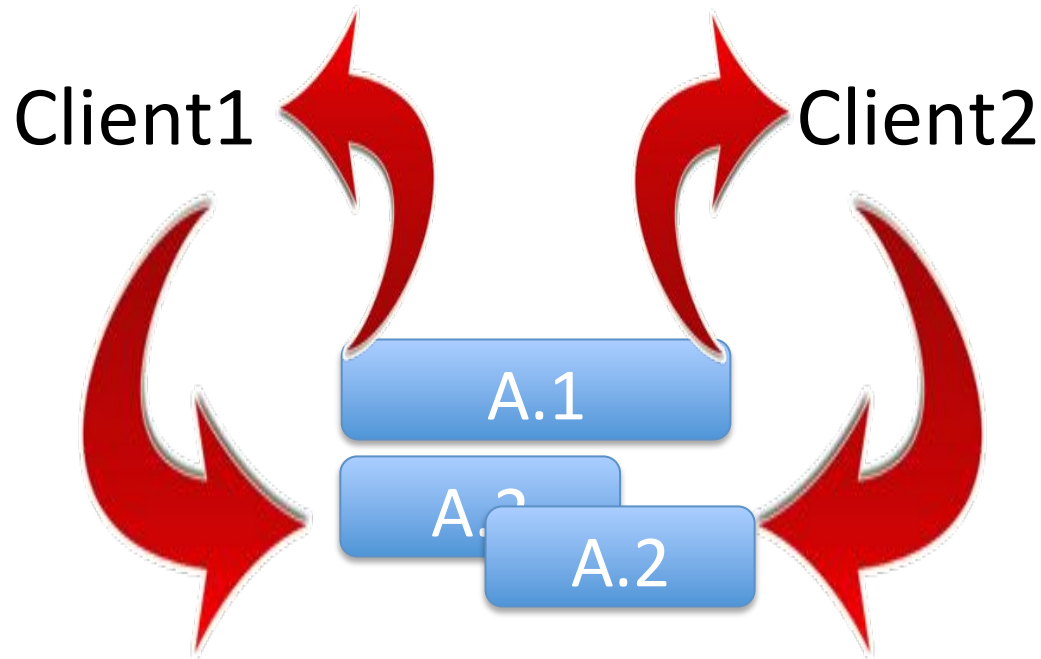


Coherence Trigger

New Version = Old Version + 1 ??

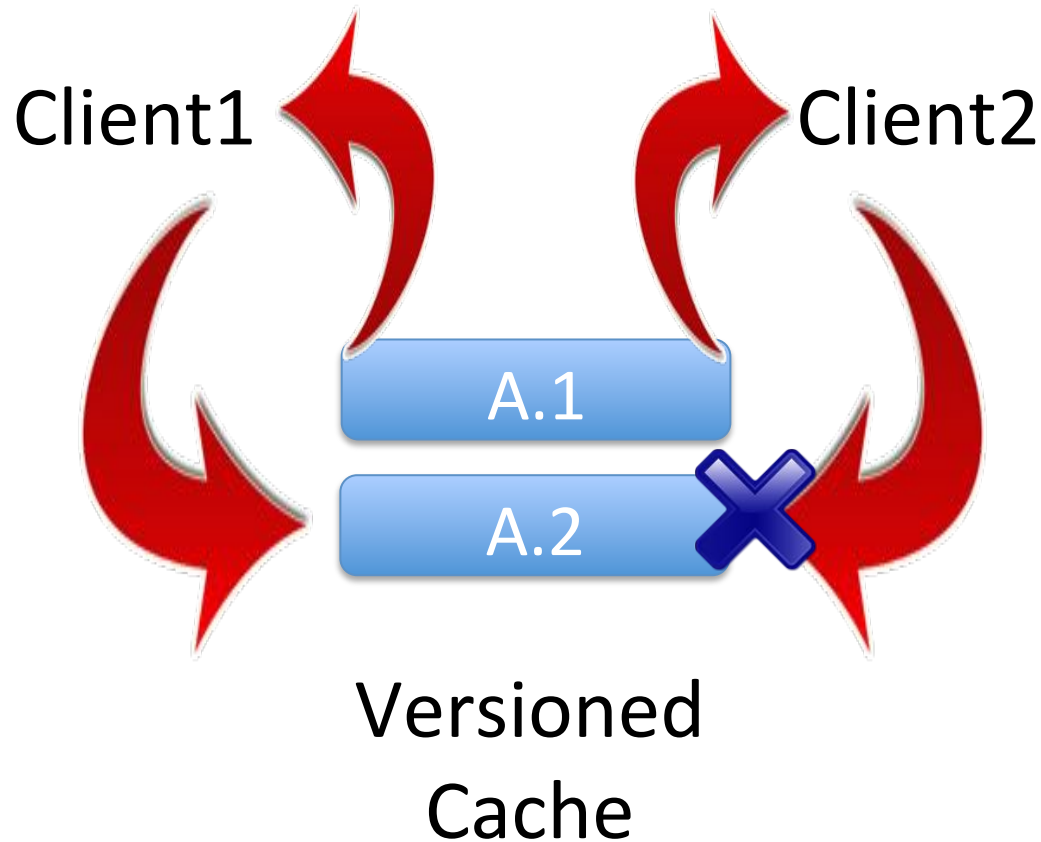
Concurrent Object Update

(2 Clients update the same object at the same time)



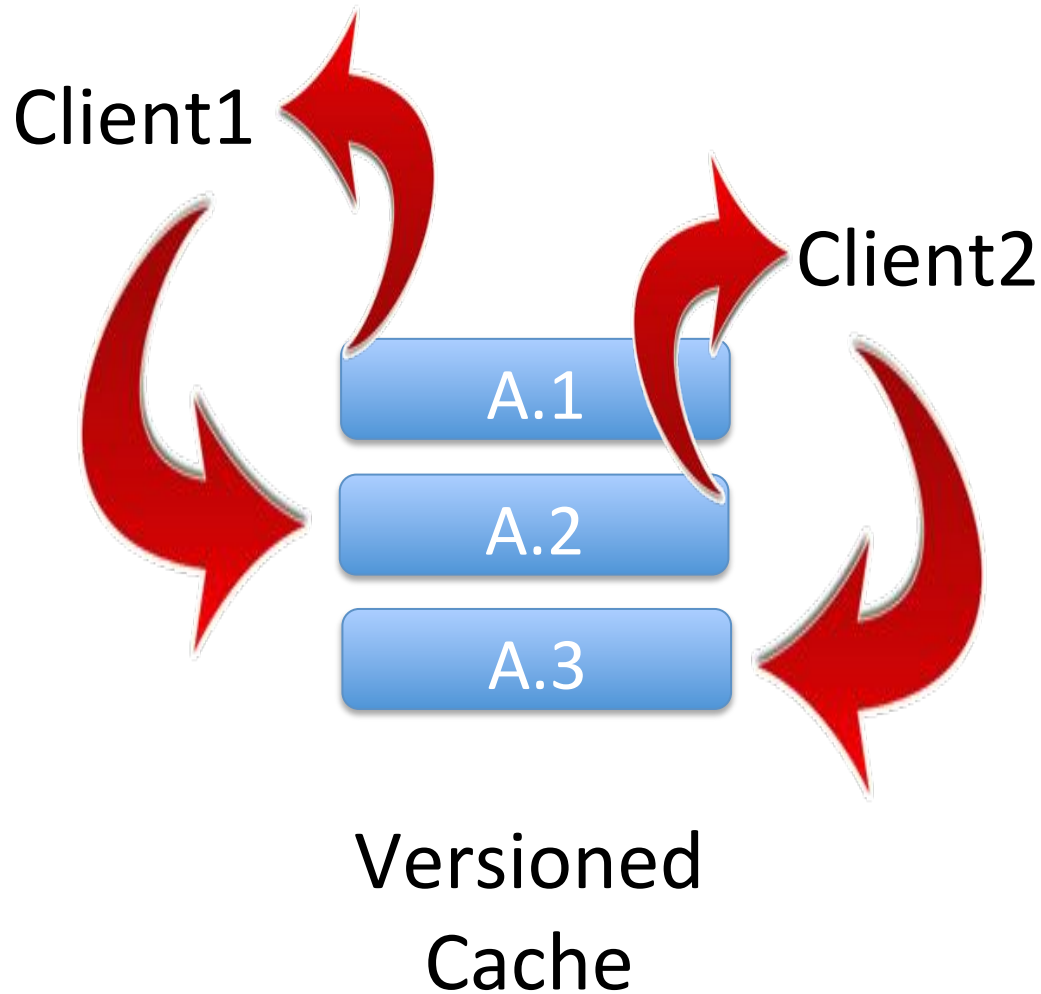
Concurrent Object Update

(Client2 fails to update dirty object)

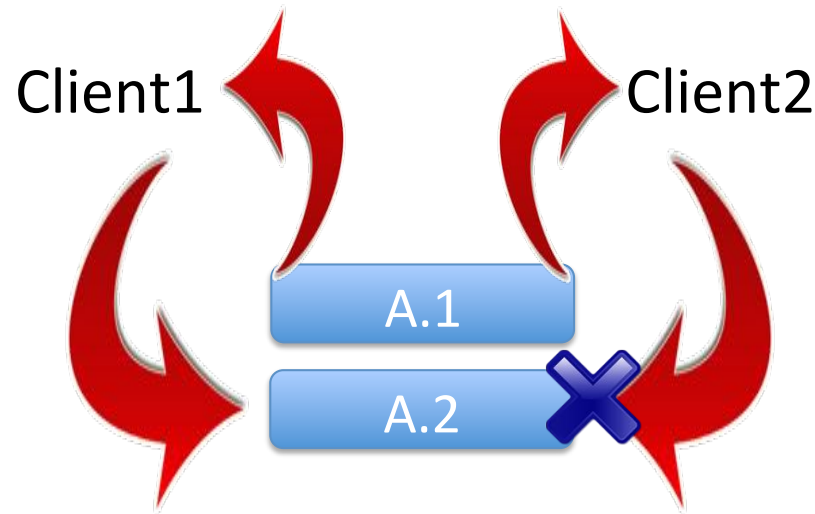


Concurrent Object Update

(Client 2 updates clean object)



So a concurrent update results in an error and must be retried.

