

# 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

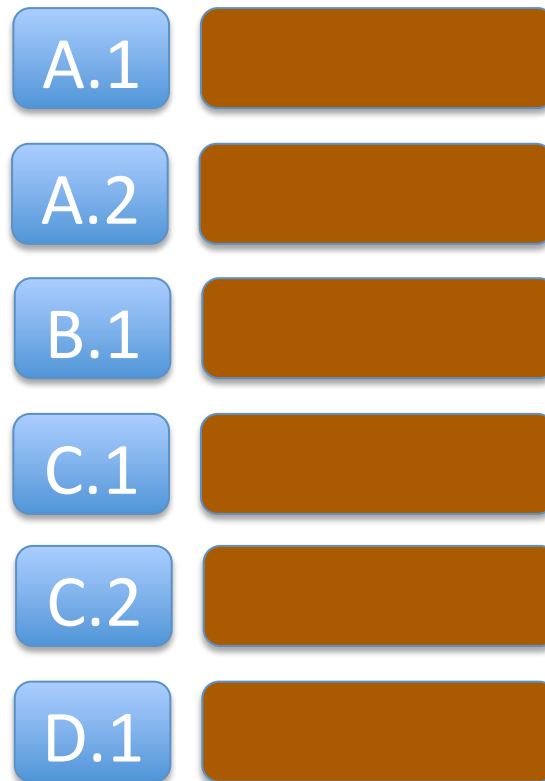
# Why do we care?

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

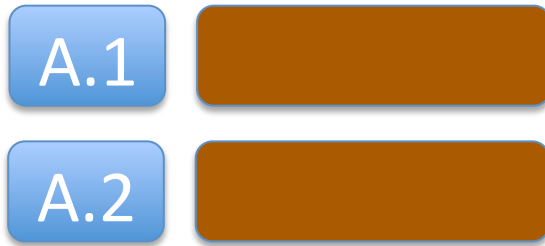
Versioned  
Cache





# Versioning your Objects

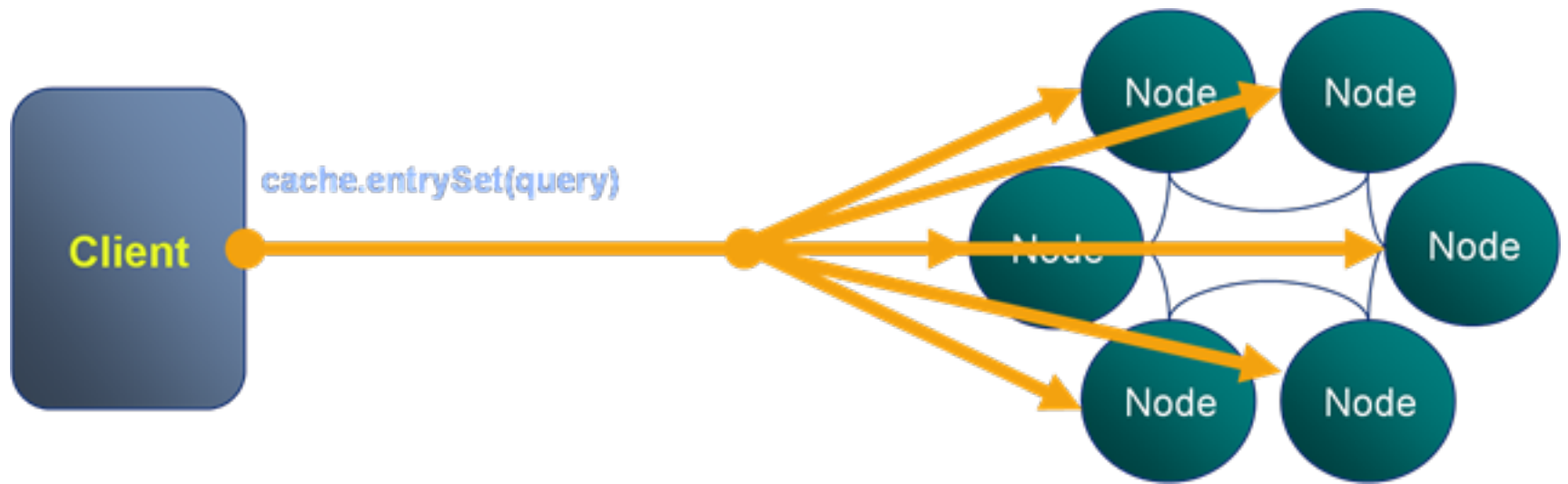
Cache



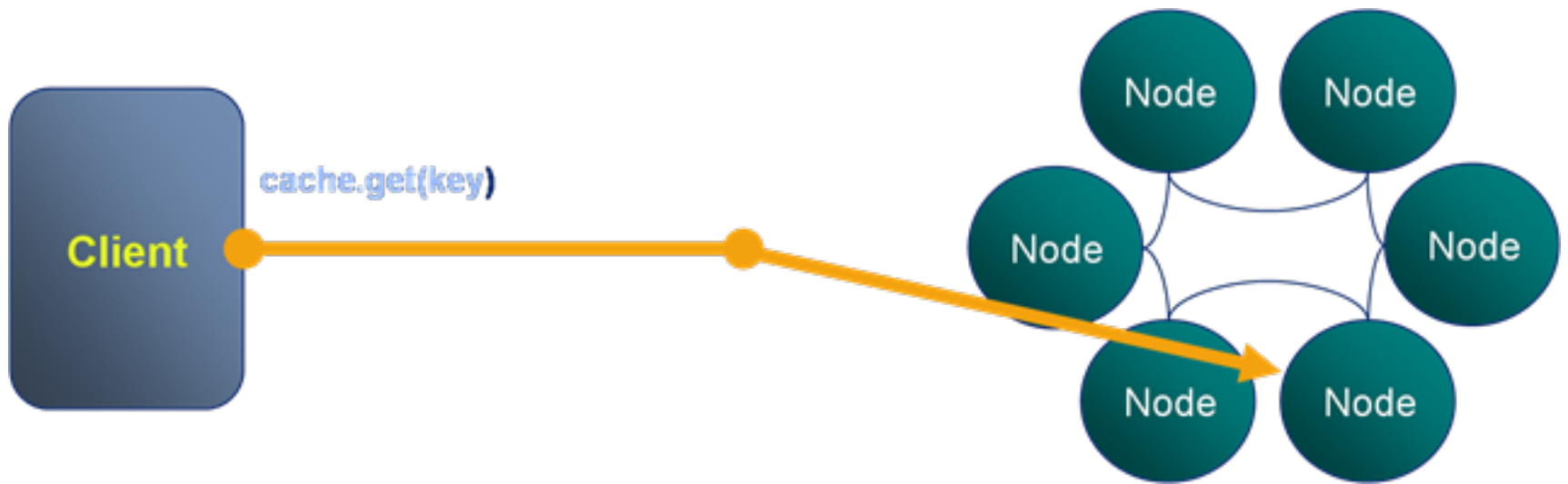
Coherence Trigger

New Version = Old Version + 1 ??

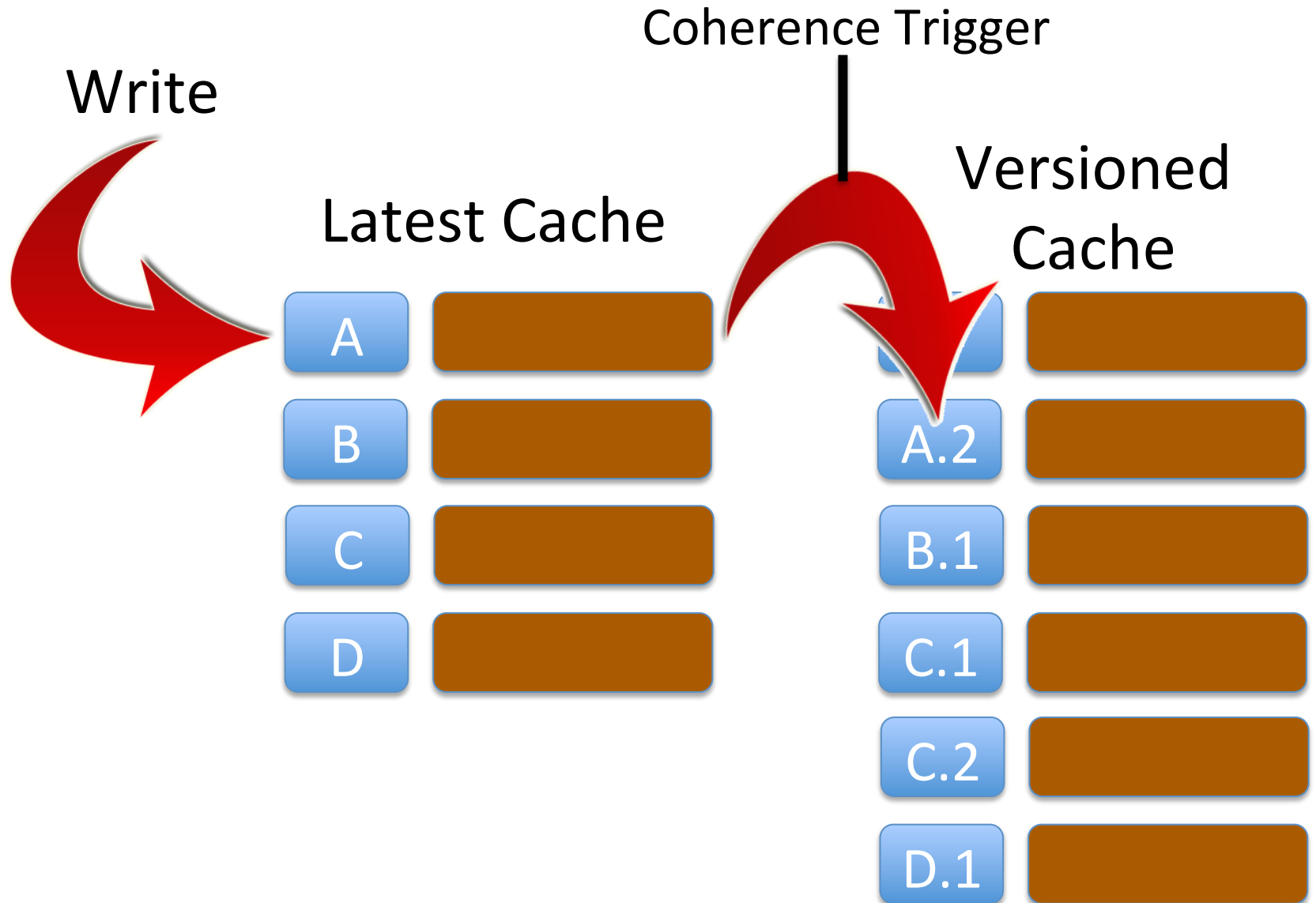
# Running a Coherence Filter



# Using Key-Based Access



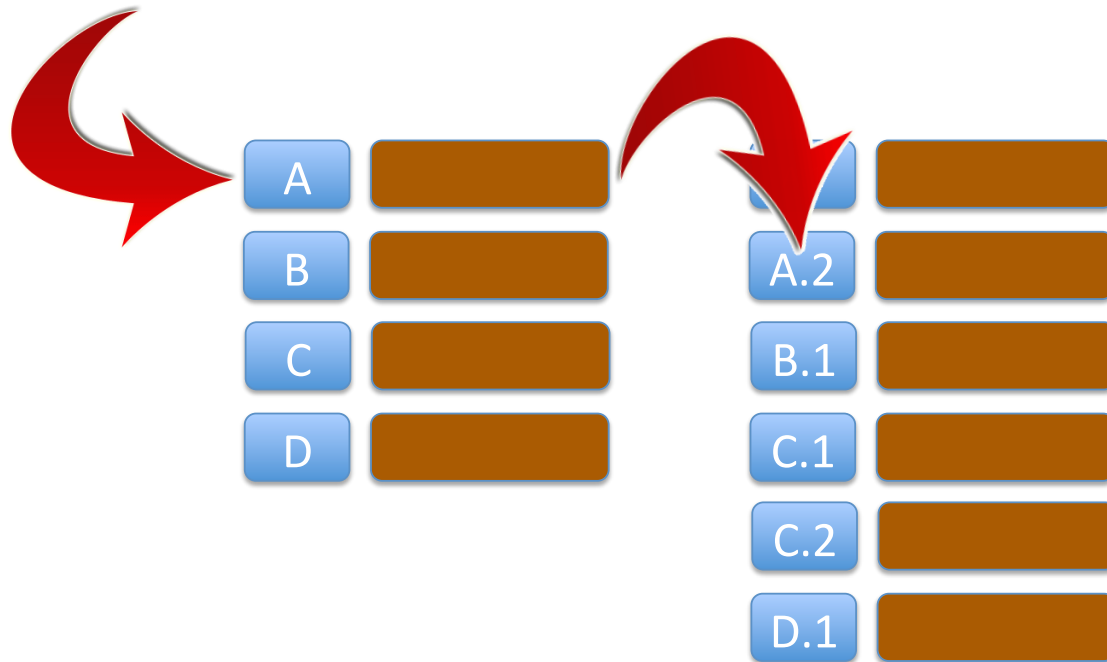
# Latest / Versioned Pattern



# Latest / Versioned Pattern

Latest Cache Key = [Business Key]

