# QCon 2012

## Progressive Architectures at the Royal Bank of Scotland

Mark Atwell

Farzad ‘Fuzz’ Pezeshkpour

Ben Stopford

# Overview

History – Front-Office-Centric

‘The City’ has been the source of industry innovation

FPML

AMQP

OpenAdaptor

LMAX Disruptor

# New Drivers

Change: Even more than before!:

- Regulation
- Competition – The ‘Race to Zero’
- Utilisation – data-centres, ‘Just Enough’ hardware

All in the face of even tighter budgets

However – there is hope!: New technologies and architectures

We will show you some of the challenges and our approaches

**Necessity has become an even greater Mother of Innovation**



RBS™

# Agenda

Ben: Pauseless Java in Low Latency Trading

Beyond Messaging with ODC

Fuzz: Risk: Increased Accuracy - Big Compute and Big Data

Mark: Data Virtualisation

Collaboration

Questions

# Development Areas

UX

Data & Resource Consolidation

GPGPUs, FPGA

Big Data (for us)

Data Visualisation

Data Virtualisation

We' re driving the shape and features of products – particularly in the last two



RBS™

# The Manhattan Processor

## Avoiding GC Pauses in High Frequency Trading

# The Problem:


**615p**

**595p**

RBS

Full GC

**Apple announces Financial Results above expectations**

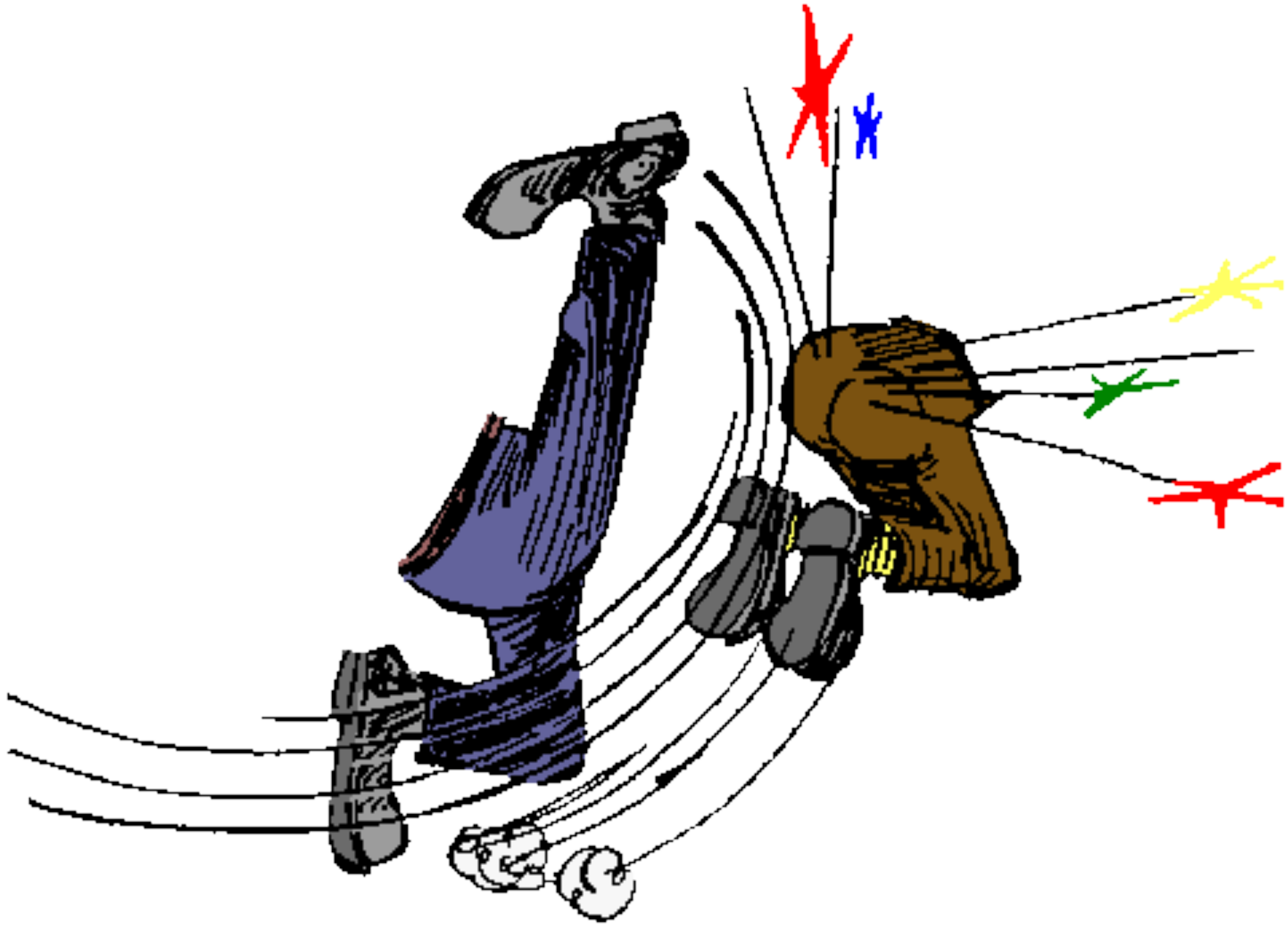
**Buy @ 595p**

**Short @ 615p**



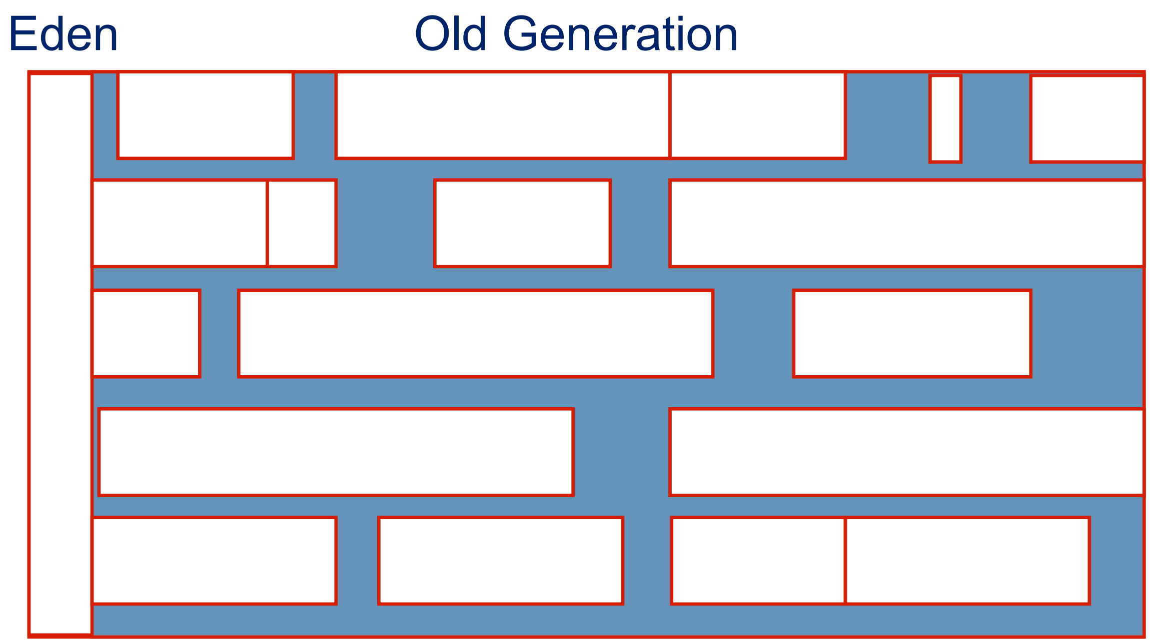
RBS™

# Angry Trader kicks techies collective behinds for losing one months P&L in a one second GC!



# So can we just tune the collector?

- Cost ~ 1 sec per GB of heap at best
- Stop-the-World happens more than you may think, even with CMS
  - Eden collections
  - Mark/Remark in CMS
  - Full GC (really hurts)
  - CMS falls back to Full GC on ‘Concurrent Mode Failure’