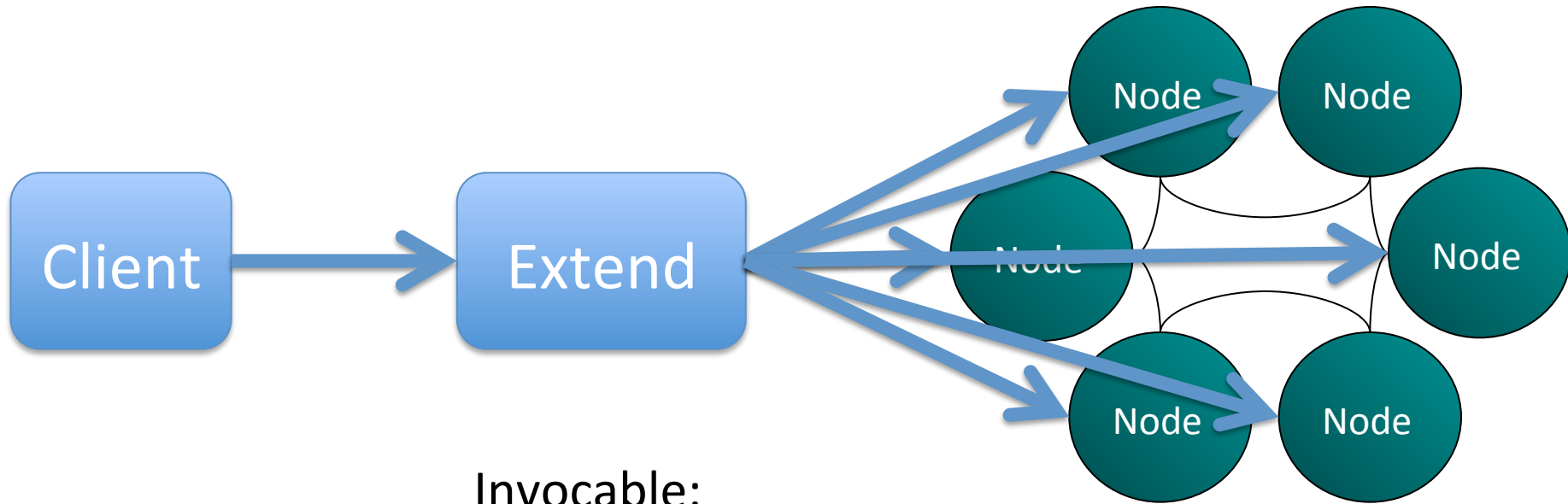


What's going to happen if we are
using putAll?

Reliable PutAll

We want putAll to tell us which objects failed the write process

Reliable PutAll



Invocable:

- Split keys by member
- Send appropriate values to each member
- Collect any exceptions returned

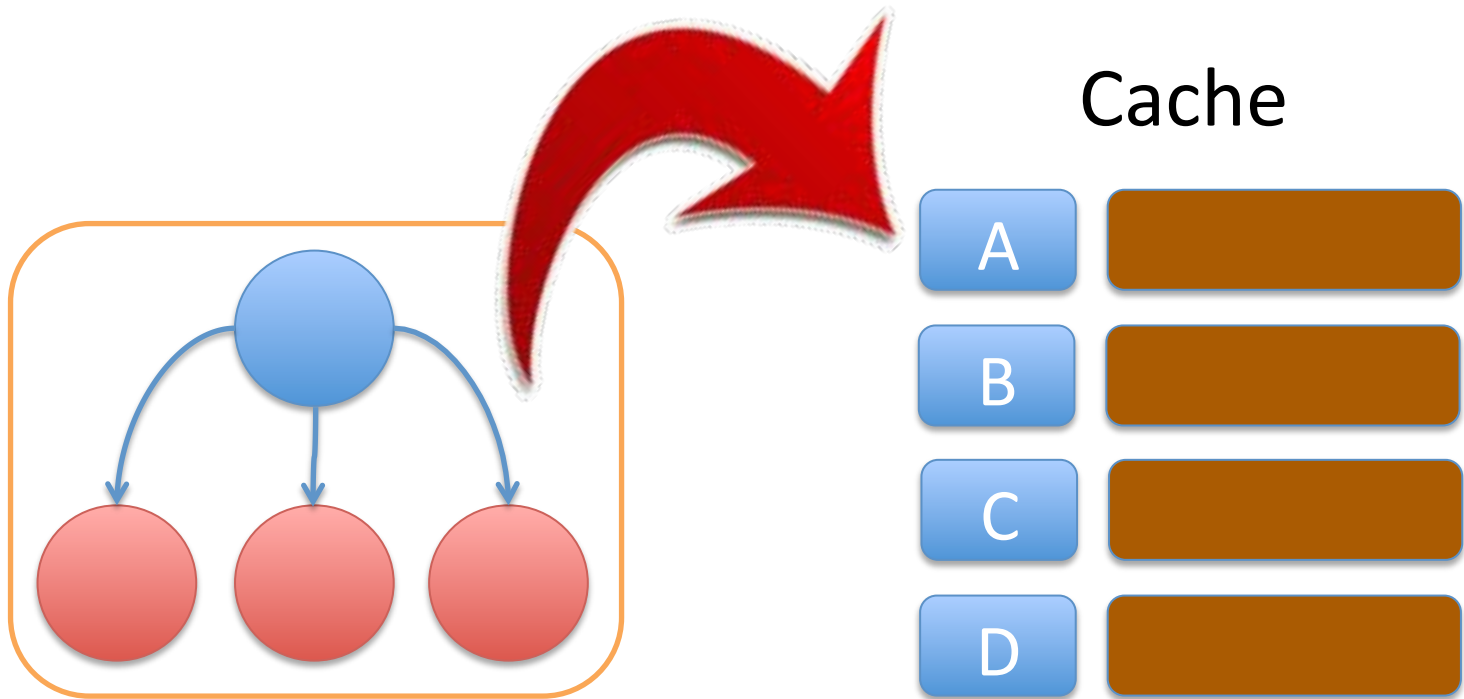
Invocable:

- Write entries to backing map (we use an EP for this)