Versioned Cache Key = [Business Key][Version]



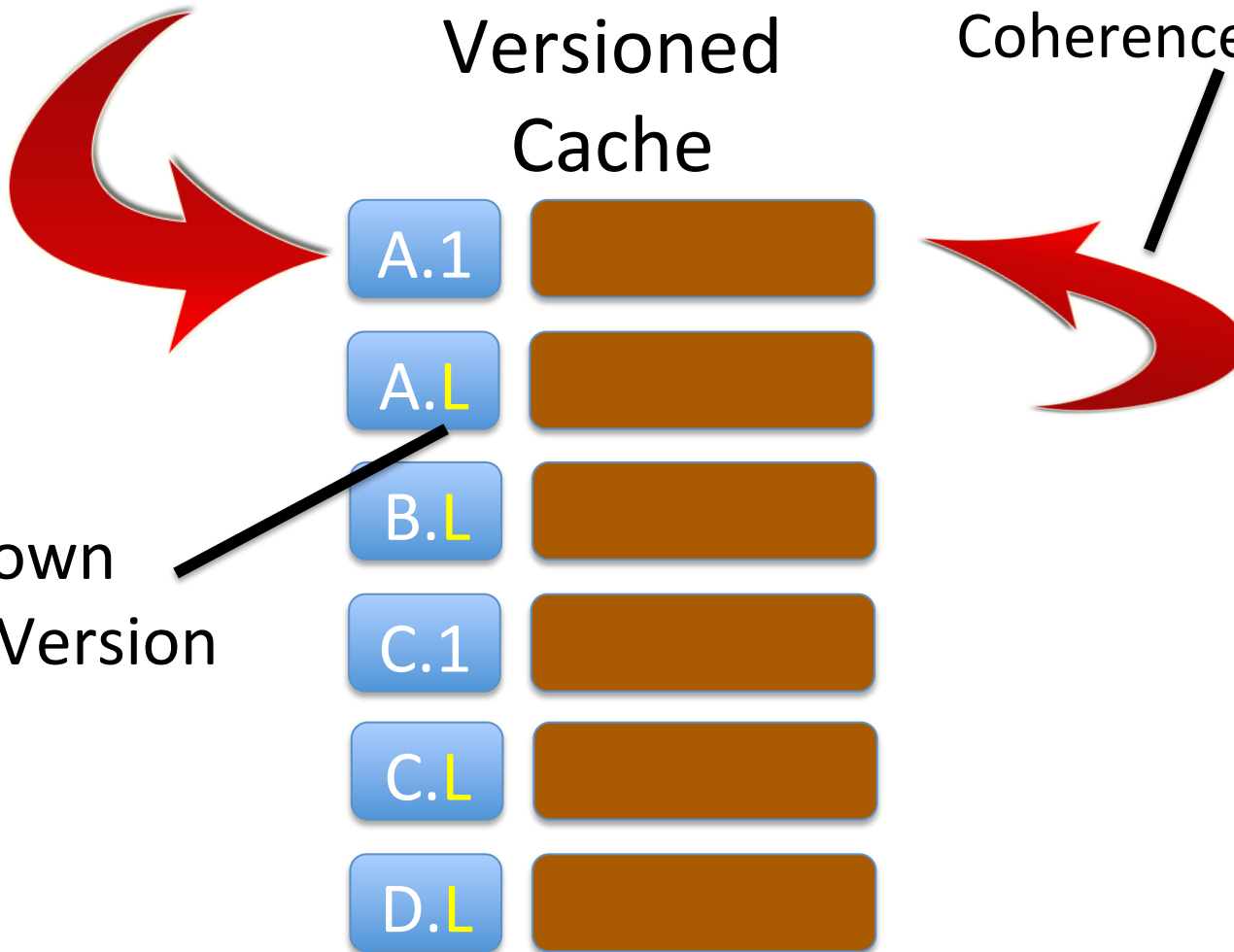
Suffers from data duplication

# Latest Marker Pattern

Write

Versioned  
Cache

Coherence Trigger



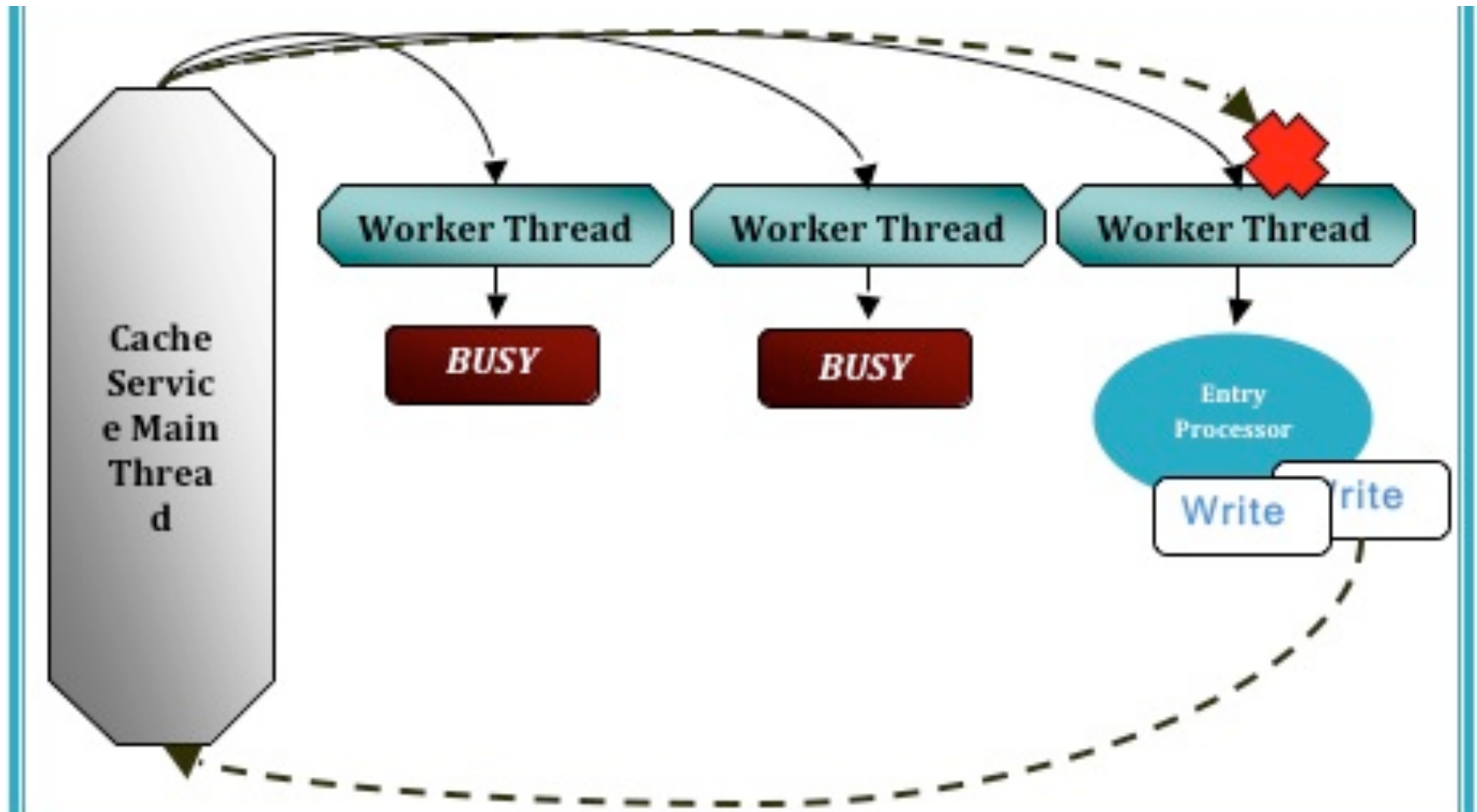
Well Known  
Marker Version

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();

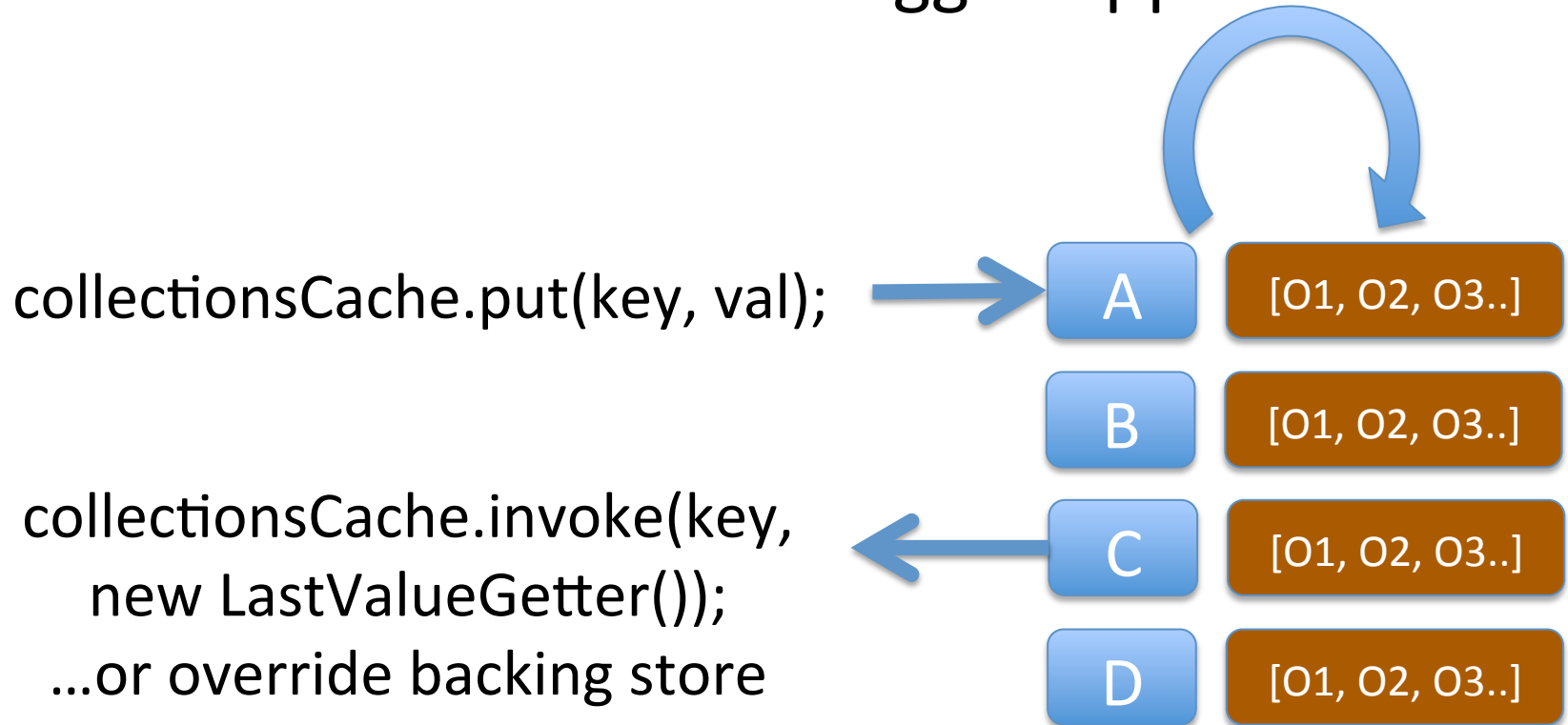
    BinaryEntry binary = (BinaryEntry)entry;
    Binary binaryValue = binaryEntry.getBinaryValue();

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

A third approach

# The Collections Cache

Trigger Appends to Collection



CollectionsCache

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

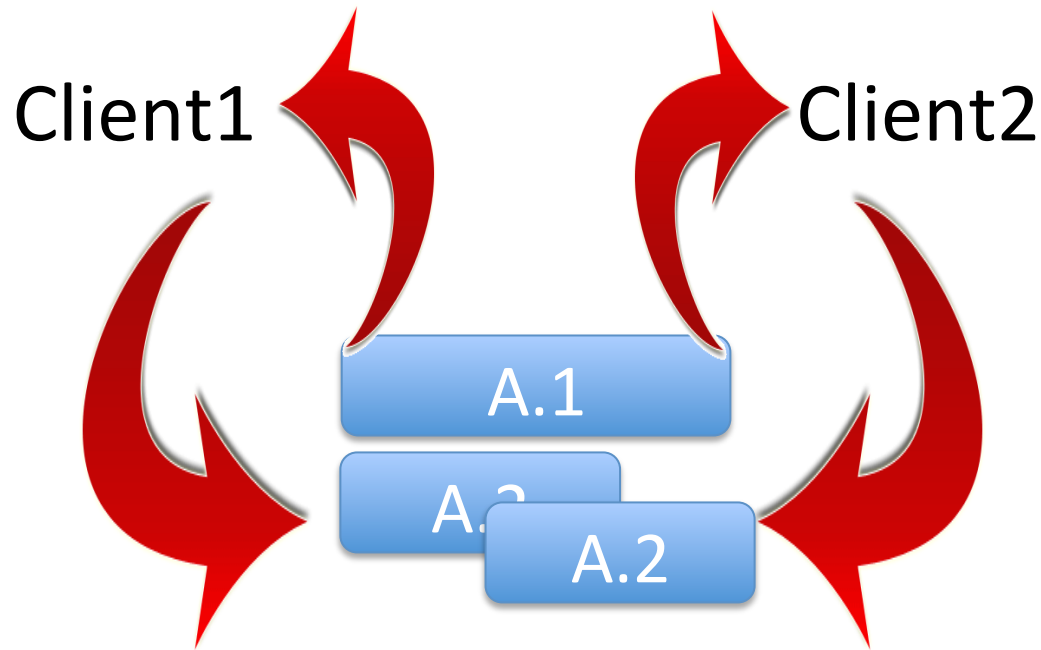


Coherence Trigger

New Version = Old Version + 1 ??