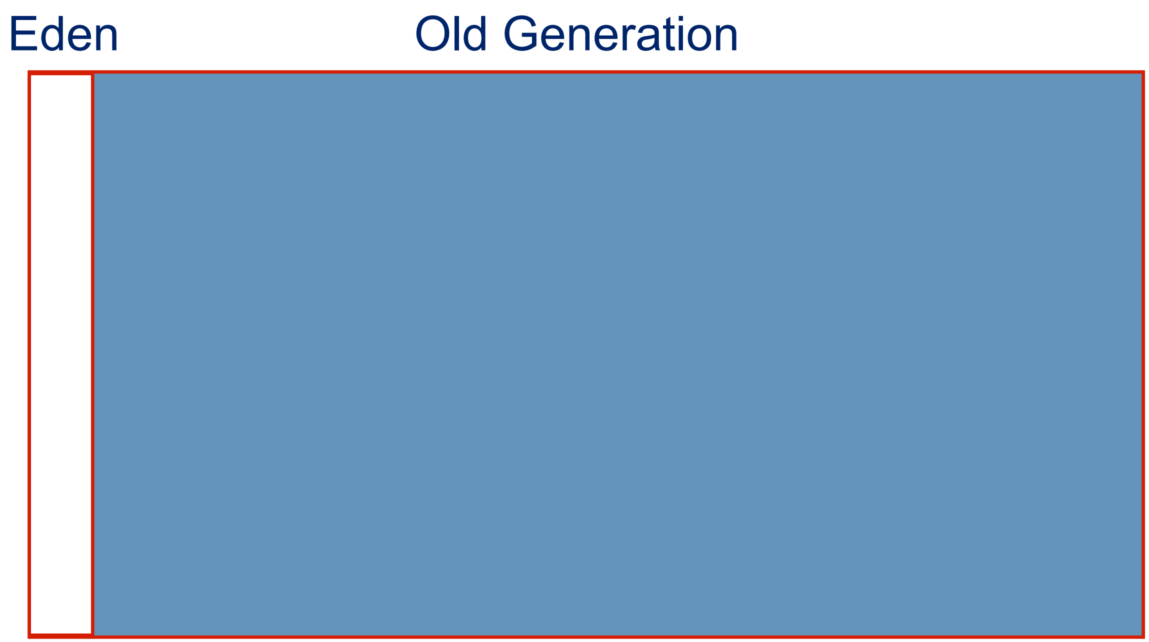


# Avoiding Pauses: Model 1

- Use a huge Eden space (several GB’s)
- Create minimal Garbage

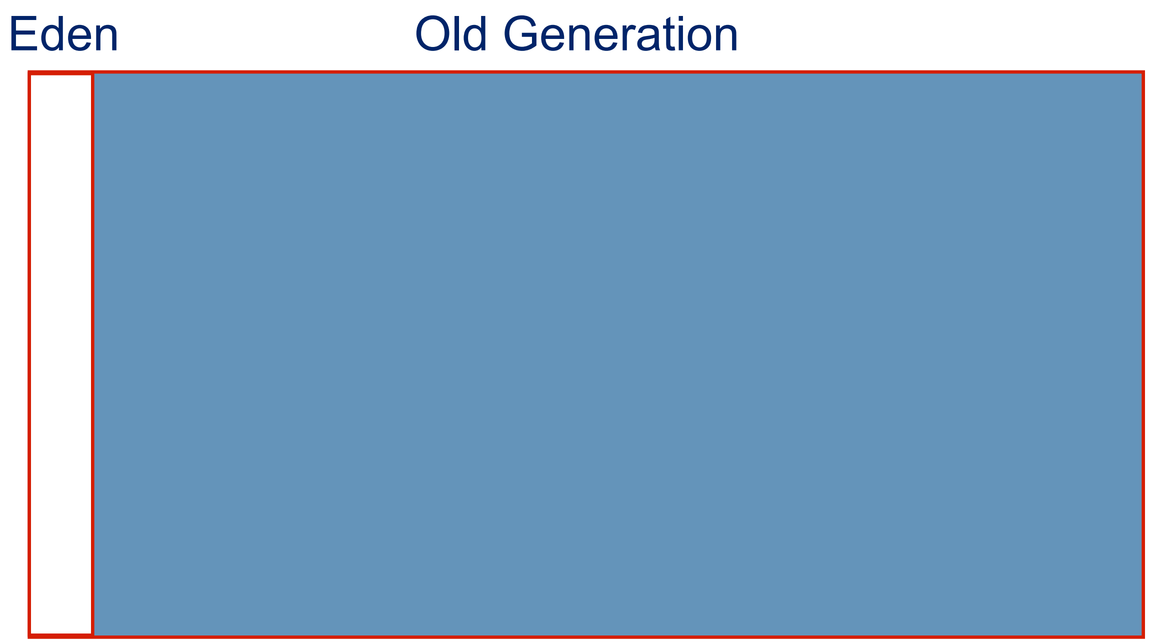
# Avoiding Pauses: Model 2

Use a small Eden that can be collected in a know time (~ 1ms typically) and avoid FullGC



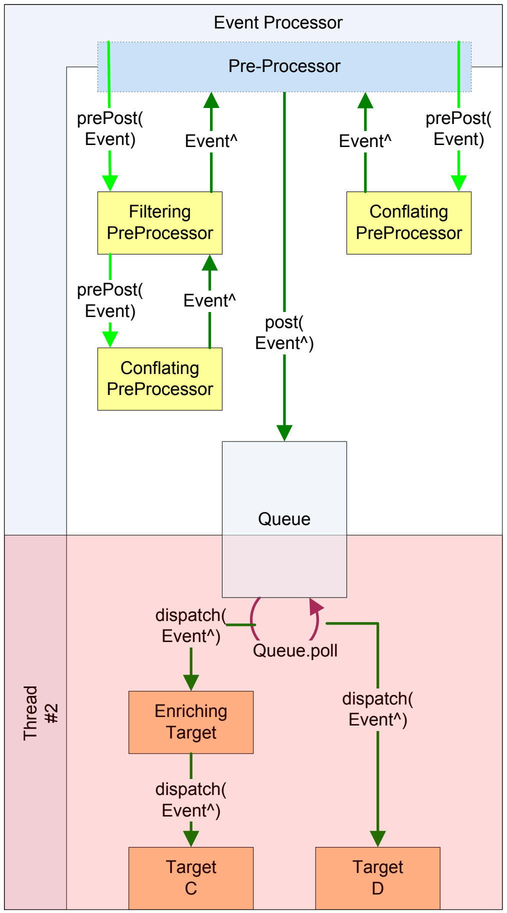
# We use Model 2

- No control over 3rd party
- It’s very hard to eliminate the production of garbage completely
- We can tolerate very small collections

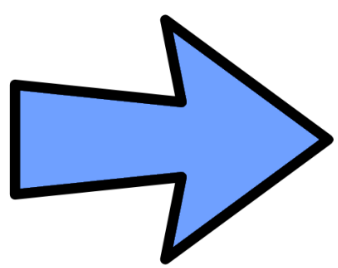


Manhattan Processor: Concurrent processing with a very low garbage footprint.

# Manhattan Processor




- In-house non-allocating queue specialised for multiple producer, single consumer requirement
- Non-blocking for offer and poll
- Non-blocking object pool
- Pluggable & chainable pre- and post-processors
- Direct in-thread dispatch where there is thread affinity between producer and consumer



Minimal Garbage

## Result

- Fast, pluggable architecture for concurrent processing
- Predictable pauses <1ms
- No full collection pauses in 15hr trading day