This gives us a reliable mechanism
for knowing what worked and what
failed

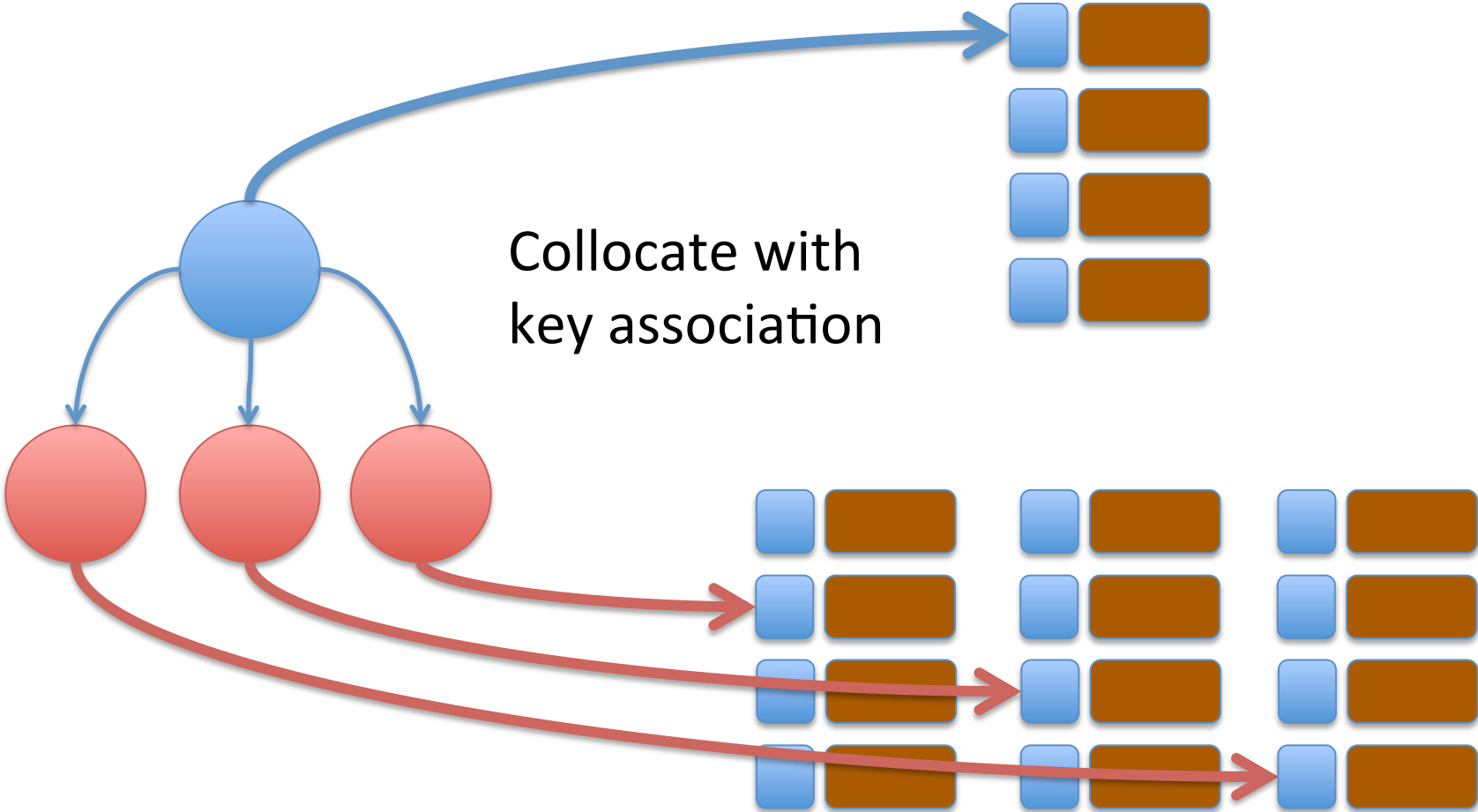
Synthesising Transactionality

The Fat Object Method

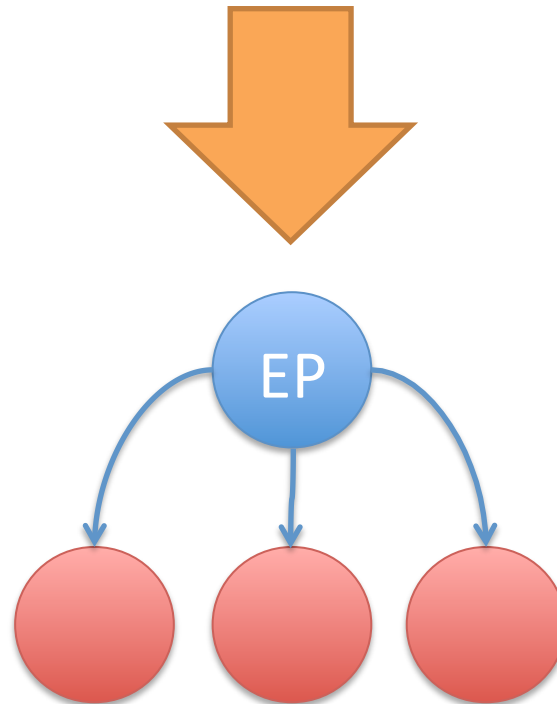


The Single Entry Point Method

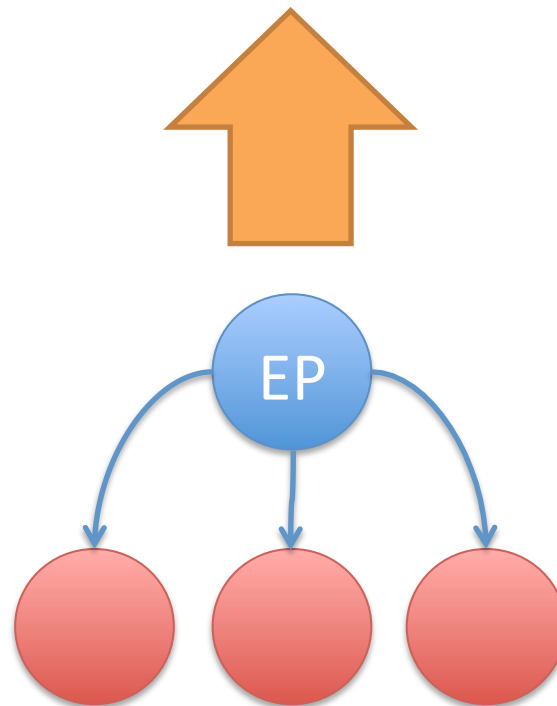
(objects are stored separately)



All **writes** synchronize on the primary object.

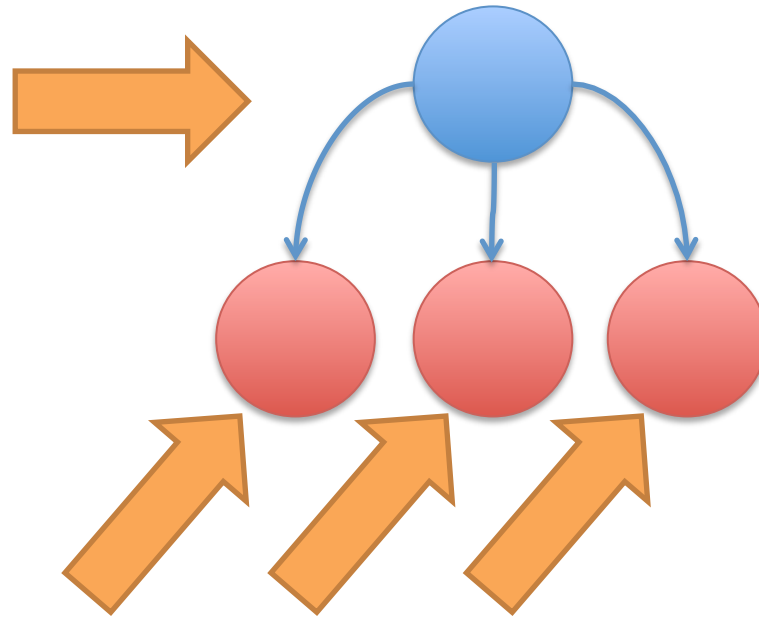


All **reads** synchronize on the primary object.