# Concurrent Object Update

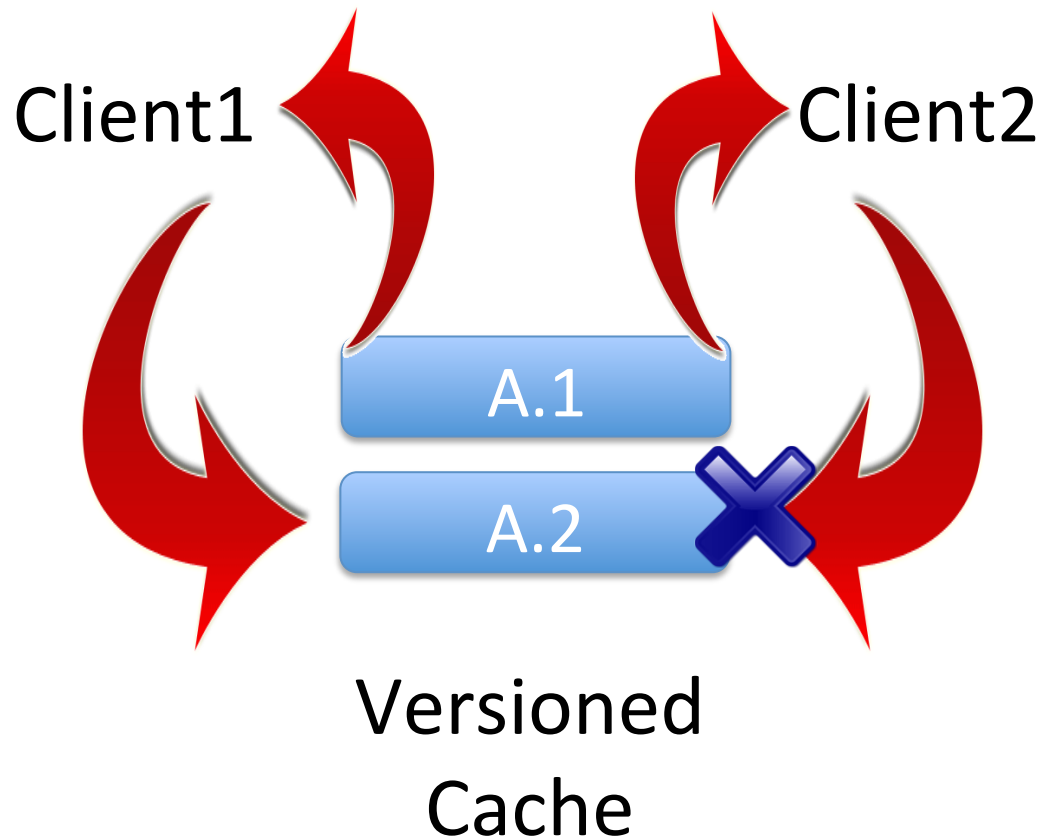
(2 Clients update the same object at the same time)





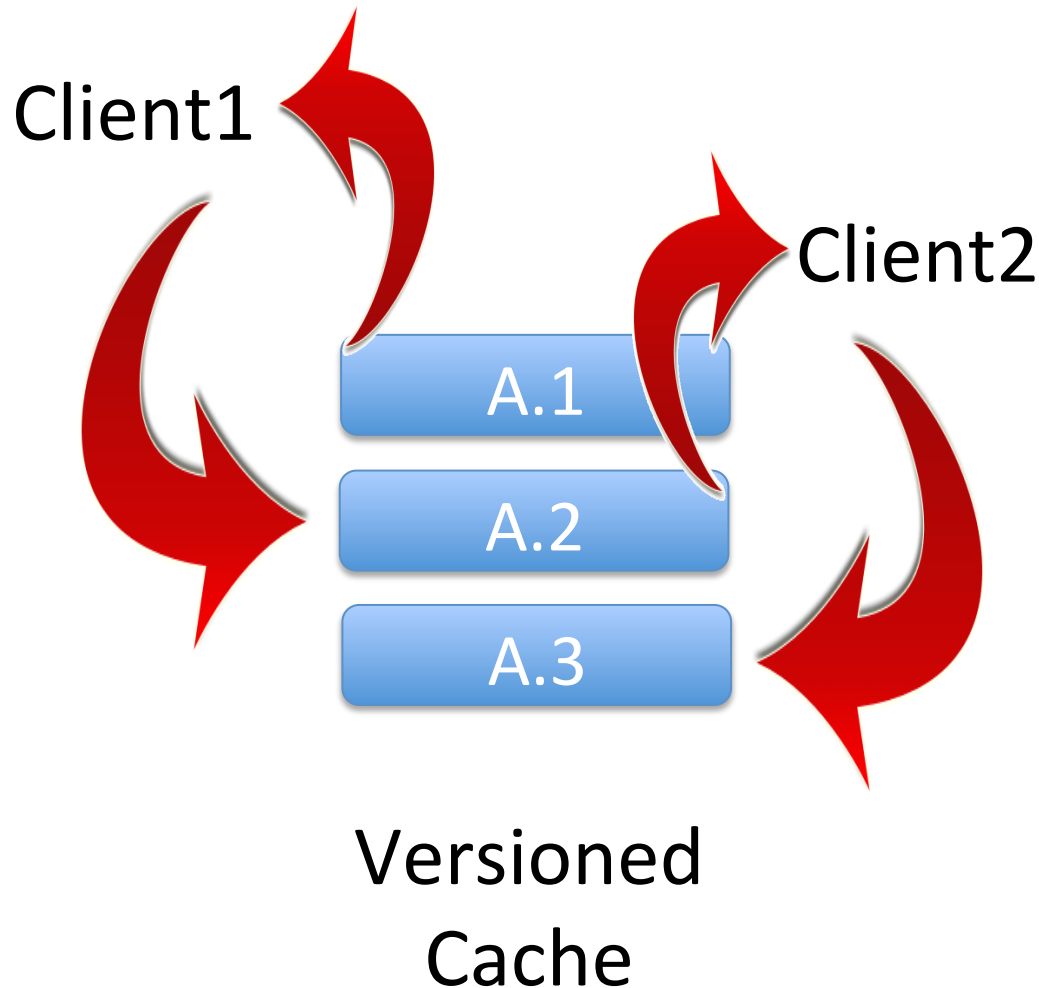
# Concurrent Object Update

(Client2 fails to update dirty object)

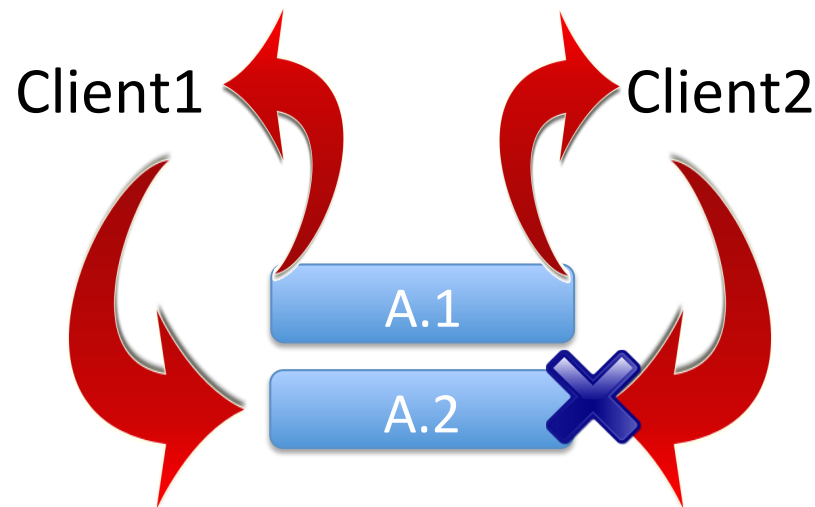


# Concurrent Object Update

(Client 2 updates clean object)



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

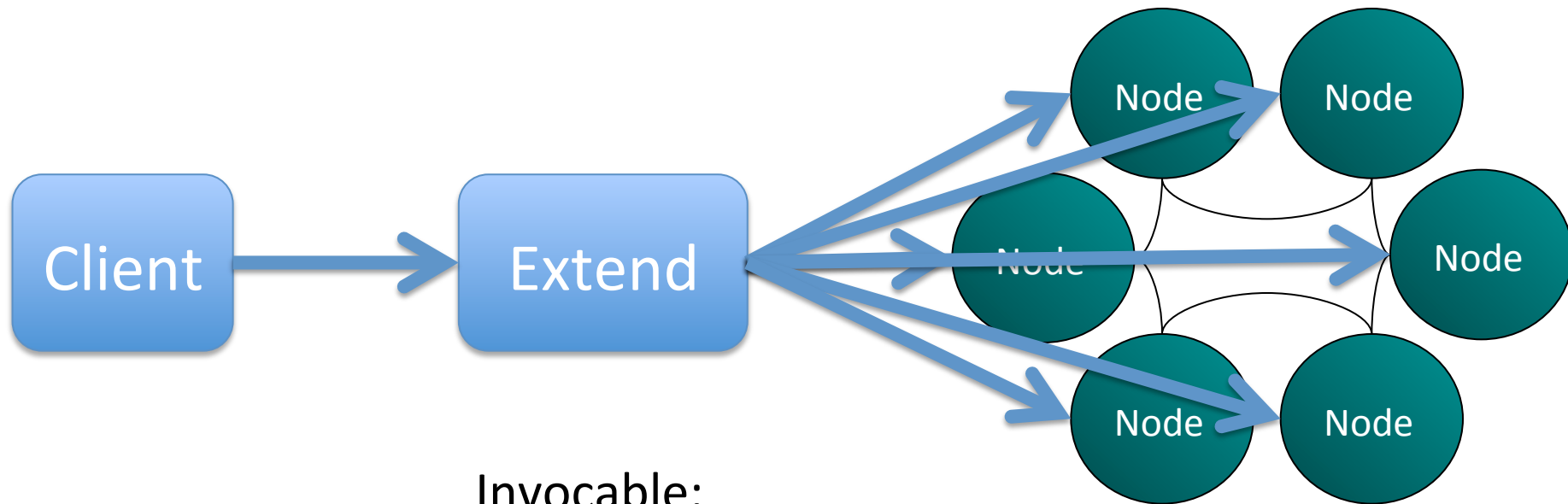


What's going to happen if we are  
using putAll?