# Risk: Accuracy ➔ Big Compute and Data

Market and Credit Risk Technology

# What’s the Big Deal?

- Risk: Predicting the Future, looking for worst case scenarios

- Measuring Risk is not simple

  - Not as difficult as predicting the weather, but not far off

- Doing it right implies a lot of data and compute

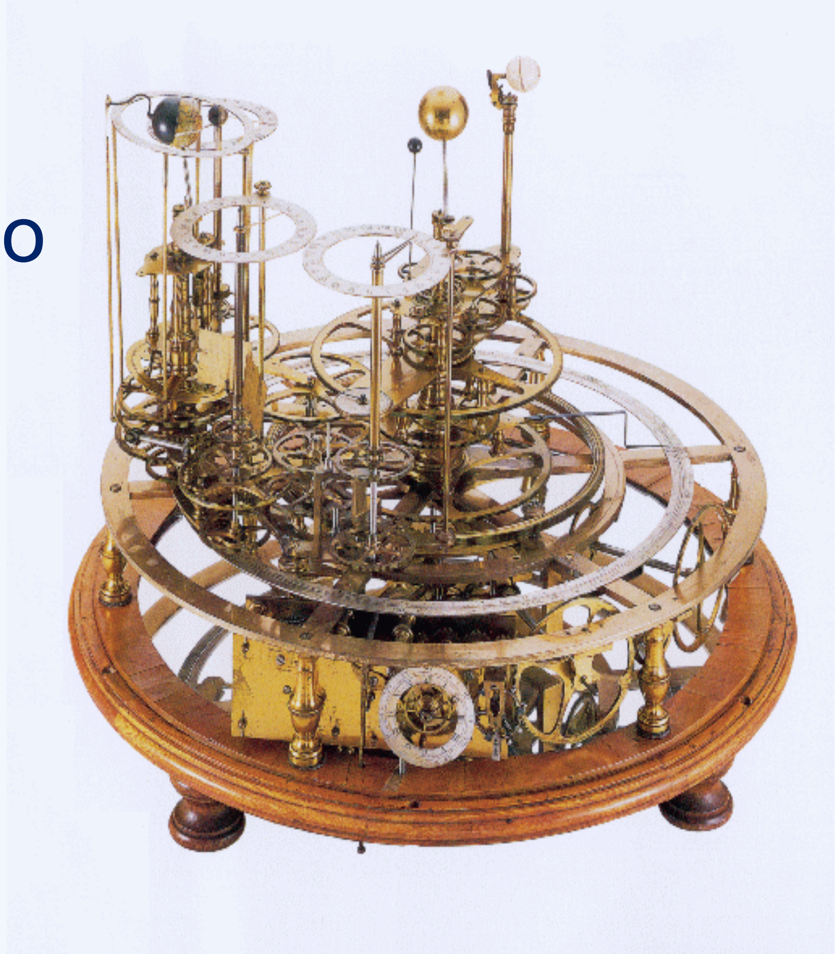
- Impact on architecture and technology choices