Writing Orphaned Objects

Write read
entry point
object last



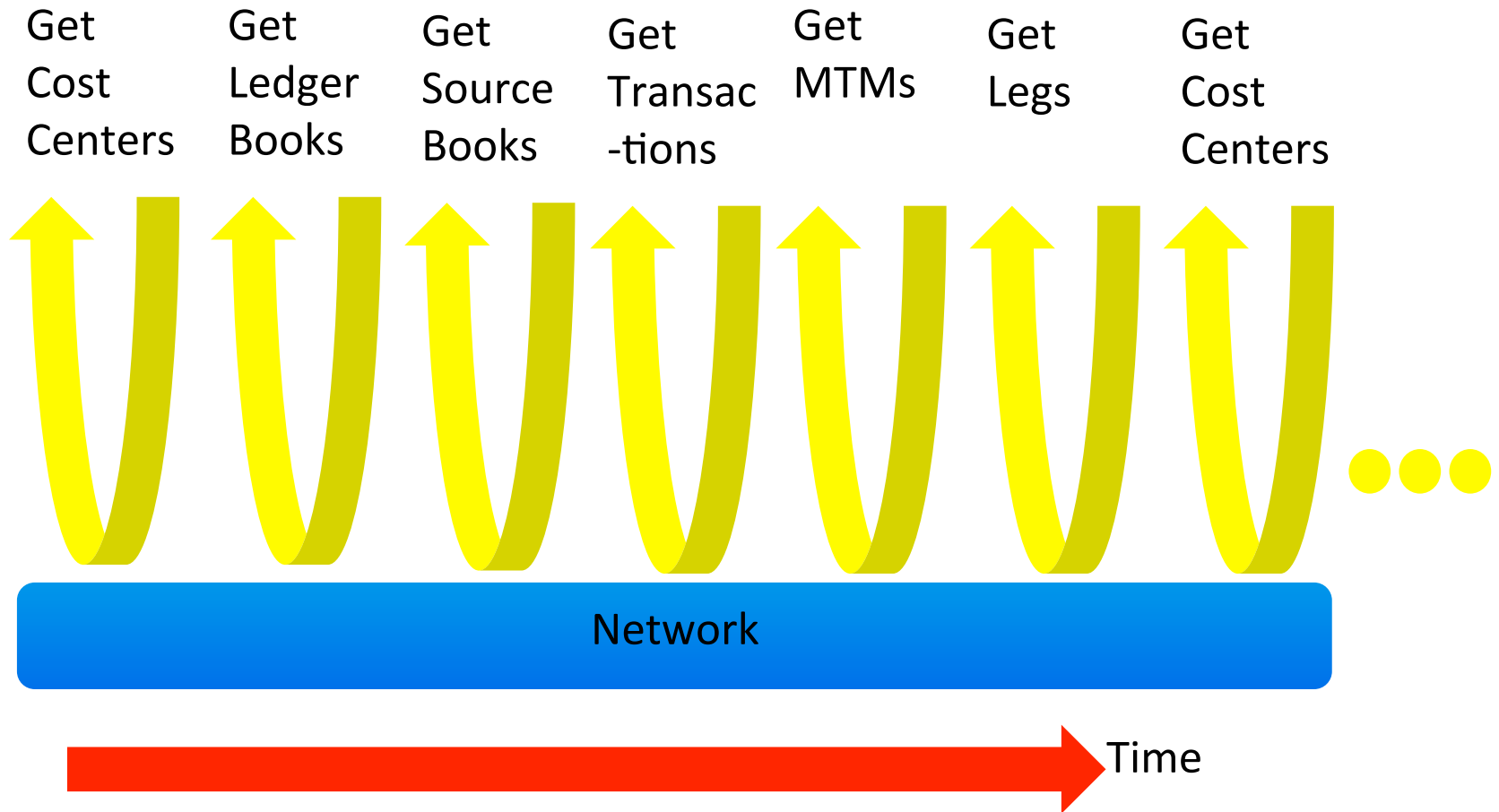
Write orphaned objects first

This mechanism is subtly flawed

Reading several objects as an atomic unit

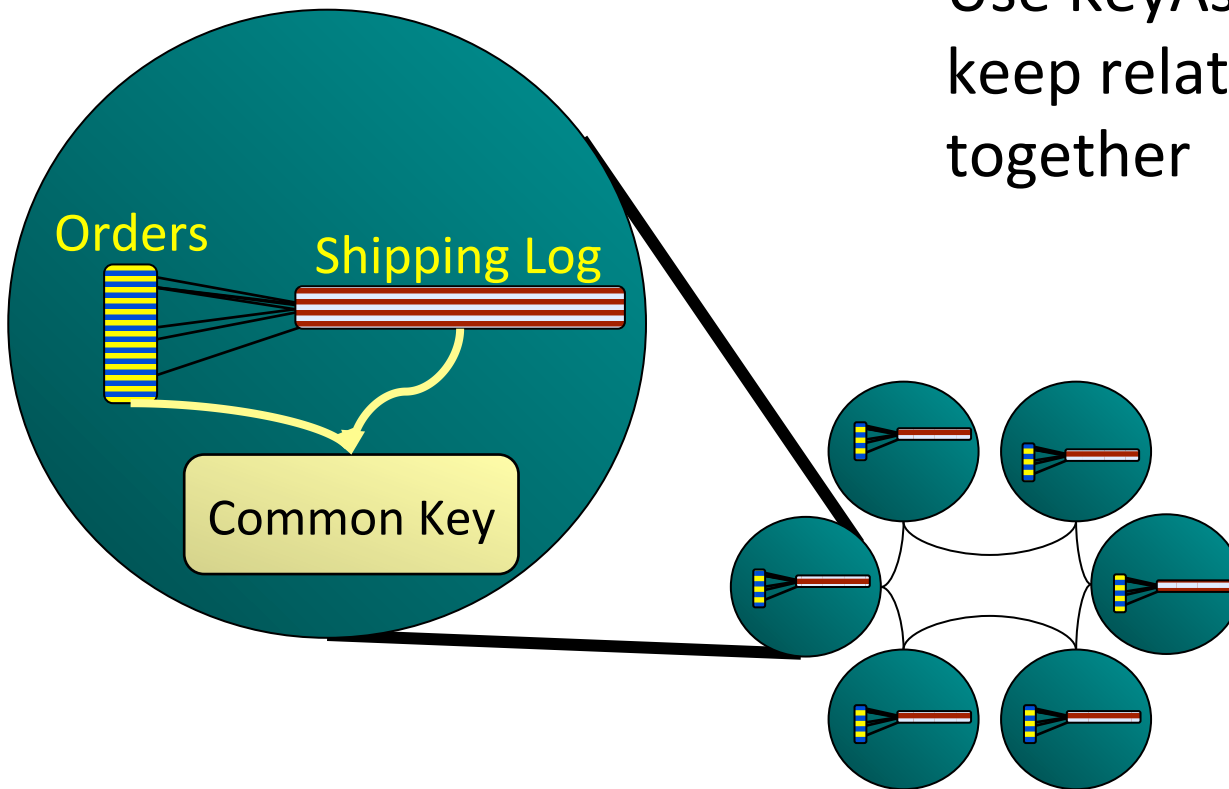
aka Joins

The trivial approach to joins

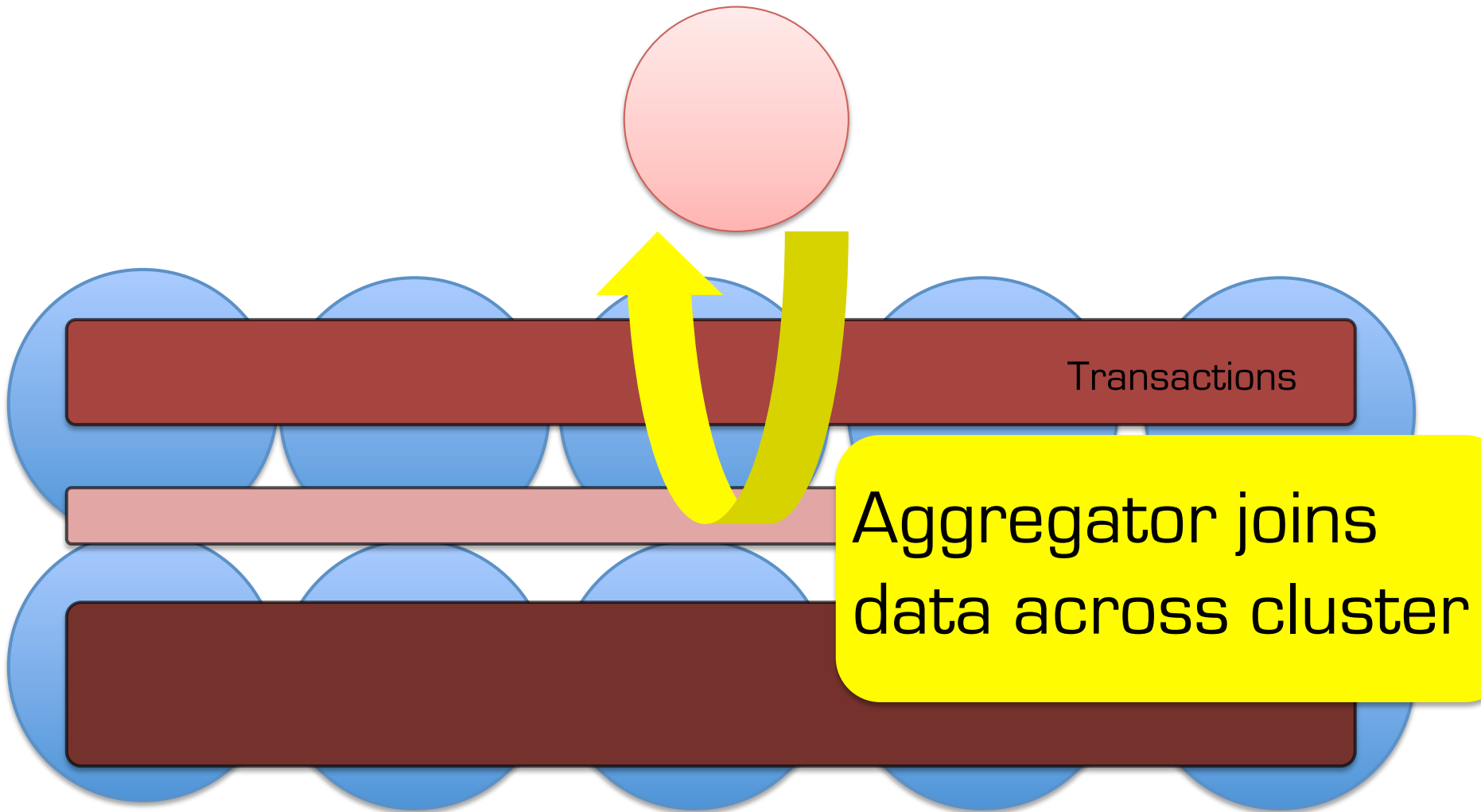


Server Side, Sharded Joins

Use KeyAssociation to keep related entities together



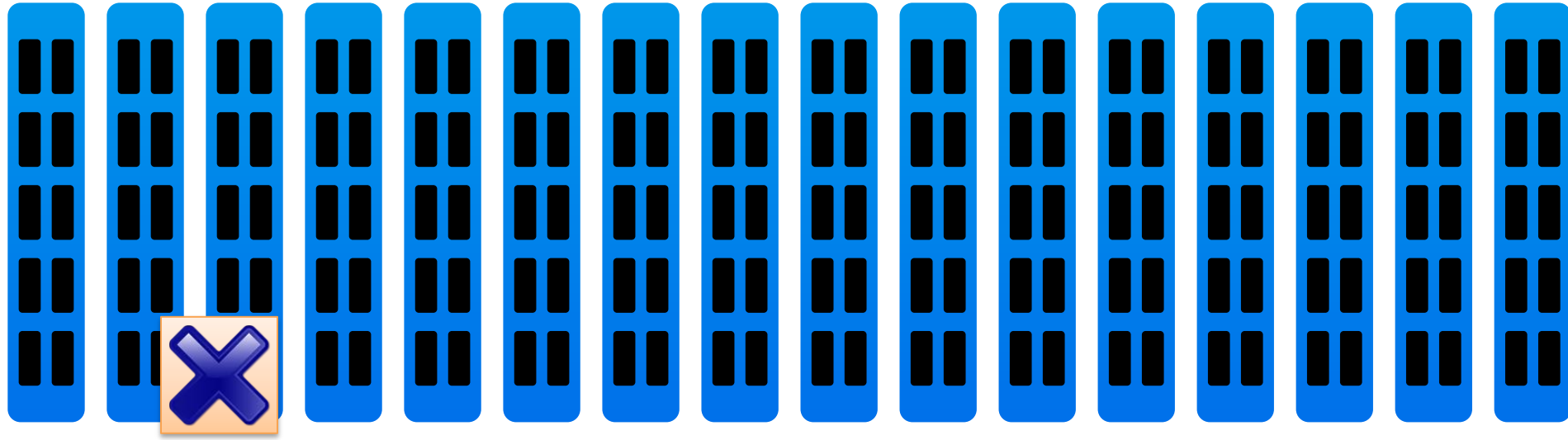
Server Side, Sharded Joins



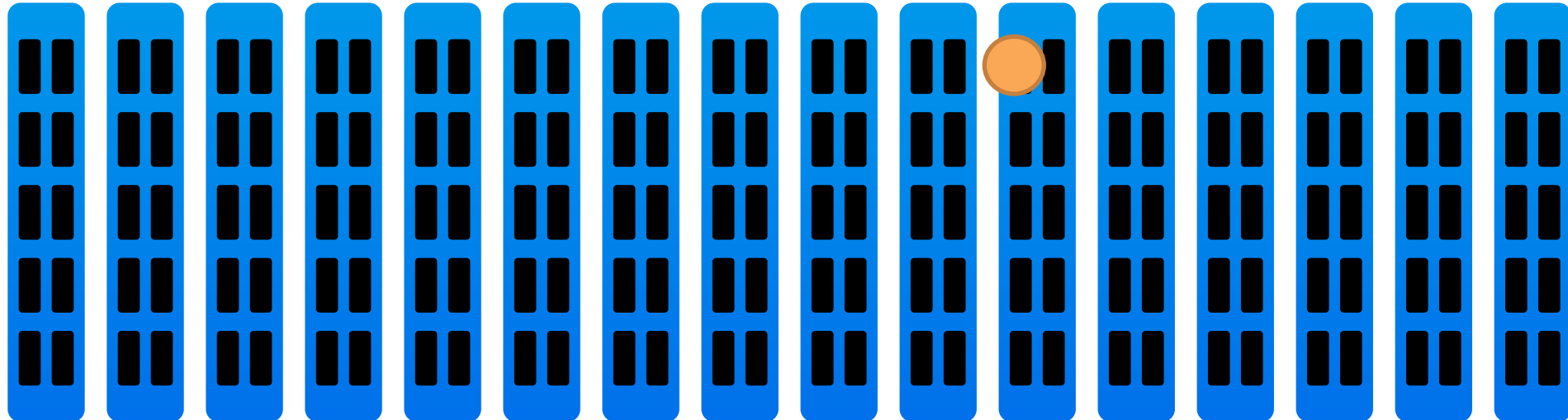
So we have a set of mechanisms for reading and writing groups of related objects.