# 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:

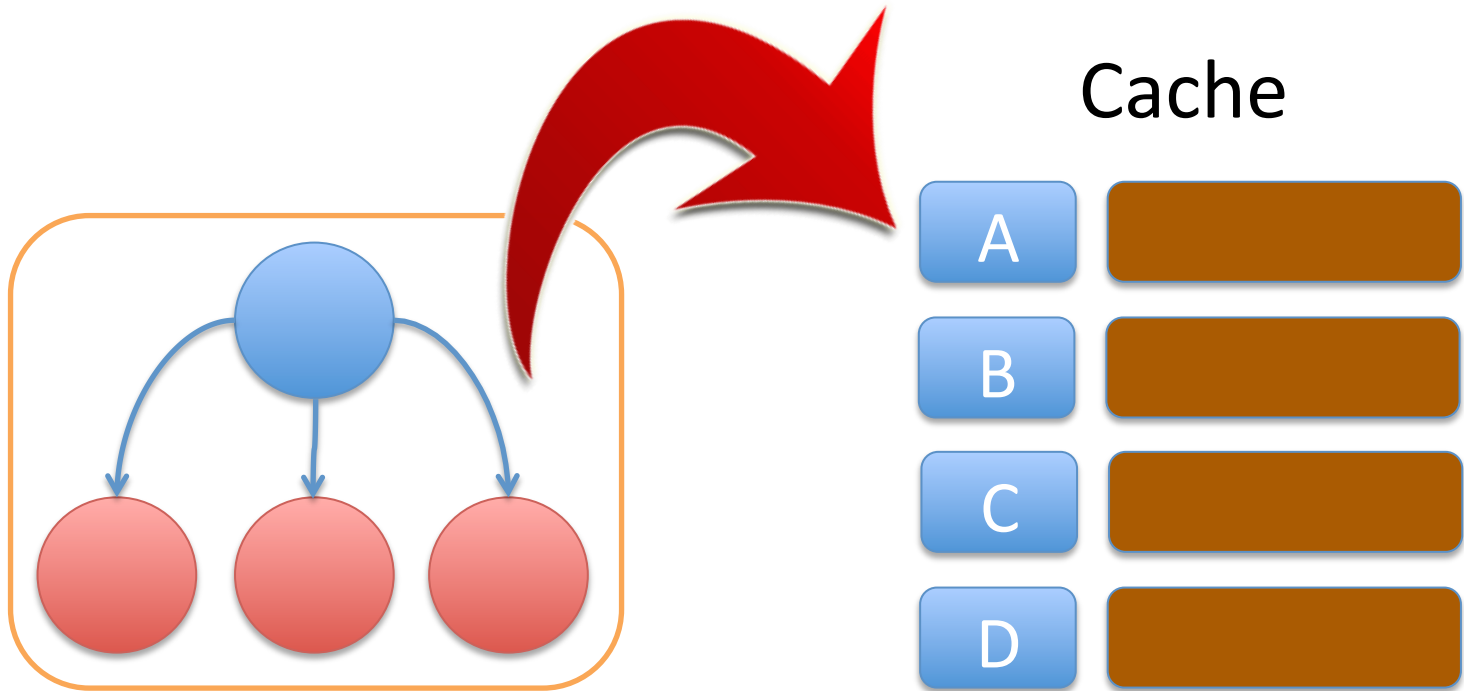
- Write entries to backing map (we use an EP for this)

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

# Synthesising Transactionality

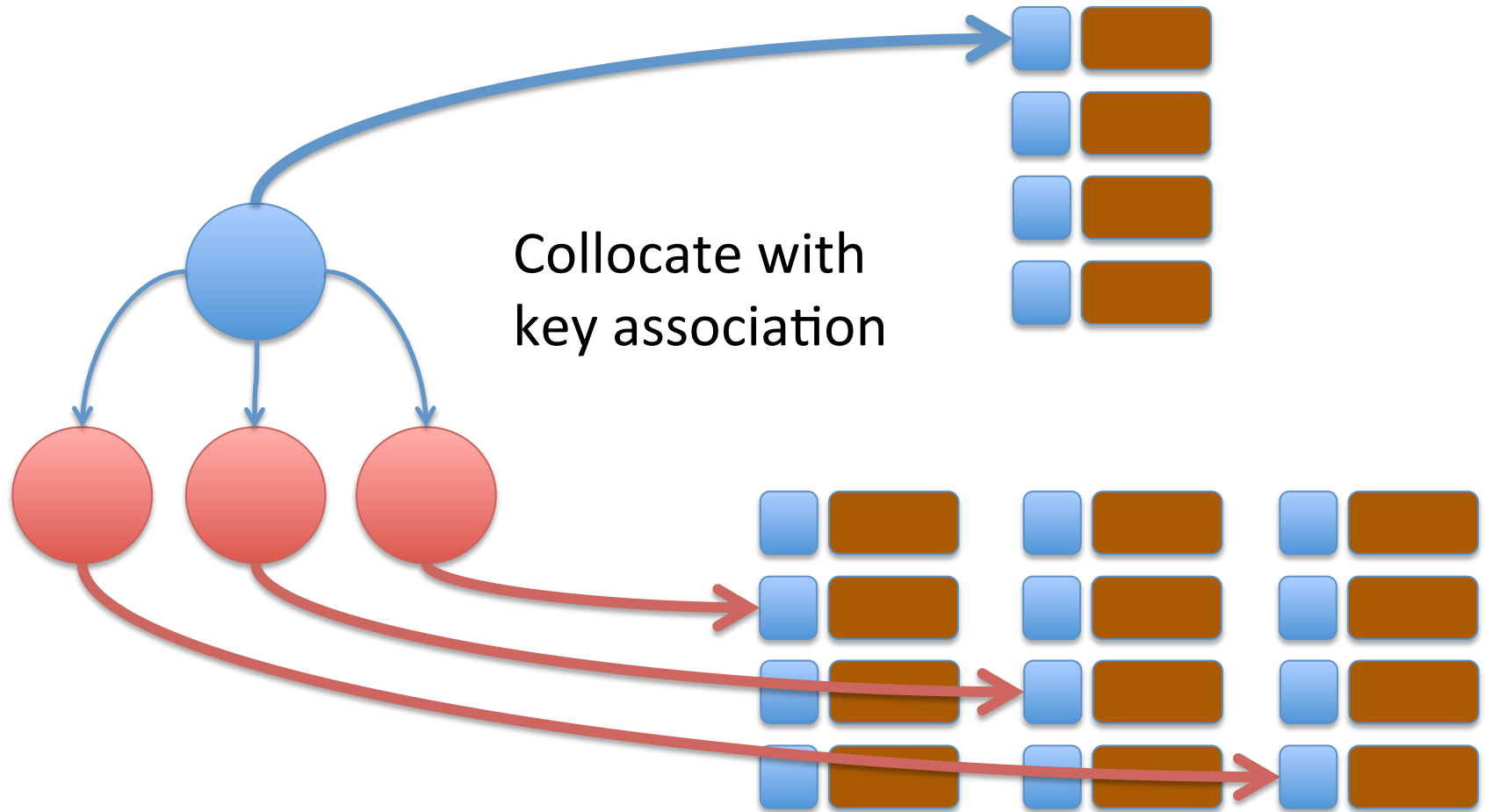


# The Fat Object Method

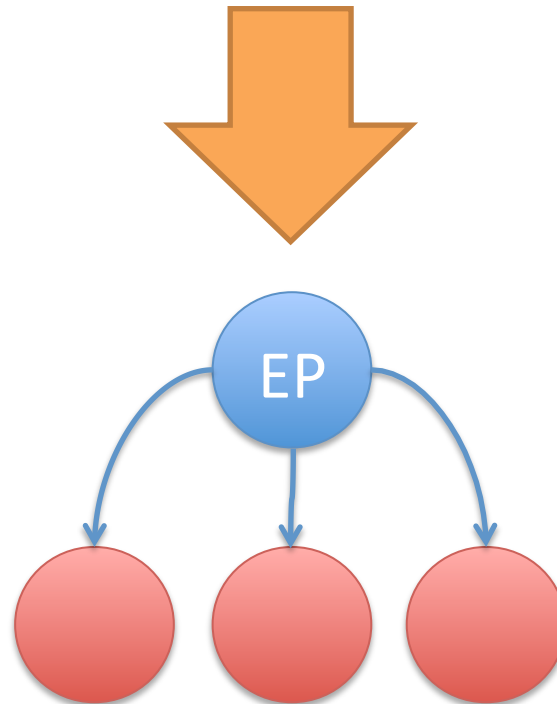


# The Single Entry Point Method

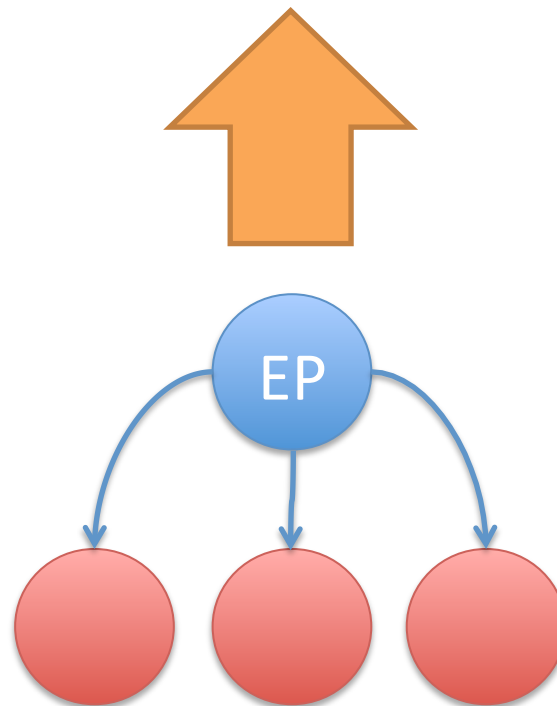
(objects are stored separately)



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

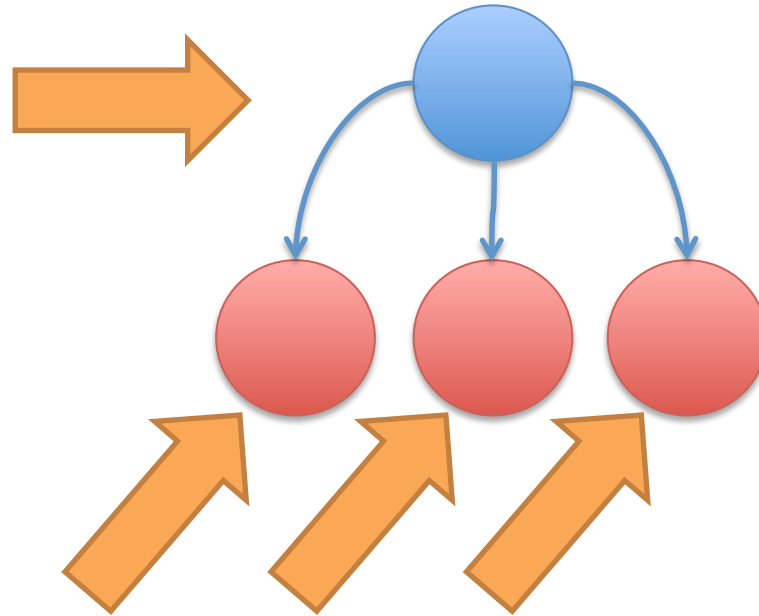


All **reads** synchronize on the  
primary object.