# Disclaimer:

**The numbers quoted in this presentation are indicative of magnitude only.**

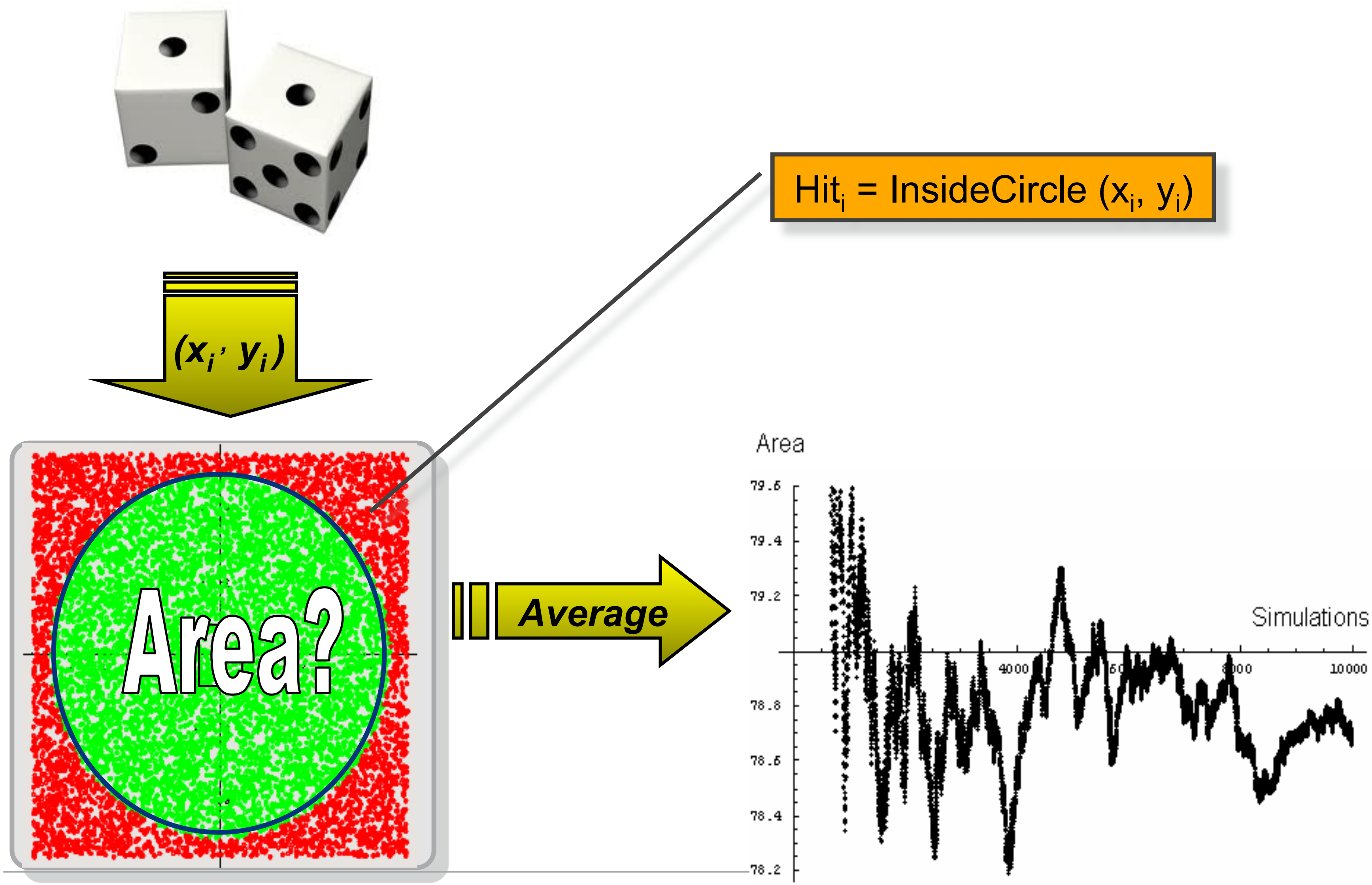
# Historically, Banks have....