Cluster Singleton Service

A service that automatically restarts
after failure



A service that automatically restarts
after failure



What is the cluster singleton good for

- Adding indexes
- Loading data
- Keeping data up to date
- Updating cluster time
- You can probably think of a bunch of others yourselves.

Code for Cluster Singleton

```
//run in a new thread on every Cache Server
while (true) {
    boolean gotLock = lockCache.lock("singletonLock", -1);
    if (gotLock) {
        //Start singletons
        wait();
    }
}
```

Implementing Consistent Views and Repeatable Queries

Bi-temporal

```
public interface MyBusinessObject{  
    //data  
    public Date getBusinessDate();  
    public Date validFrom();  
    public Date validTo();  
}
```



Business
Time

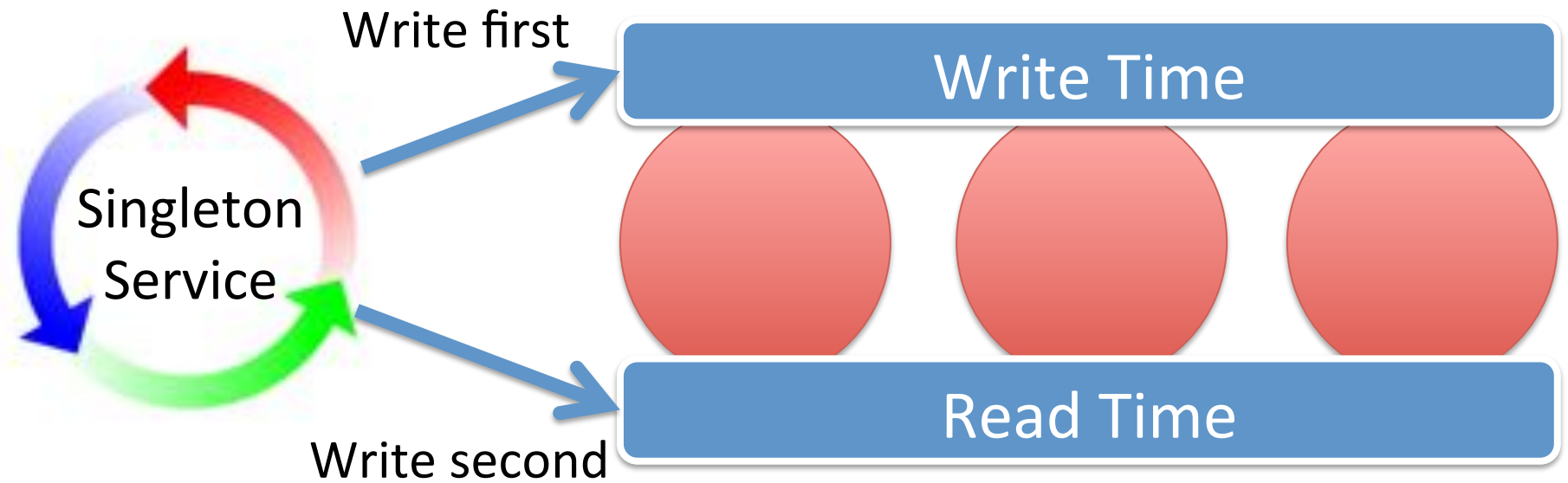
System
Time

Where does the System Time
come from?

You can't use the
`System.currentTimeMillis()` in a
distributed environment!

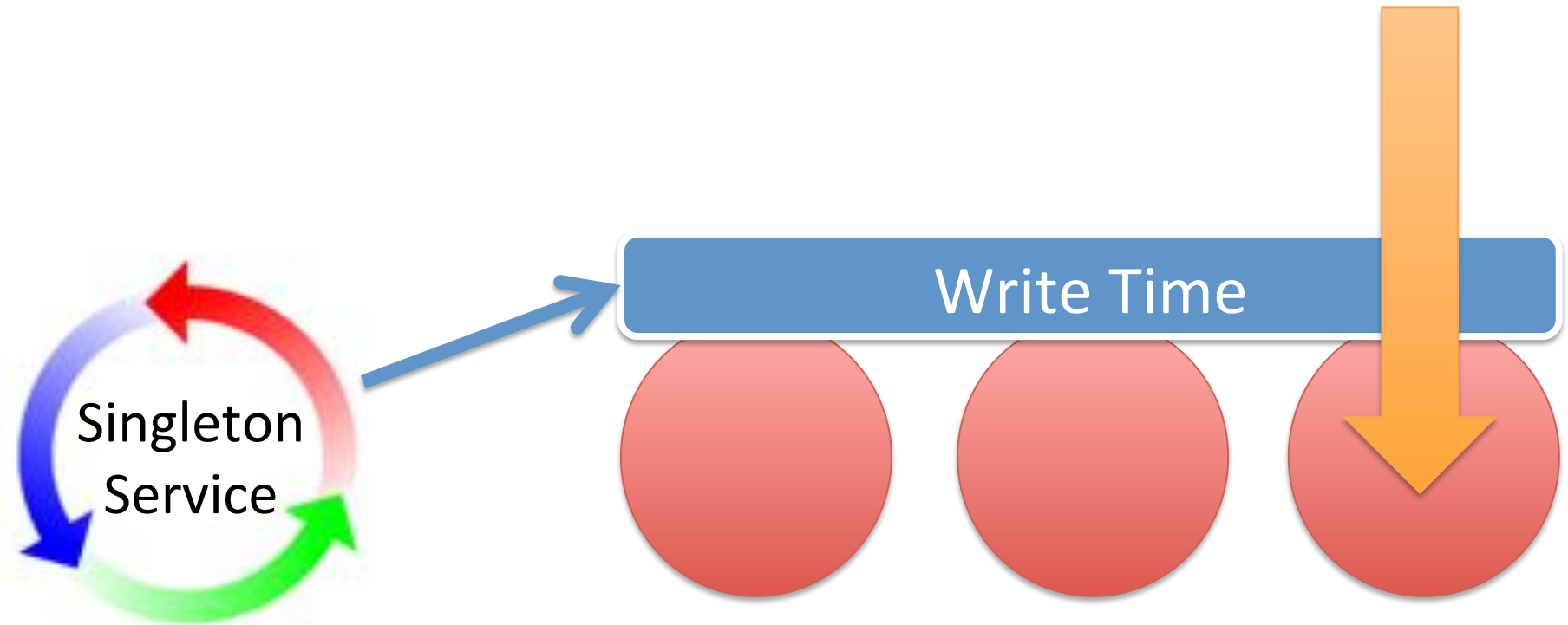
You need a cluster synchronised
clock

Repeatable Time: A guaranteed Tick

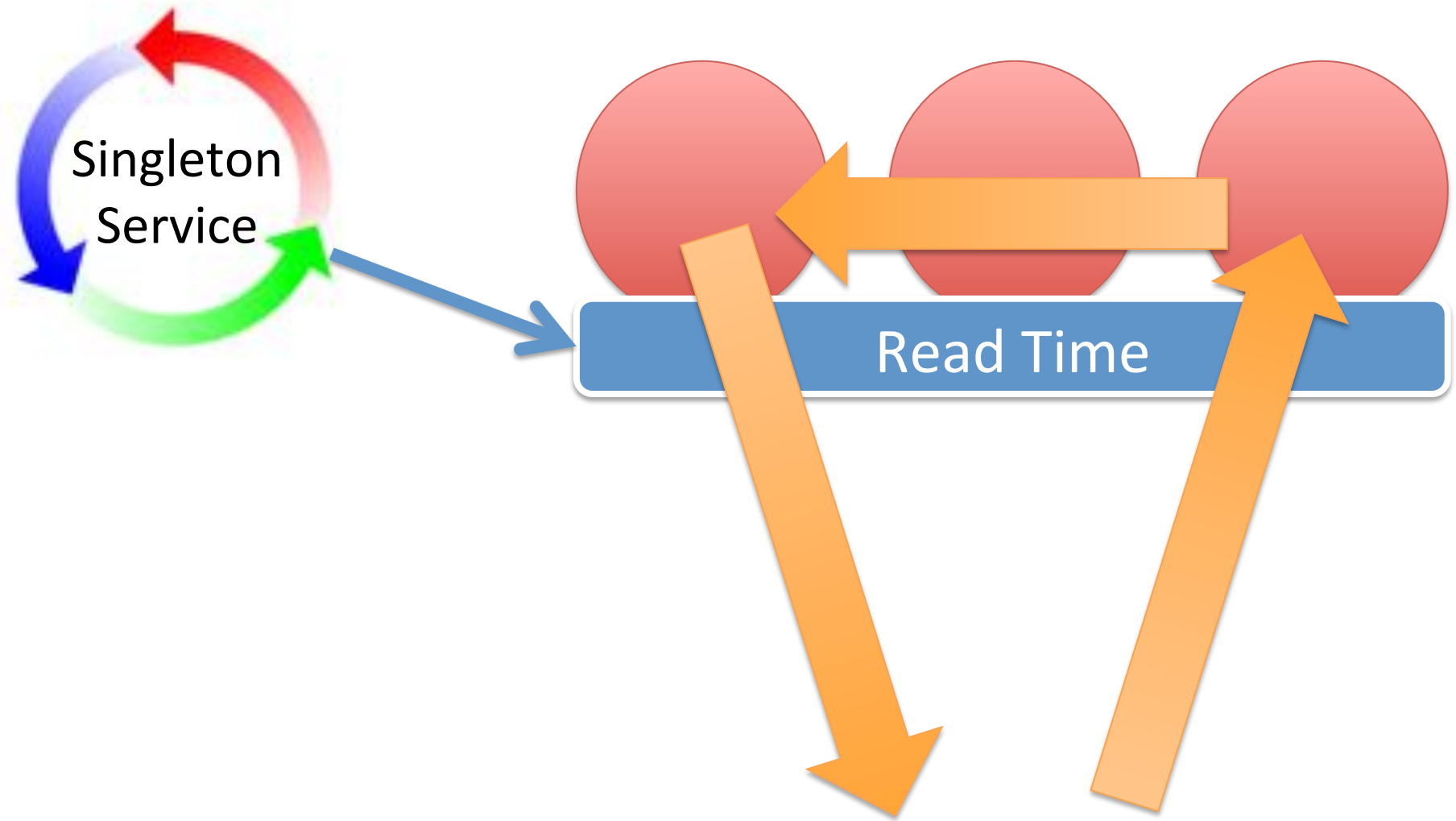


Replicated Caches
(pessimistic)