# Writing Orphaned Objects

Write read  
entry point  
object last



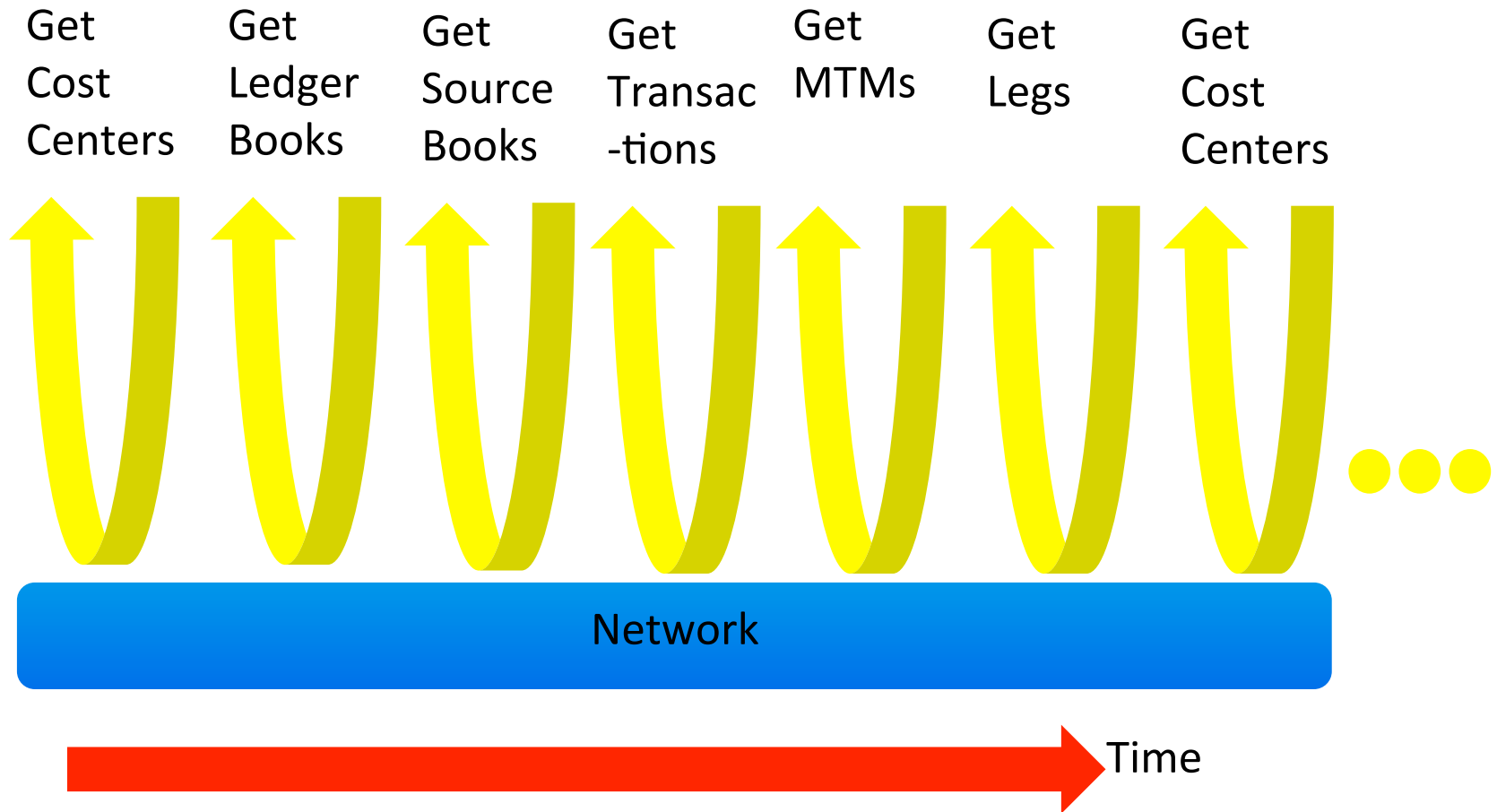
Write orphaned objects first

This mechanism is subtly flawed

Reading several objects as an  
atomic unit

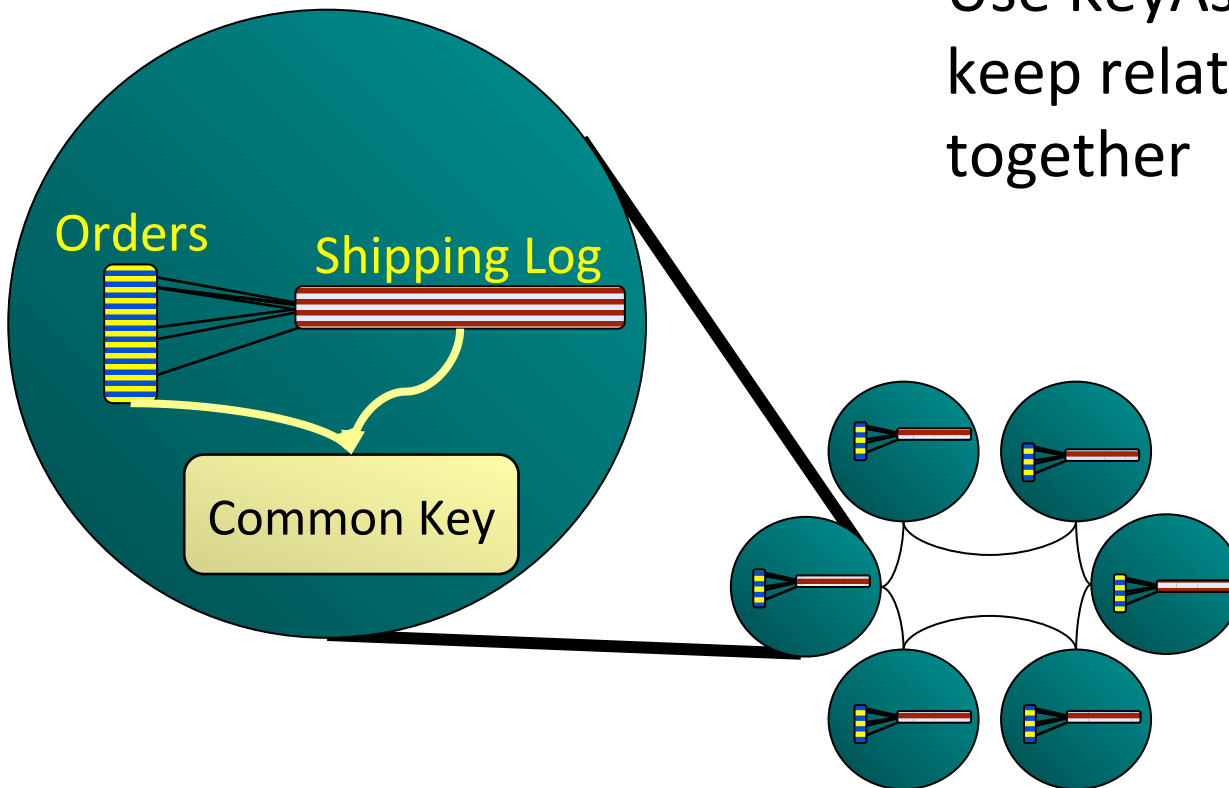
aka Joins

# The trivial approach to joins



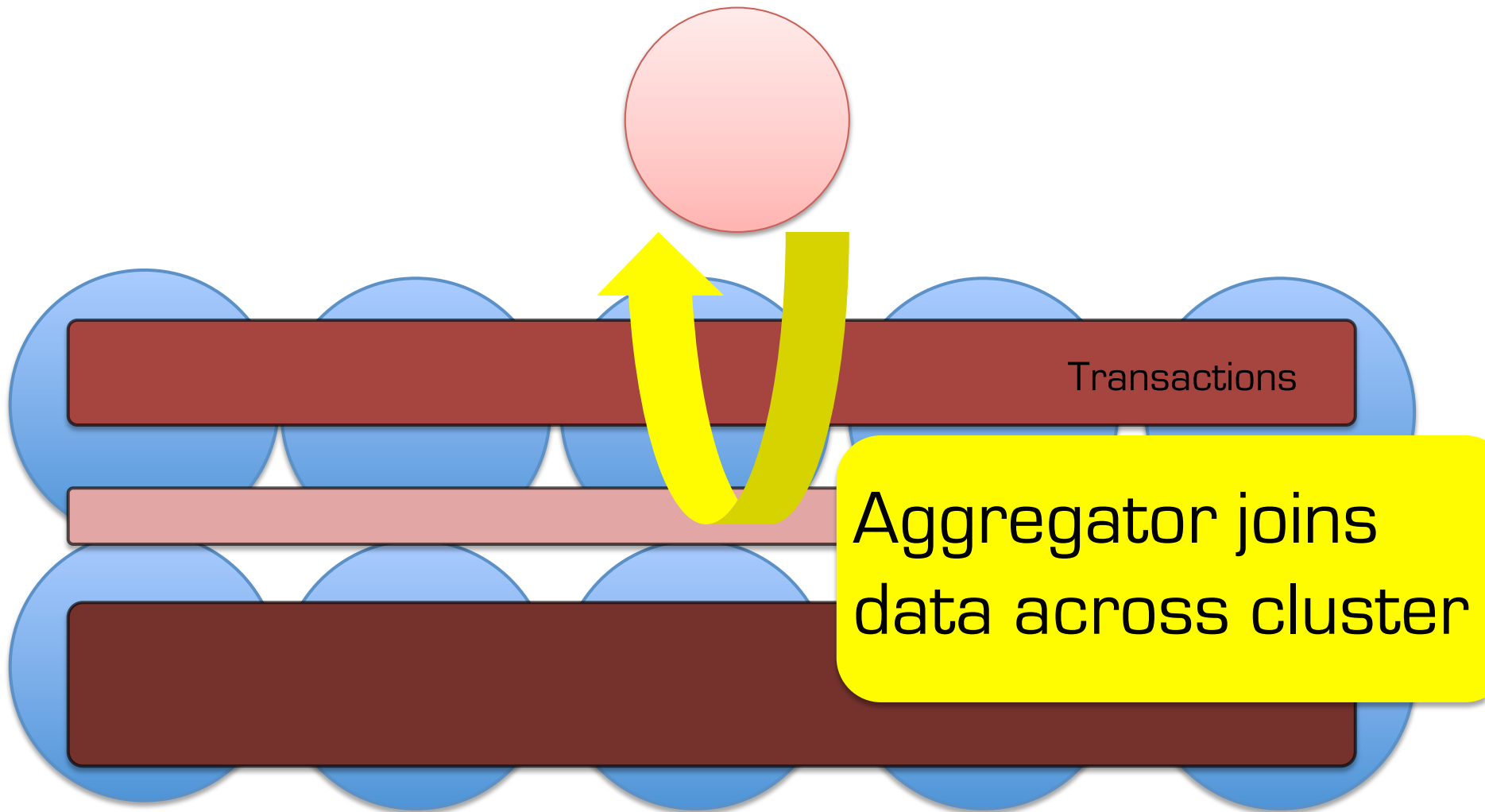
# Server Side, Sharded Joins

Use KeyAssociation to keep related entities together





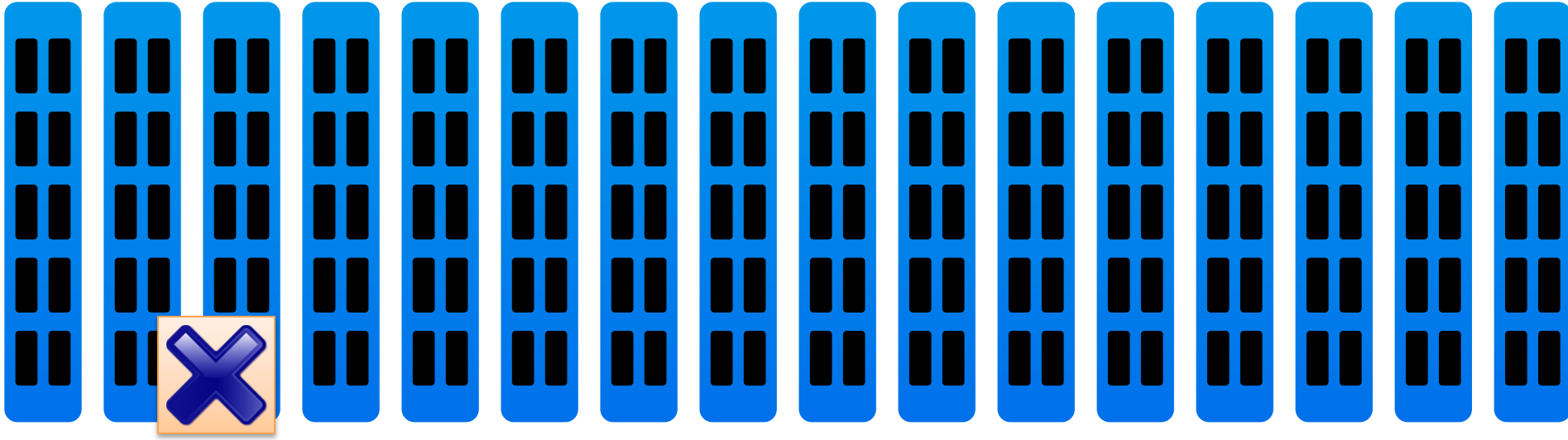
# Server Side, Sharded Joins



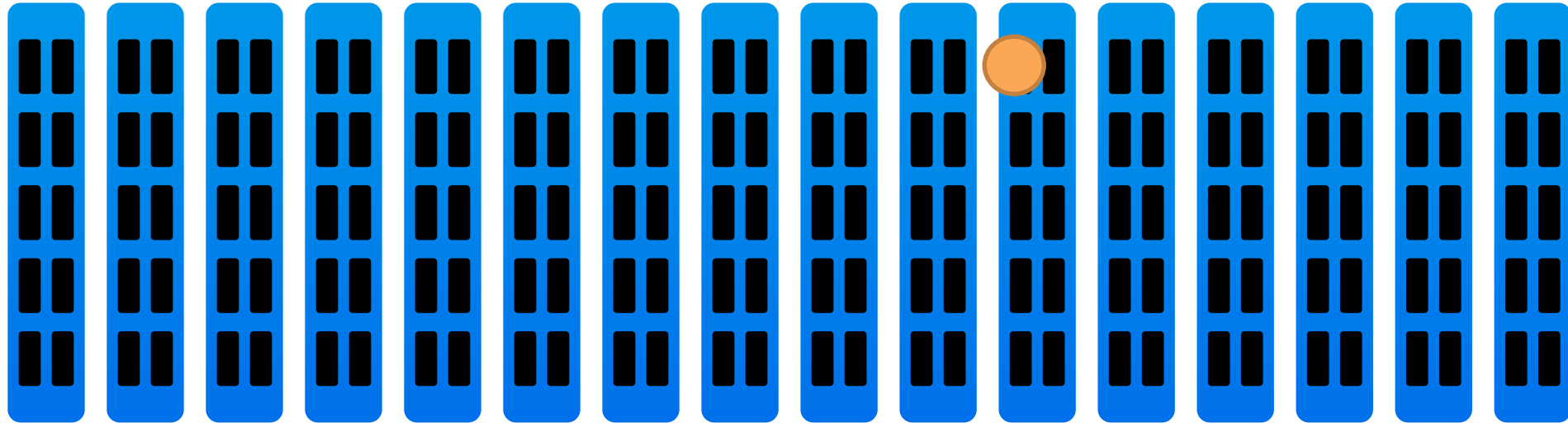
So we have a set of mechanisms for  
reading and writing groups of  
related objects.