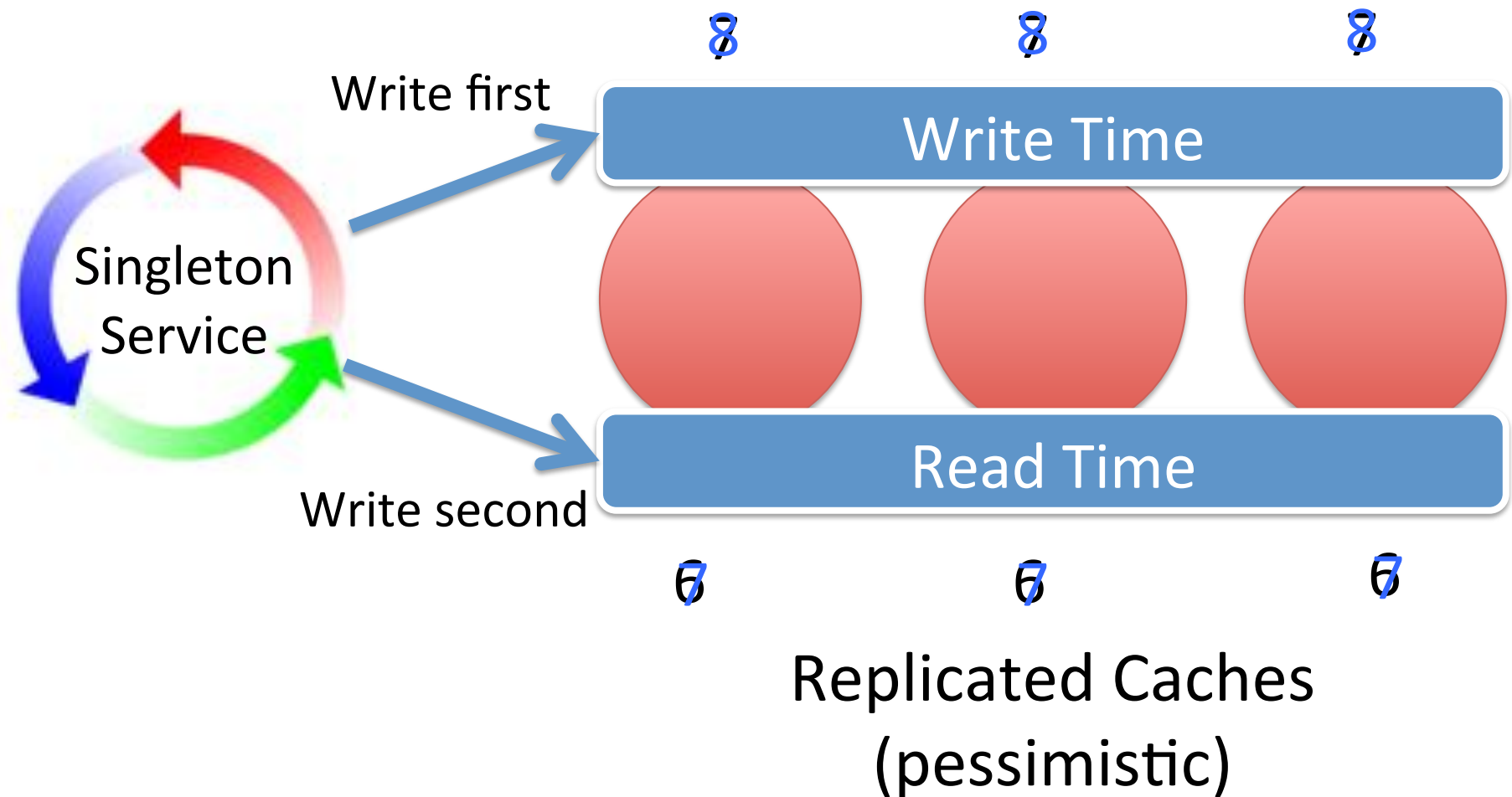
As we add objects we timestamp them with Write Time



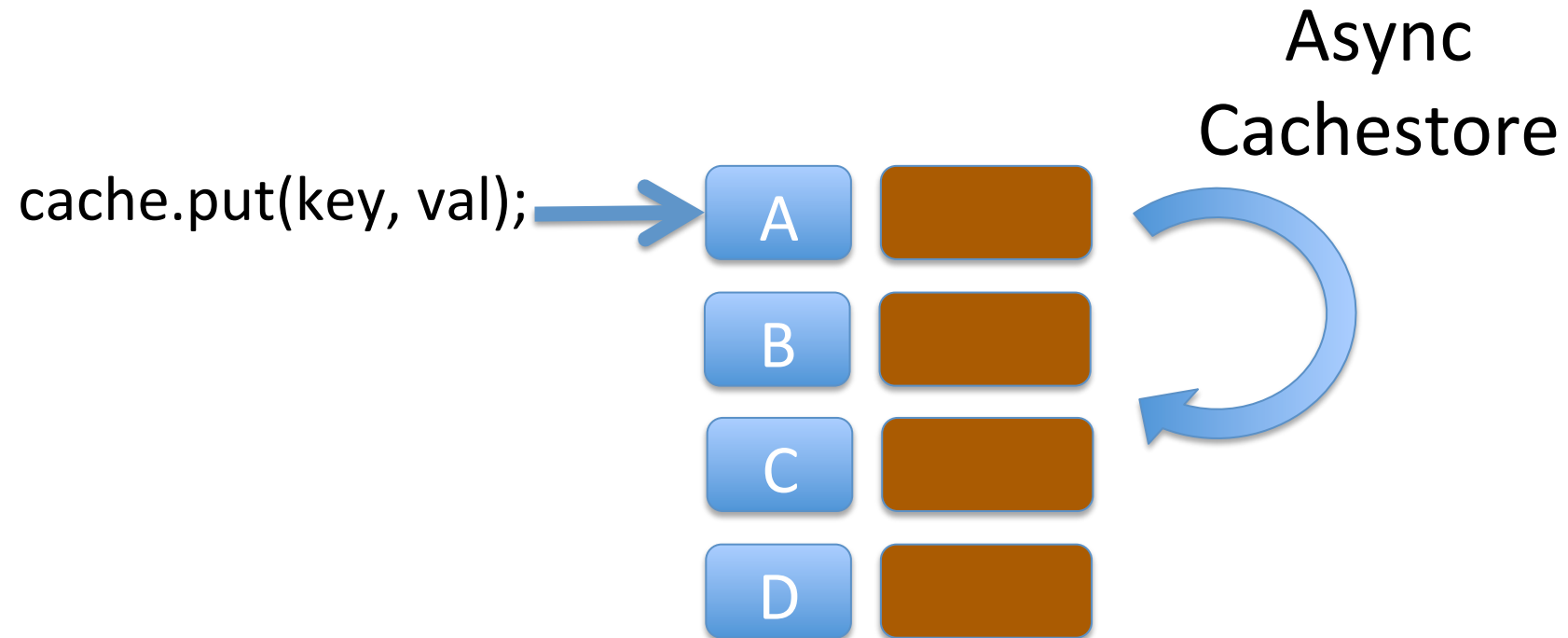
When we read objects we use Read Time



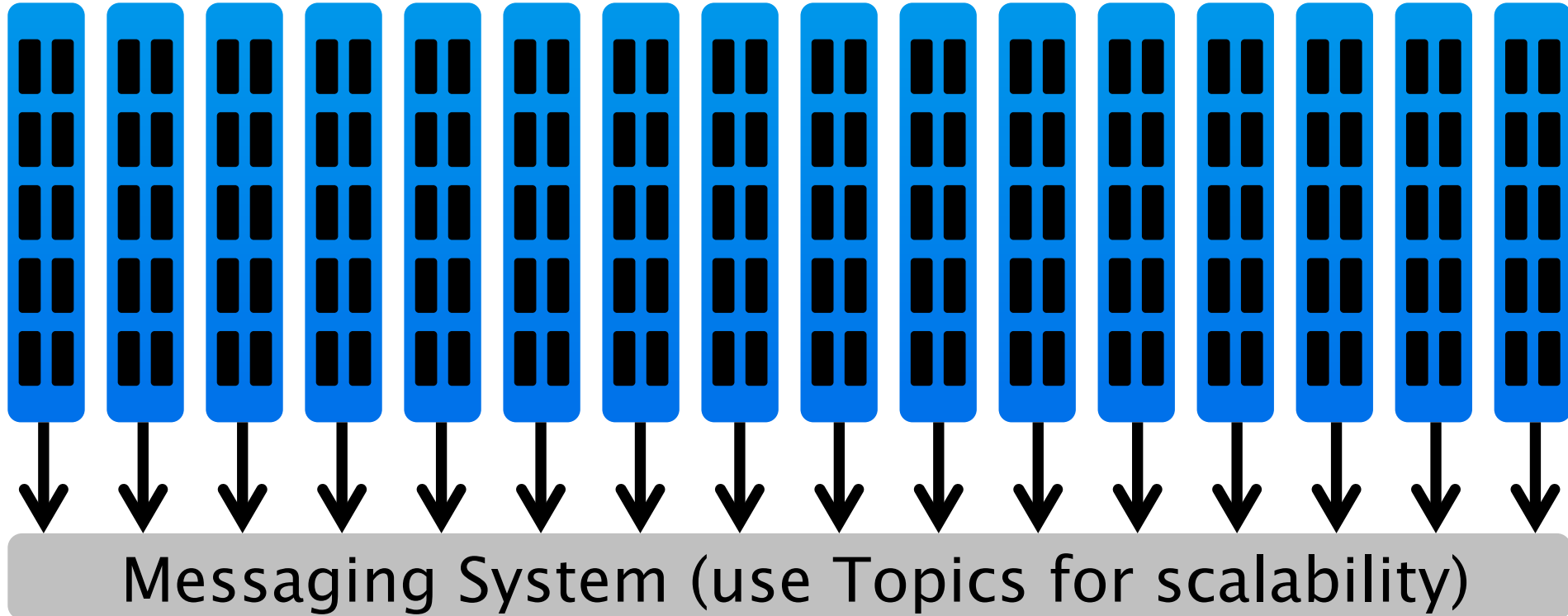
Repeatable Time: A guaranteed Tick



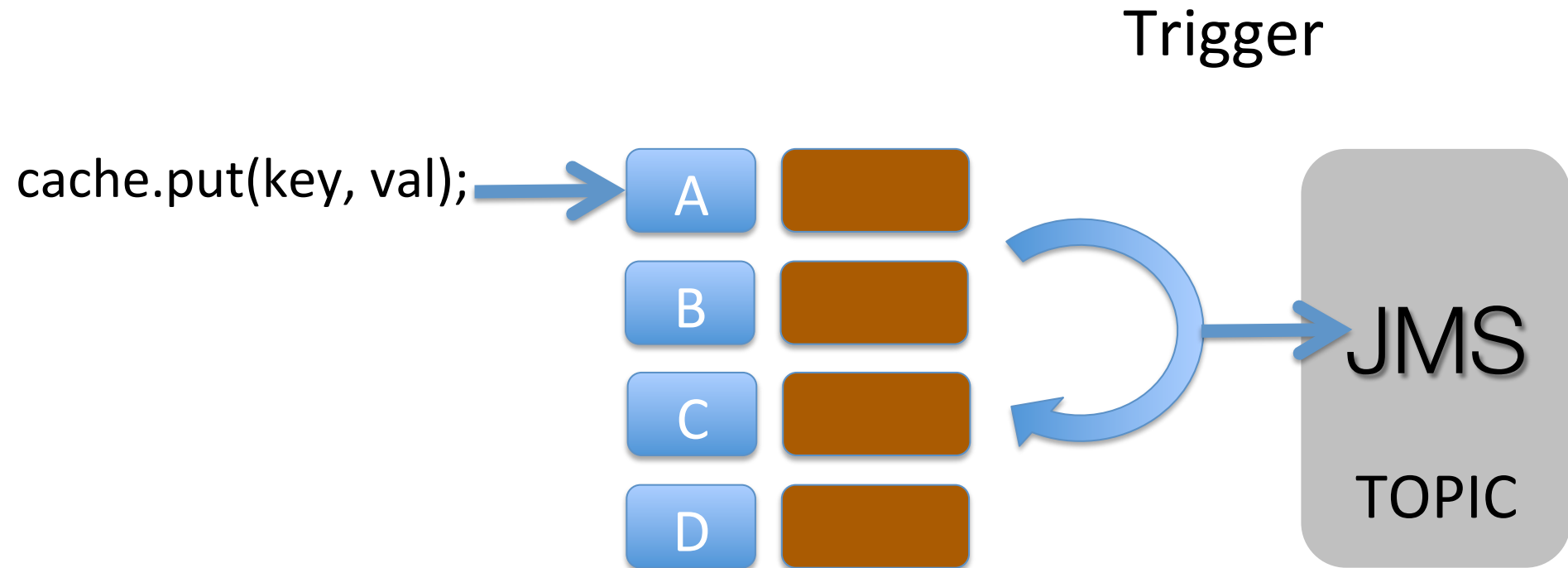
Event Based Processing



Messaging as a System of Record



Messaging as a System of Record

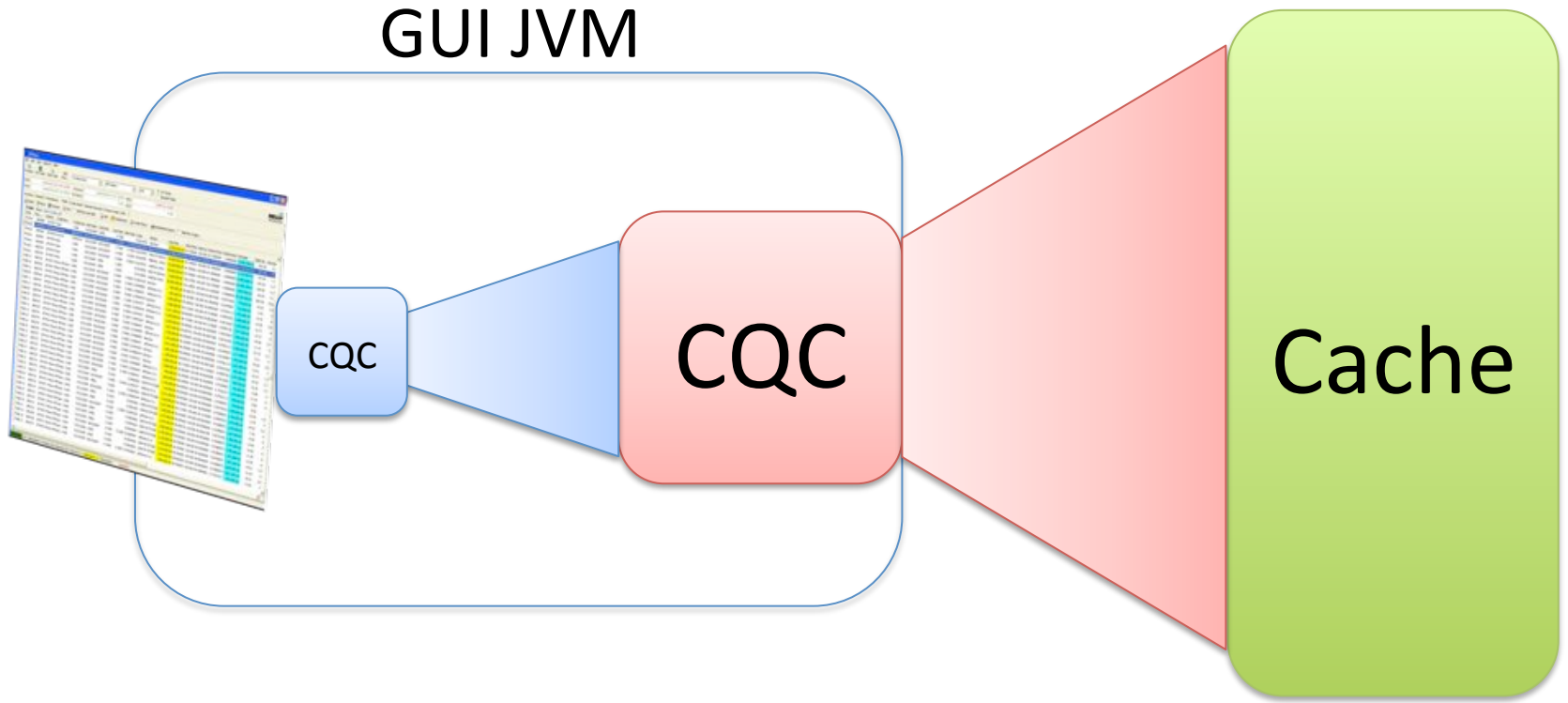


Easy Grid Implementation in GUIs

The screenshot displays a financial trading software interface with a grid of trade data. The interface includes a menu bar, a toolbar, and a main data table. The table columns include Fund, Pec..., Shell ID, Trade Type, Trade Code, Start Date, End Date, Last Rate, New Rate, Coup, Broker, Orig Price, Dirty Price, Hair Cut, Haircut Price, Trade Factor, Principal, and Daily Int. The data rows represent various trade transactions, such as Reverse and Money FR Repo, with associated rates and prices.