# 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

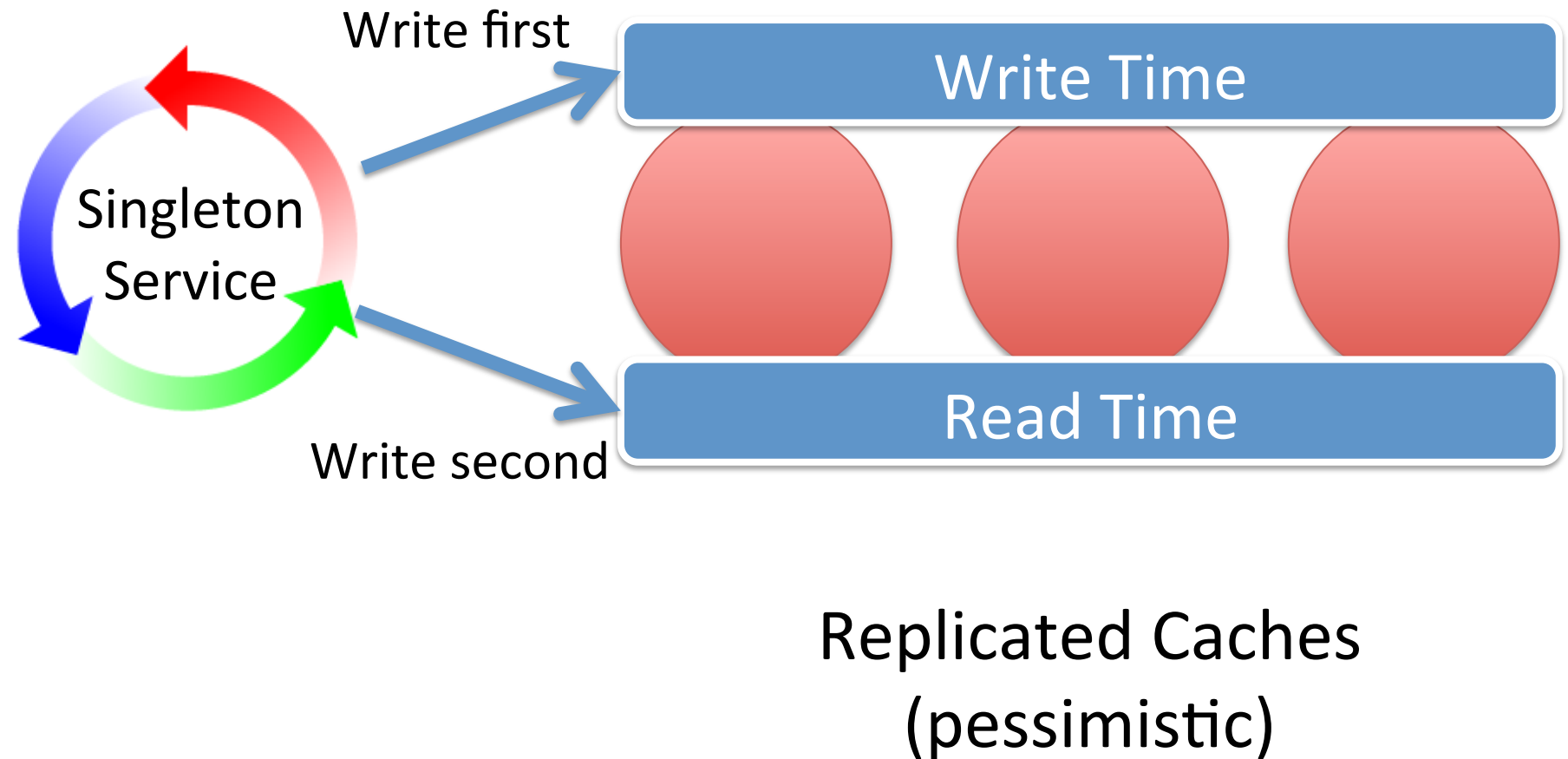
System  
Time

Where does the System Time  
come from?

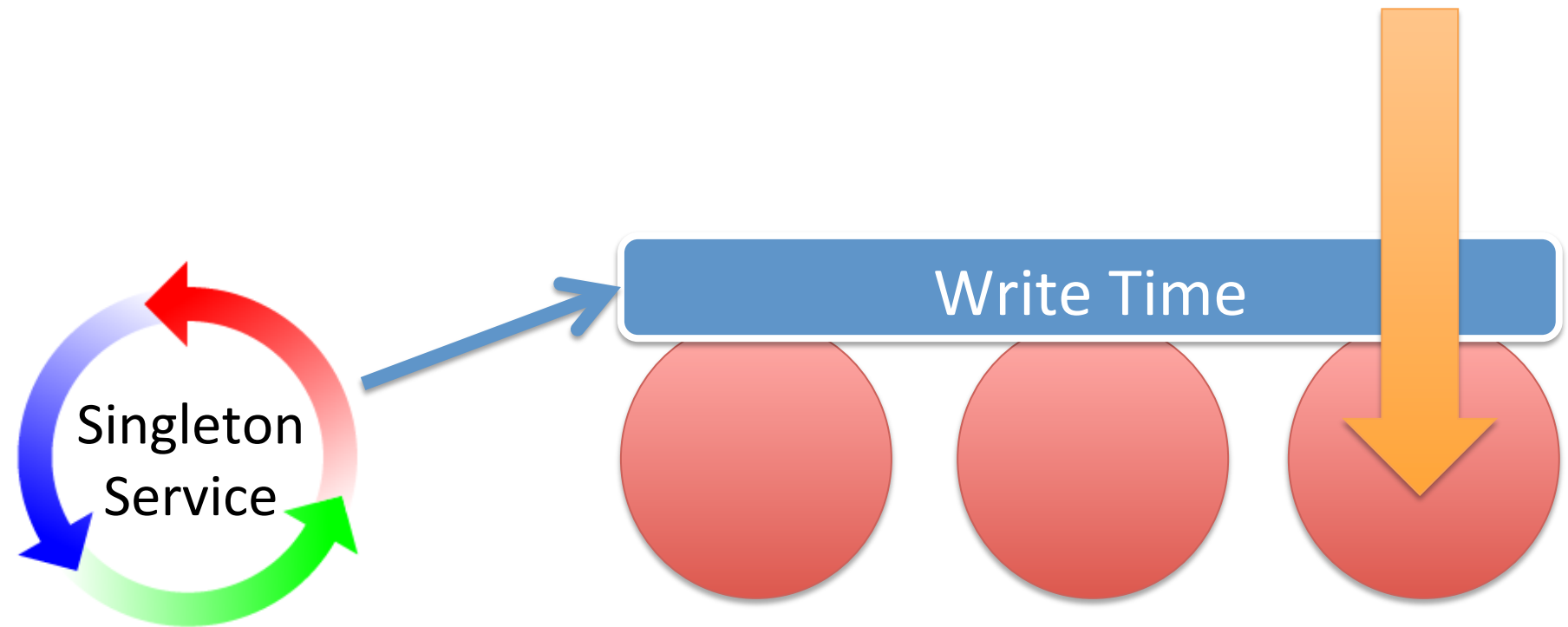
You can't use the  
`System.currentTimeMillis()` in a  
distributed environment!

You need a cluster synchronised  
clock

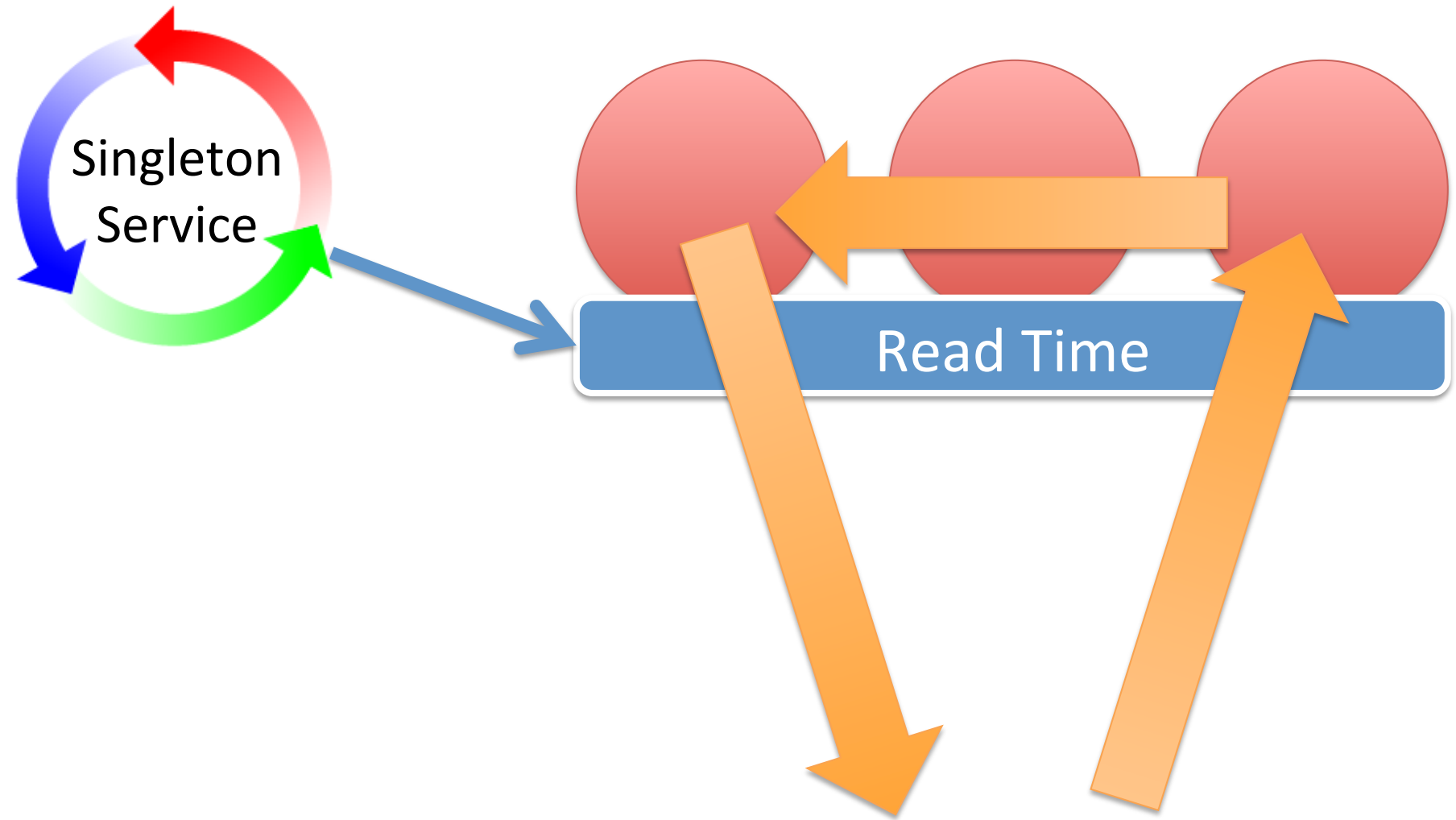
# Repeatable Time: A guaranteed Tick



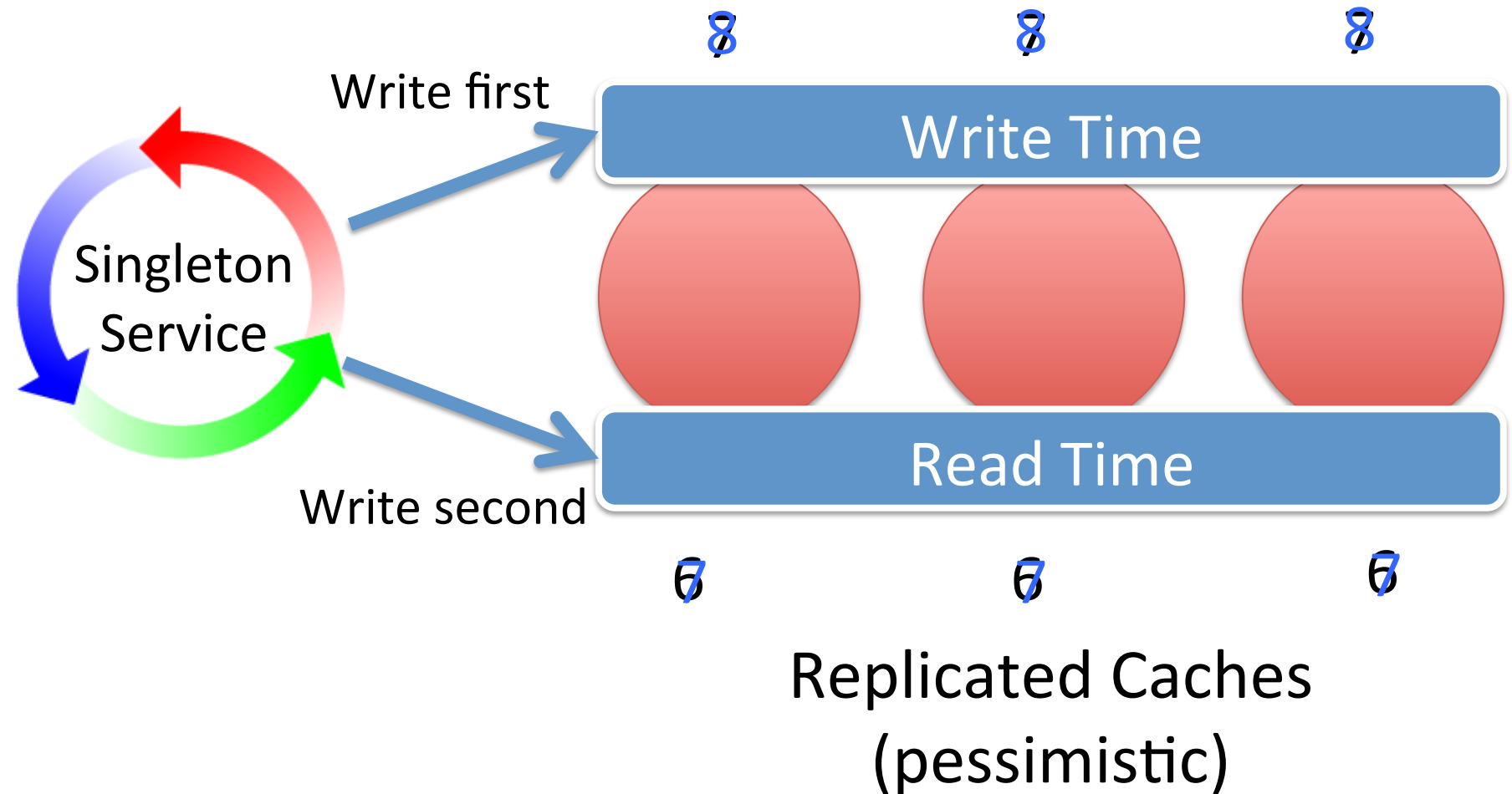
As we add objects we timestamp them  
with Write Time



# When we read objects we use Read Time

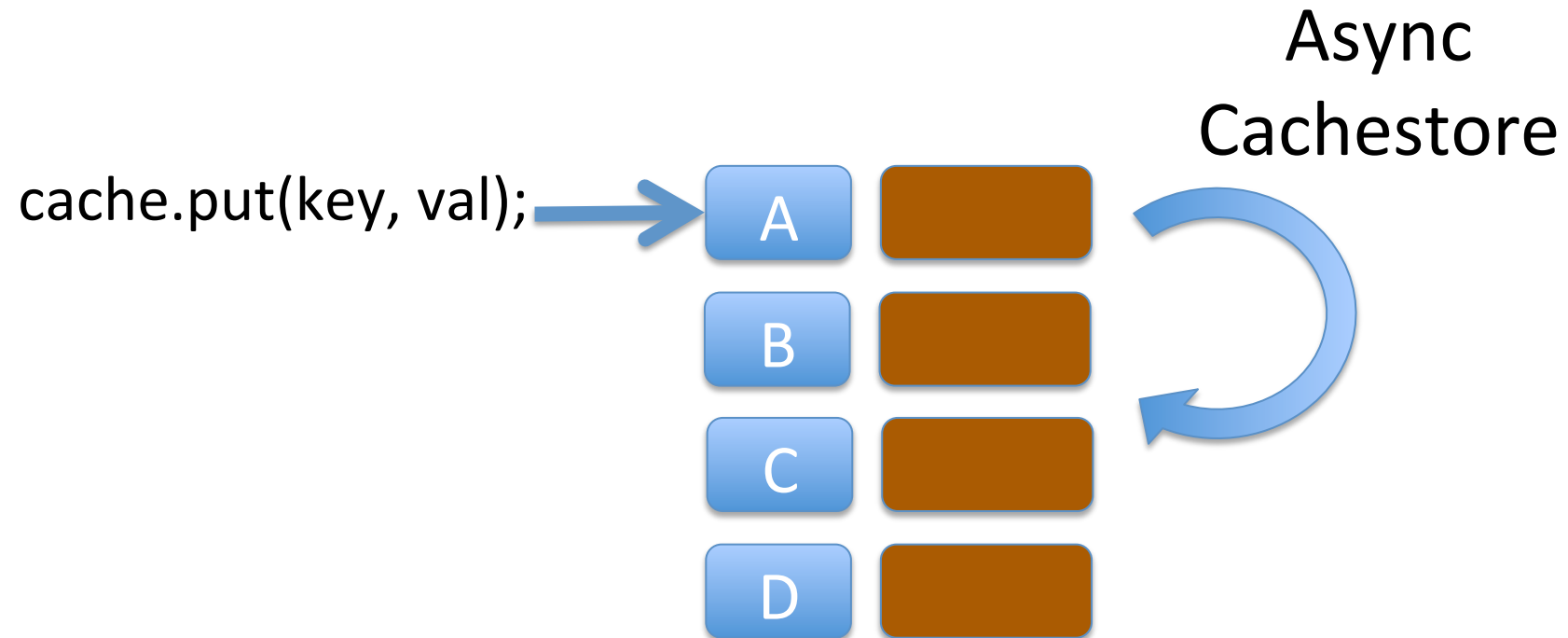


# Repeatable Time: A guaranteed Tick

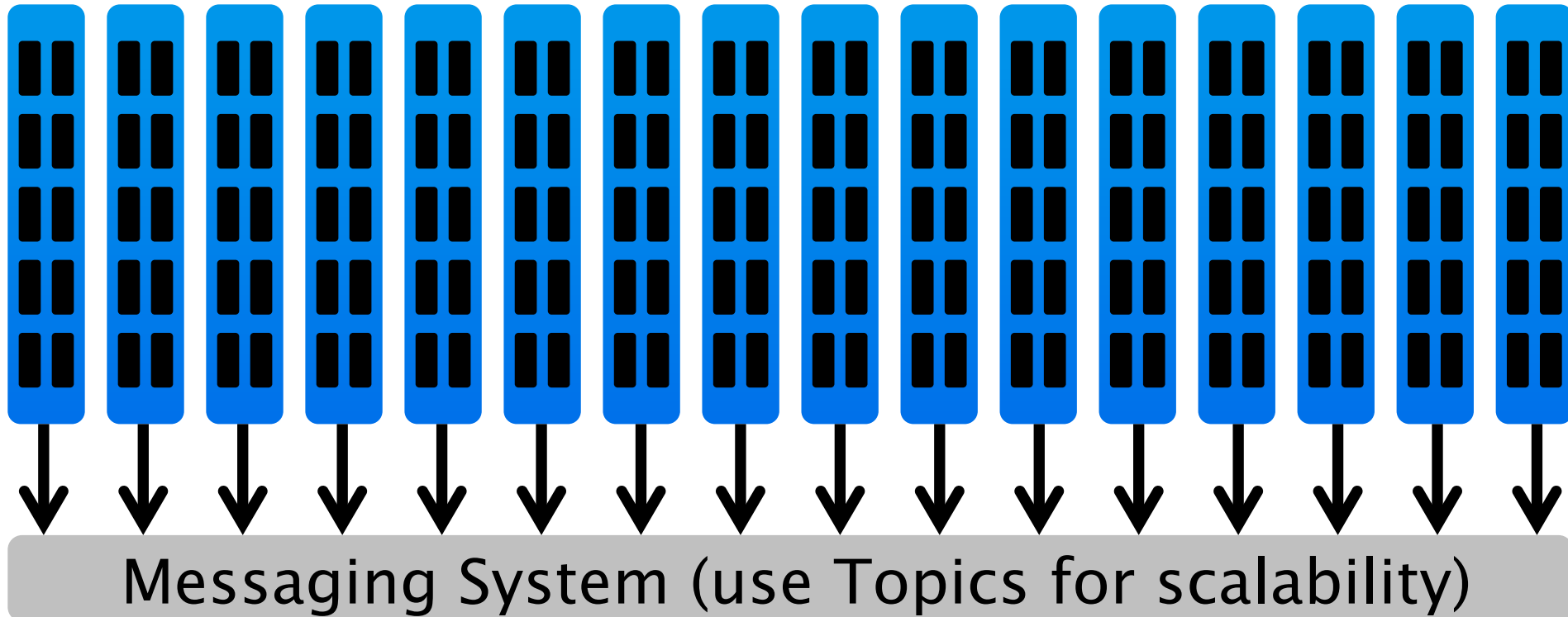




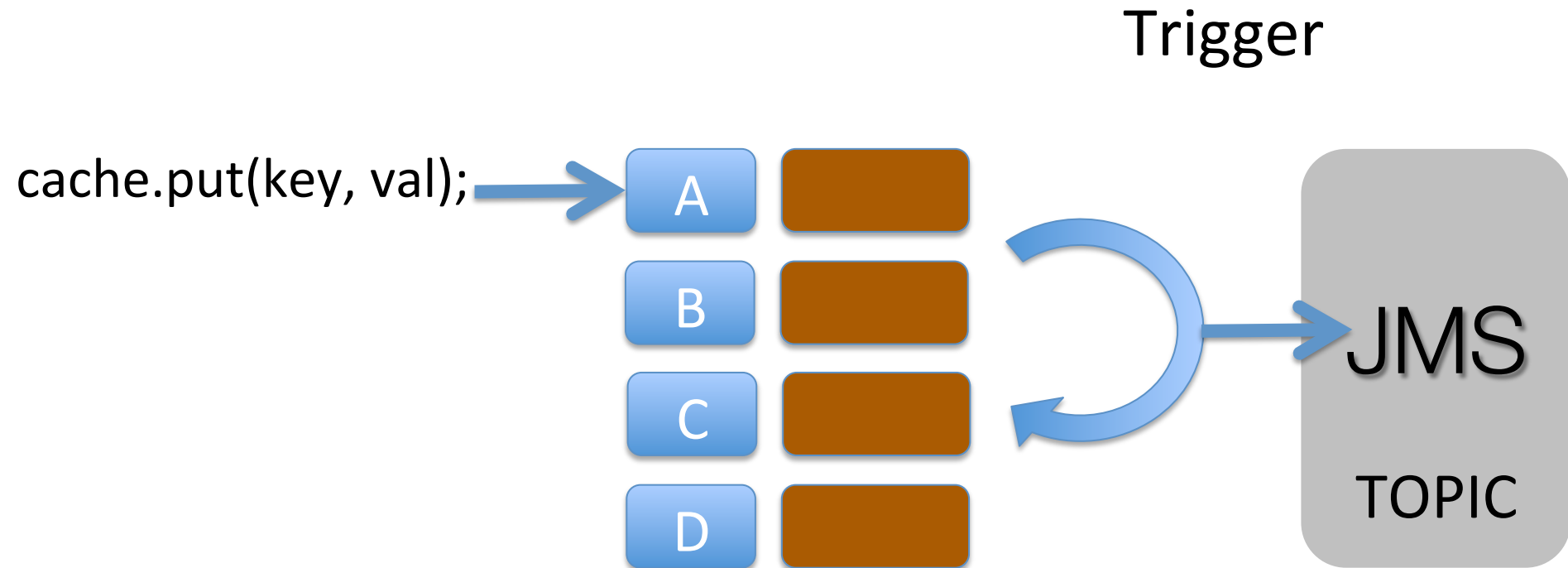
# Event Based Processing



# Messaging as a System of Record



# Messaging as a System of Record



# Easy Grid Implementation in GUIs

FileEditViewReportsHelp

ProductsPar TradeShell TradeFind...Trading EntityAll LedgersUSDGC TotalsSpecials Totals

Cash:-3,253,831,011.49 (USD)Finance:3,280,515,637.15 (USD)P&L:-3,085.63 (USD)Long:3,400,515,637.15 (USD)Tri-Party:0.00GCF:0.00

InventorySpecialsCounterpartyTradesTrade ShellsInterest Rate RiskProduct TotalsMisc

DetailsViewsScreenerFindHide Buys and SellsEditTradeActionsTrade HistorySpreadsheet ImportHide Firm Trades