Fund	Pec...	Shell ID	Trade Type	Trade Code	Start Date	End Date	Last Rate	New Rate	Coup	Broker	Orig Price	Dirty Price	Hair Cut	Haircut Price	Trade Factor	Principal	Daily Int.	Accoun
Finance	3083805	3074927	Repo	LOAN	03/12/2009	OPEN	0.7500		912810P52	Barclays	20,000,000.00	94.799000	100.000	94.79900000	1.00000000	16,957,800.98	-394.95	-3.1
Finance	3083802	3074925	Reverse	BORROW	03/12/2009	03/31/2009	0.7500	0.7500	912810P52	Bank of America	10,000,000.00	94.799000	100.000	94.79900000	1.00000000	4,734,200.00	167.39	1.5
Finance	3083802	3074925	Reverse	BORROW	03/12/2009	03/31/2009	0.7500	0.7500	912810P52	Bank of America	10,000,000.00	94.799000	100.000	94.79900000	1.00000000	4,734,200.00	167.39	1.5
Finance	3083802	3074925	Reverse	BORROW	03/12/2009	03/31/2009	0.7500	0.7500	912810P52	Bank of America	10,000,000.00	94.799000	100.000	94.79900000	1.00000000	4,734,200.00	167.39	1.5
Finance	3083801	3074924	Repo	LOAN	03/12/2009	03/31/2009	0.4200	0.4200	912810P52	MERRILL LYNCH	22,000,000.00	94.799000	100.000	94.79900000	1.00000000	10,963,500.00	-243.29	-1.4
Finance	3083800	3074923	Repo	LOAN	03/12/2009	03/31/2009	0.5000	0.5000	912810P52	Goldman Sachs	30,000,000.00	94.799000	100.000	94.79900000	1.00000000	28,436,700.00	-394.95	-2.3
Finance	3082794	3074918	Repo	LOAN	02/13/2009	OPEN	1.8000		912810E26	Bank of America	2,000,000.00	150.375000	100.000	150.37500000	1.00000000	3,007,300.00	-83.54	-2.7
Finance	3082793	3074929	Repo	LOAN	02/13/2009	OPEN	1.8000		31371HE90	Goldman Sachs	20,000,000.00	101.625000	100.000	101.62500000	1.00000000	20,325,000.00	-564.58	-18.6
FUND 27	3082750	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31399VQ29	Barclays	904,000.00	96.086720	100.000	96.08672000	0.82331036	770,000.00	-19.25	-1.0
FUND 01	3082750	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396F9H5	Jefferies & Co.	645,000.00	96.303000	100.000	96.30300000	0.81404567	516,000.00	-4.01	-2
FUND 13	3082749	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396F9H5	Jefferies & Co.	1,853,000.00	96.589030	100.000	96.58903000	0.87461113	1,797,000.00	-12.42	-6
FUND 09	3082748	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31399VQW4	Barclays	1,500,000.00	96.712000	100.000	96.71200000	0.99957252	1,490,000.00	-37.00	-2.0
FUND 11	3082747	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31394WZ37	Jefferies & Co.	2,451,000.00	96.852000	100.000	96.85200000	0.90133203	2,196,000.00	-16.71	-9
FUND 23	3082746	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31399VQW4	Jefferies & Co.	1,099,000.00	96.712000	100.000	96.71200000	0.99957252	1,077,000.00	-12.27	-6
FUND 05	3082745	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31394WZ37	Barclays	2,200,000.00	96.852000	100.000	96.85200000	0.90133203	1,960,000.00	-49.00	-2.7
FUND 20	3082744	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31399VQ29	Jefferies & Co.	1,821,000.00	96.086720	100.000	96.08672000	0.82331036	1,470,000.00	-11.43	-6
FUND 21	3082743	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31396F9H5	Barclays	1,750,000.00	96.589030	100.000	96.58903000	0.87461113	1,596,000.00	-37.70	-2.1
FUND 14	3082742	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396F9H5	Jefferies & Co.	1,463,000.00	96.303000	100.000	96.30300000	0.81404567	1,371,000.00	-9.11	-5
FUND 19	3082740	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31396F9H5	Barclays	1,611,000.00	96.303000	100.000	96.30300000	0.81404567	1,490,000.00	-36.25	-2.0
FUND 30	3082740	3074914	Money FR Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396F9H5	Jefferies & Co.	1,981,000.00	96.589030	100.000	96.58903000	0.87461113	1,821,000.00	-12.41	-7
FUND 23	3082739	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31399VQ29	Barclays	1,200,000.00	96.086720	100.000	96.08672000	0.82331036	969,000.00	-24.23	-1.3
FUND 23	3082738	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31399VQW4	Barclays	2,000,000.00	96.712000	100.000	96.71200000	0.99957252	1,975,000.00	-49.33	-2.7
FUND 30	3082737	3074913	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31394WZ37	Barclays	1,000,000.00	96.852000	100.000	96.85200000	0.90133203	990,000.00	-22.25	-1.2
FUND 27	3082723	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396F9H5	Jefferies & Co.	909,000.00	96.303000	100.000	96.30300000	0.81404567	726,000.00	-6.88	-3
FUND 22	3082722	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396G263	Jefferies & Co.	1,500,000.00	96.440540	100.000	96.44054000	0.79101017	1,339,000.00	-11.04	-6
FUND 19	3082721	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396F9H5	Jefferies & Co.	1,190,000.00	96.589030	100.000	96.58903000	0.87461113	990,000.00	-8.95	-5
FUND 14	3082720	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	149,000.00	96.554000	100.000	96.55400000	0.91888074	134,000.00	-0.82	-
FUND 01	3082719	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31394WZ37	Jefferies & Co.	3,941,000.00	96.852000	100.000	96.85200000	0.90133203	3,494,000.00	-32.90	-1.8
FUND 22	3082718	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	1,220,000.00	96.554000	100.000	96.55400000	0.91888074	1,134,000.00	-6.75	-3
FUND 14	3082717	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31399VQW4	Jefferies & Co.	4,241,000.00	96.712000	100.000	96.71200000	0.99957252	3,924,000.00	-99.52	-2.2
FUND 25	3082716	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	1,246,000.00	96.554000	100.000	96.55400000	0.91888074	1,220,000.00	-7.46	-4
FUND 11	3082715	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	1,797,000.00	96.554000	100.000	96.55400000	0.91888074	1,627,000.00	-9.94	-6
FUND 15	3082714	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31399VQ29	Jefferies & Co.	1,615,000.00	96.086720	100.000	96.08672000	0.82331036	1,485,000.00	-13.84	-7
FUND 07	3082713	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	1,966,000.00	96.554000	100.000	96.55400000	0.91888074	1,771,000.00	-10.82	-6
FUND 02	3082712	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	2,159,000.00	96.554000	100.000	96.55400000	0.91888074	1,905,000.00	-11.95	-6
FUND 30	3082711	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	2,170,000.00	96.554000	100.000	96.55400000	0.91888074	1,972,000.00	-12.05	-6
FUND 11	3082710	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396F9H5	Jefferies & Co.	2,362,000.00	96.303000	100.000	96.30300000	0.81404567	3,907,000.00	-18.01	-1.0
FUND 18	3082709	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	2,119,000.00	96.554000	100.000	96.55400000	0.91888074	2,028,000.00	-12.28	-6
FUND 29	3082708	3074911	Money FR Repo	LOAN	01/21/2009	OPEN	0.2200		31396H2H6	CANTOR FITZGERALD	3,326,000.00	96.554000	100.000	96.55400000	0.91888074	3,162,000.00	-18.41	-1.0
FUND 13	3082707	3074910	Money FR Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396G263	Jefferies & Co.	1,856,000.00	96.440540	100.000	96.44054000	0.79101017	1,466,000.00	-13.68	-7

CQCs on a CQC

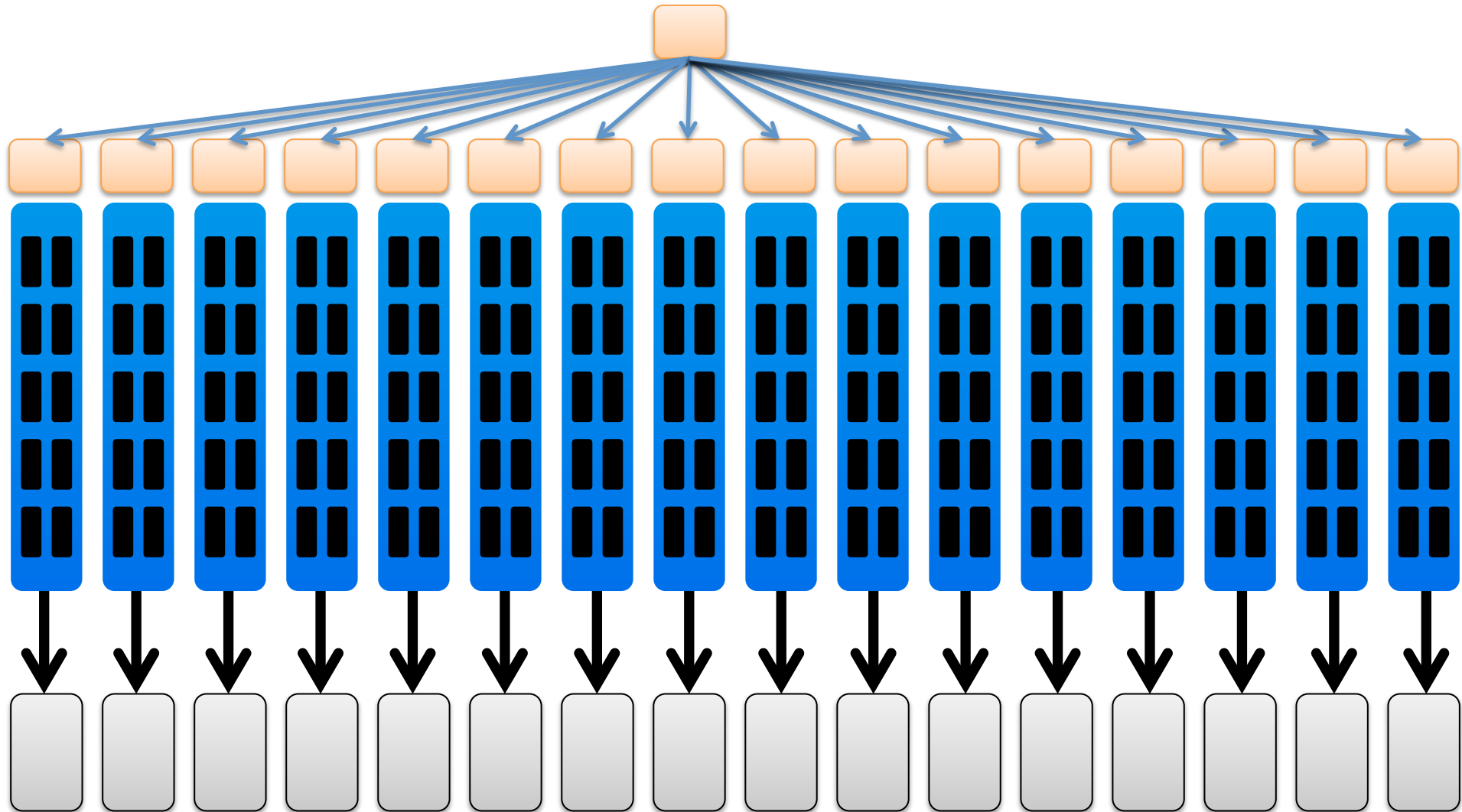


Define this in config

How do you release quickly to a
Coherence cluster?

Rolling Restart?

Disk-Persist



Final Thoughts

Data is the most important
commodity that you have

Keep it safe

Use a Partition Listener

Have Proactive Monitoring of
Memory

Version your Objects

Thanks

Slides & related articles available at:

<http://www.benstopford.com>