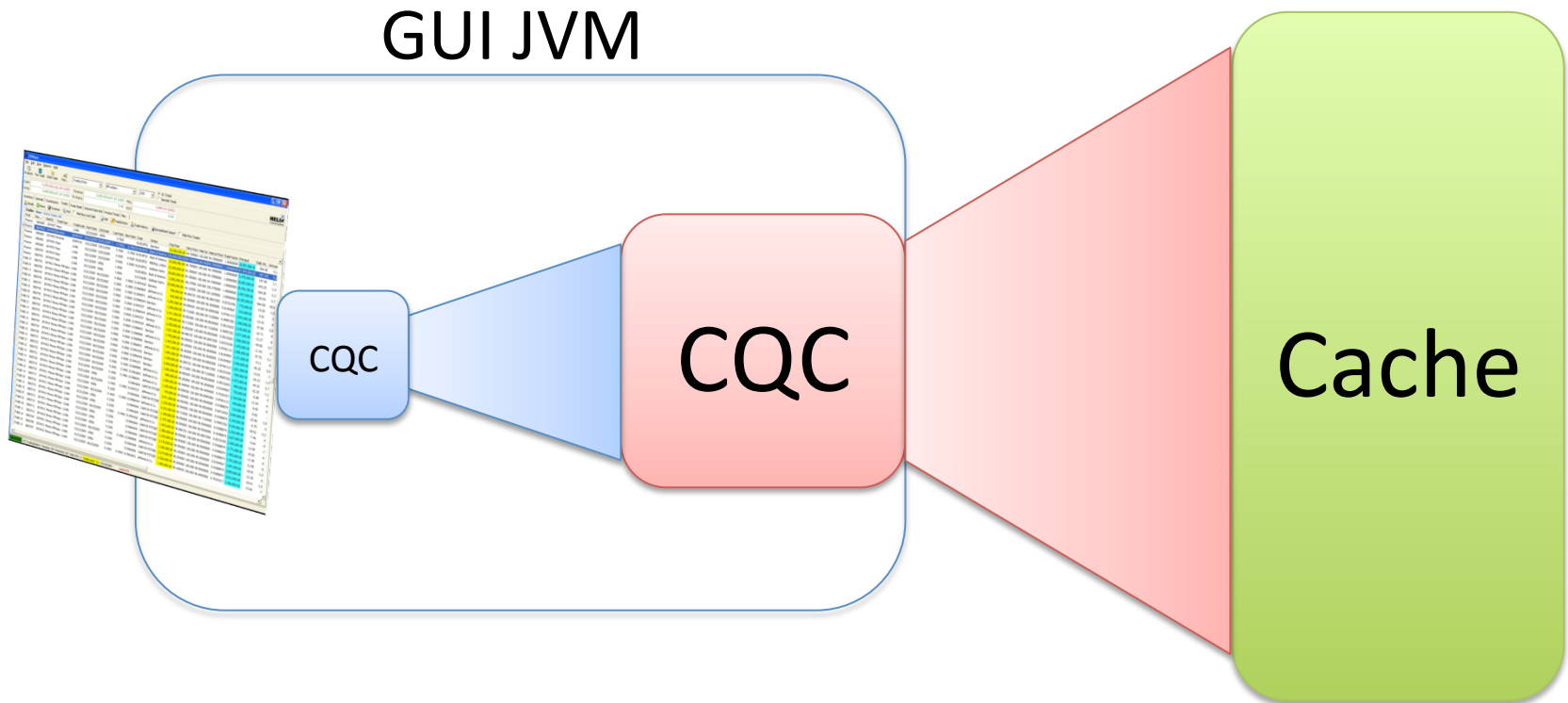
TradesView: Active Trades / All

Fund	Plec...	Shell ID	TradeType	TradeCode	Start Date	End Date	Last Rate	New Rate	Cusip	Broker	Orig Face	Dirty Price	HairCut	Haircut Price	Trade Factor	Principal	Daily Int.	Accrued
Finance	3883805	3074937	Repo	LOAN	03/10/2009	OPEN	0.7500		912810FS2	Barclays	20,000,000.00	94.789000	100.000	94.78900000	1.00000000	18,957,800.00	-394.95	-3.1
Finance	3883803	3074935	Reverse	BORROW	03/12/2009	03/31/2009	0.7500	0.7500	912810FS2	Bank of America	50,000,000.00	94.789000	100.000	94.78900000	1.00000000	47,394,500.00	987.39	5.9
Finance	3883802	3074935	Reverse	BORROW	03/12/2009	03/31/2009	0.7500	0.7500	912810FS2	Bank of America	10,000,000.00	94.789000	100.000	94.78900000	1.00000000	9,478,900.00	197.48	1.1
Finance	3883801	3074934	Repo	LOAN	03/12/2009	03/23/2009	0.4200	0.4200	912810FS2	MERRILL LYNCH	22,000,000.00	94.789000	100.000	94.78900000	1.00000000	20,853,580.00	-243.29	-1.4
Finance	3883800	3074933	Repo	LOAN	03/12/2009	03/23/2009	0.5000	0.5000	912810FS2	Goldman Sachs	30,000,000.00	94.789000	100.000	94.78900000	1.00000000	28,436,700.00	-394.95	-2.3
Finance	3883794	3074918	Repo	LOAN	02/13/2009	OPEN	1.0000		912810ED6	Bank of America	2,000,000.00	150.37500	100.000	150.375000	1.00000000	3,007,500.00	-83.54	-2.7
Finance	3883793	3074929	Repo	LOAN	02/13/2009	OPEN	1.0000		31371HDP0	Goldman Sachs	20,000,000.00	101.62500	100.000	101.625000	1.00000000	20,325,000.00	-564.58	-18.6
FUND 27	3883751	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31395V4Q9	Barclays	954,000.00	98.086720	100.000	98.08672000	0.82331036	770,000.00	-19.25	-1.0
FUND 01	3883750	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396FBH5	Jefferies & Co.	645,000.00	98.383000	100.000	98.38300000	0.81404567	516,000.00	-4.01	-2
FUND 13	3883749	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396RKP1	Jefferies & Co.	1,853,000.00	98.589030	100.000	98.58903000	0.87461113	1,597,000.00	-12.42	-6
FUND 09	3883748	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31395NW44	Barclays	1,500,000.00	98.712000	100.000	98.71200000	0.99957252	1,480,000.00	-37.00	-2.0
FUND 11	3883747	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31394V237	Jefferies & Co.	2,411,000.00	98.852000	100.000	98.85200000	0.90133203	2,148,000.00	-16.71	-9
FUND 23	3883746	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31395NW44	Jefferies & Co.	1,599,000.00	98.712000	100.000	98.71200000	0.99957252	1,577,000.00	-12.27	-6
FUND 05	3883745	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31394V237	Barclays	2,200,000.00	98.852000	100.000	98.85200000	0.90133203	1,960,000.00	-49.00	-2.7
FUND 20	3883744	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31395V4Q9	Jefferies & Co.	1,821,000.00	98.086720	100.000	98.08672000	0.82331036	1,470,000.00	-11.43	-6
FUND 21	3883743	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31396RKP1	Barclays	1,750,000.00	98.589030	100.000	98.58903000	0.87461113	1,508,000.00	-37.70	-2.1
FUND 14	3883742	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396FBH5	Jefferies & Co.	1,463,000.00	98.383000	100.000	98.38300000	0.81404567	1,171,000.00	-9.11	-5
FUND 19	3883741	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31396FBH5	Barclays	1,811,000.00	98.383000	100.000	98.38300000	0.81404567	1,450,000.00	-36.25	-2.0
FUND 10	3883740	3074914	Money Fill Repo	LOAN	01/21/2009	05/15/2009	0.2800	0.2800	31396RKP1	Jefferies & Co.	1,881,000.00	98.589030	100.000	98.58903000	0.87461113	1,621,000.00	-12.61	-7
FUND 23	3883739	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31395V4Q9	Barclays	1,200,000.00	98.086720	100.000	98.08672000	0.82331036	969,000.00	-24.23	-1.3
FUND 23	3883738	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31395NW44	Barclays	2,000,000.00	98.712000	100.000	98.71200000	0.99957252	1,973,000.00	-49.33	-2.7
FUND 10	3883737	3074913	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.9000	0.9000	31394V237	Barclays	1,000,000.00	98.852000	100.000	98.85200000	0.90133203	890,000.00	-22.25	-1.2
FUND 27	3883723	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396FBH5	Jefferies & Co.	909,000.00	98.383000	100.000	98.38300000	0.81404567	728,000.00	-6.88	-3
FUND 22	3883722	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396GAD3	Jefferies & Co.	1,500,000.00	98.440540	100.000	98.44054000	0.79181017	1,169,000.00	-11.04	-6
FUND 19	3883721	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396RKP1	Jefferies & Co.	1,100,000.00	98.589030	100.000	98.58903000	0.87461113	948,000.00	-8.95	-5
FUND 14	3883720	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	149,000.00	98.554000	100.000	98.55400000	0.91888074	134,000.00	-0.82	-
FUND 01	3883719	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31394V237	Jefferies & Co.	3,911,000.00	98.852000	100.000	98.85200000	0.90133203	3,484,000.00	-32.90	-1.8
FUND 22	3883718	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	1,220,000.00	98.554000	100.000	98.55400000	0.91888074	1,104,000.00	-6.75	-3
FUND 14	3883717	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31395NW44	Jefferies & Co.	4,241,000.00	98.712000	100.000	98.71200000	0.99957252	4,184,000.00	-39.52	-2.2
FUND 25	3883716	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	1,348,000.00	98.554000	100.000	98.55400000	0.91888074	1,220,000.00	-7.46	-4
FUND 11	3883715	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	1,797,000.00	98.554000	100.000	98.55400000	0.91888074	1,627,000.00	-9.94	-5
FUND 15	3883714	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31395V4Q9	Jefferies & Co.	1,815,000.00	98.086720	100.000	98.08672000	0.82331036	1,465,000.00	-13.84	-7
FUND 07	3883713	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	1,956,000.00	98.554000	100.000	98.55400000	0.91888074	1,771,000.00	-10.82	-6
FUND 02	3883712	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	2,159,000.00	98.554000	100.000	98.55400000	0.91888074	1,955,000.00	-11.95	-6
FUND 30	3883711	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	2,178,000.00	98.554000	100.000	98.55400000	0.91888074	1,972,000.00	-12.05	-6
FUND 11	3883710	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396FBH5	Jefferies & Co.	2,382,000.00	98.383000	100.000	98.38300000	0.81404567	1,907,000.00	-18.01	-1.0
FUND 18	3883709	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	2,219,000.00	98.554000	100.000	98.55400000	0.91888074	2,009,000.00	-12.28	-6
FUND 29	3883708	3074911	Money Fill Repo	LOAN	01/21/2009	OPEN	0.2200		31396HG46	CANTOR FITZGEI	3,326,000.00	98.554000	100.000	98.55400000	0.91888074	3,012,000.00	-18.41	-1.0
FUND 13	3883707	3074910	Money Fill Repo	LOAN	01/21/2009	06/15/2009	0.3400	0.3400	31396GAD3	Jefferies & Co.	1,856,000.00	98.440540	100.000	98.44054000	0.79181017	1,446,000.00	-13.66	-7

XYZ Trading/helifs Pending: 0/0 Rejected: 0/0 Wait ACK: 0 Unallocated: 1/1 03/18/2009 support01

HELIXfinancial system

# CQCs on a CQC

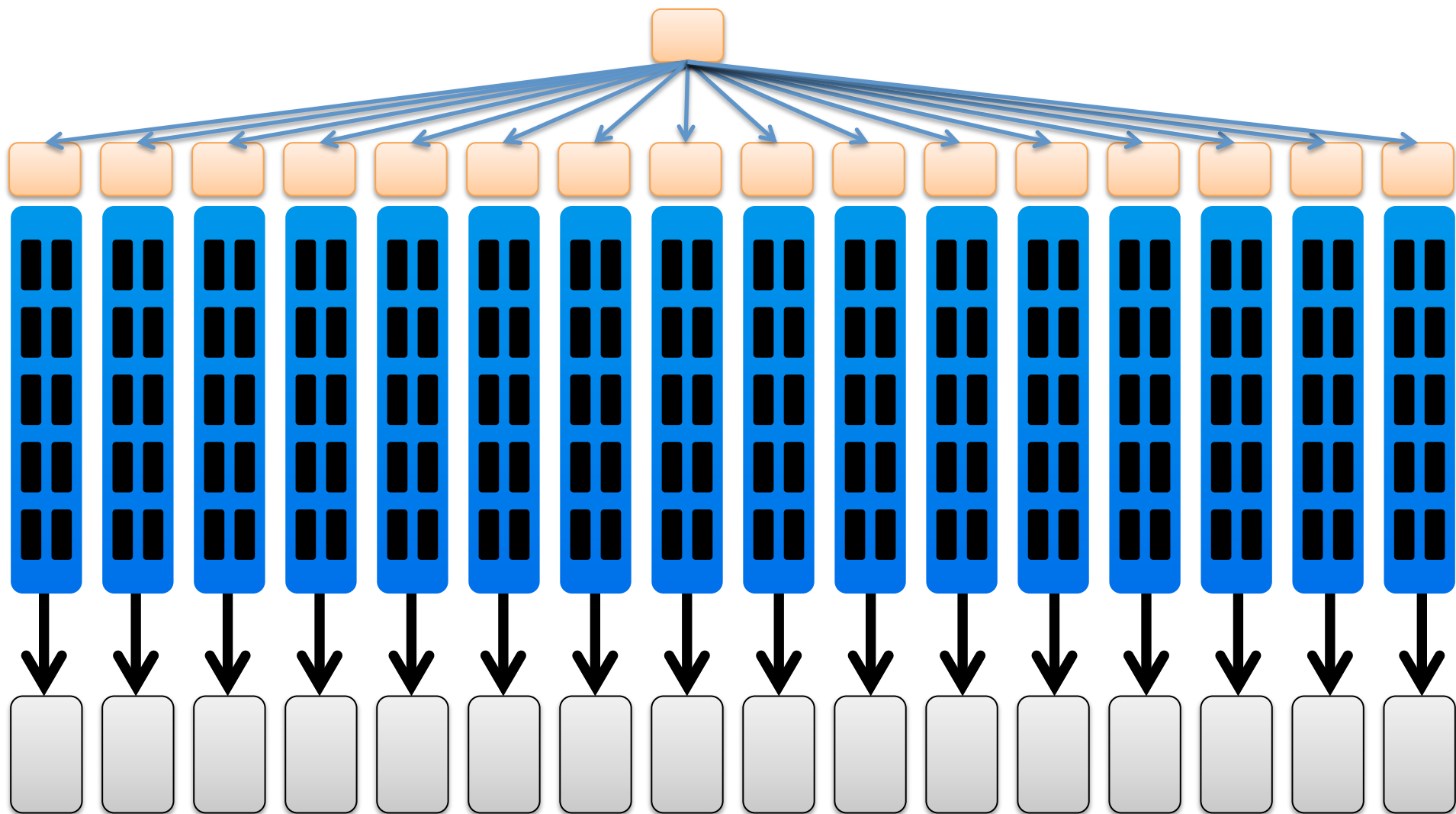


Define this in config

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>