- Linear approximation
- Single-step Monte Carlo
- Multi-step Monte Carlo

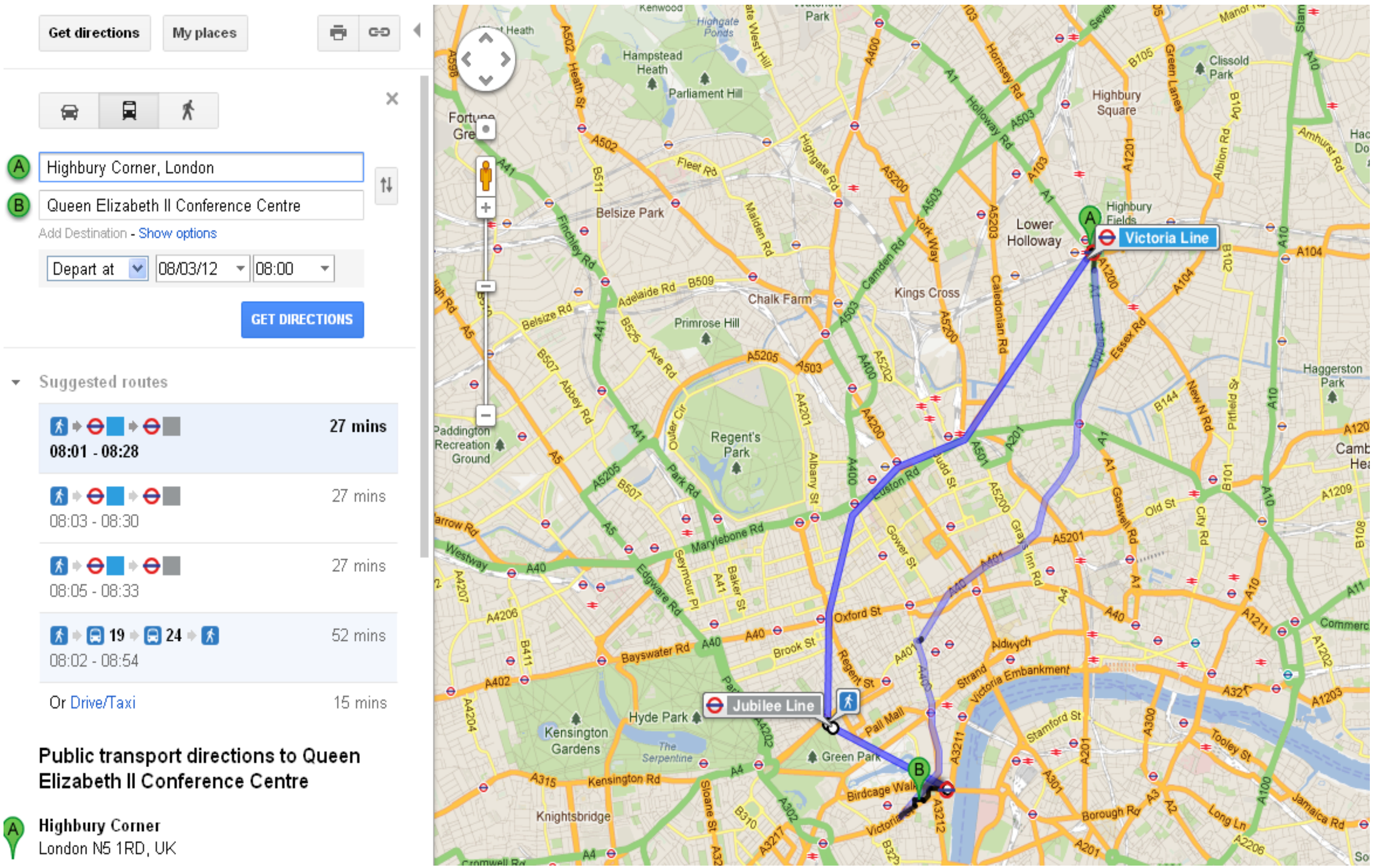


**The Market is a complex system and needs to be modelled as such**

# Monte-Carlo: Example



# Single Step vs Multi Step Monte-Carlo



# Monte-Carlo of my route to QCon

- 10s of relevant routes to Westminster
- 100s of events that affect the journey (weather, traffic jams, dropping off the kids to school ...)
- Each of these is correlated with other events

  e.g. Traffic and Weather
- Events can happen at <u>many</u> discrete points in time
- Need 1000's of random numbers to simulate this

# Monte-Carlo: Data Volumes

## Single Step Simulation

- 1’000 paths
- 1 correlated random number per path
- ~50 observables
- Updated quarterly

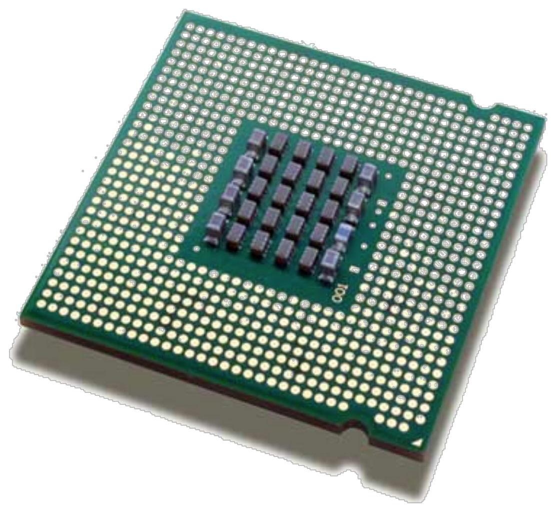
## Multi Step Simulation

- 10’000 paths
- 25’000 correlated random numbers per path
- 1000’s observables
- Updated monthly

**Multi-Step requires significantly more Data**

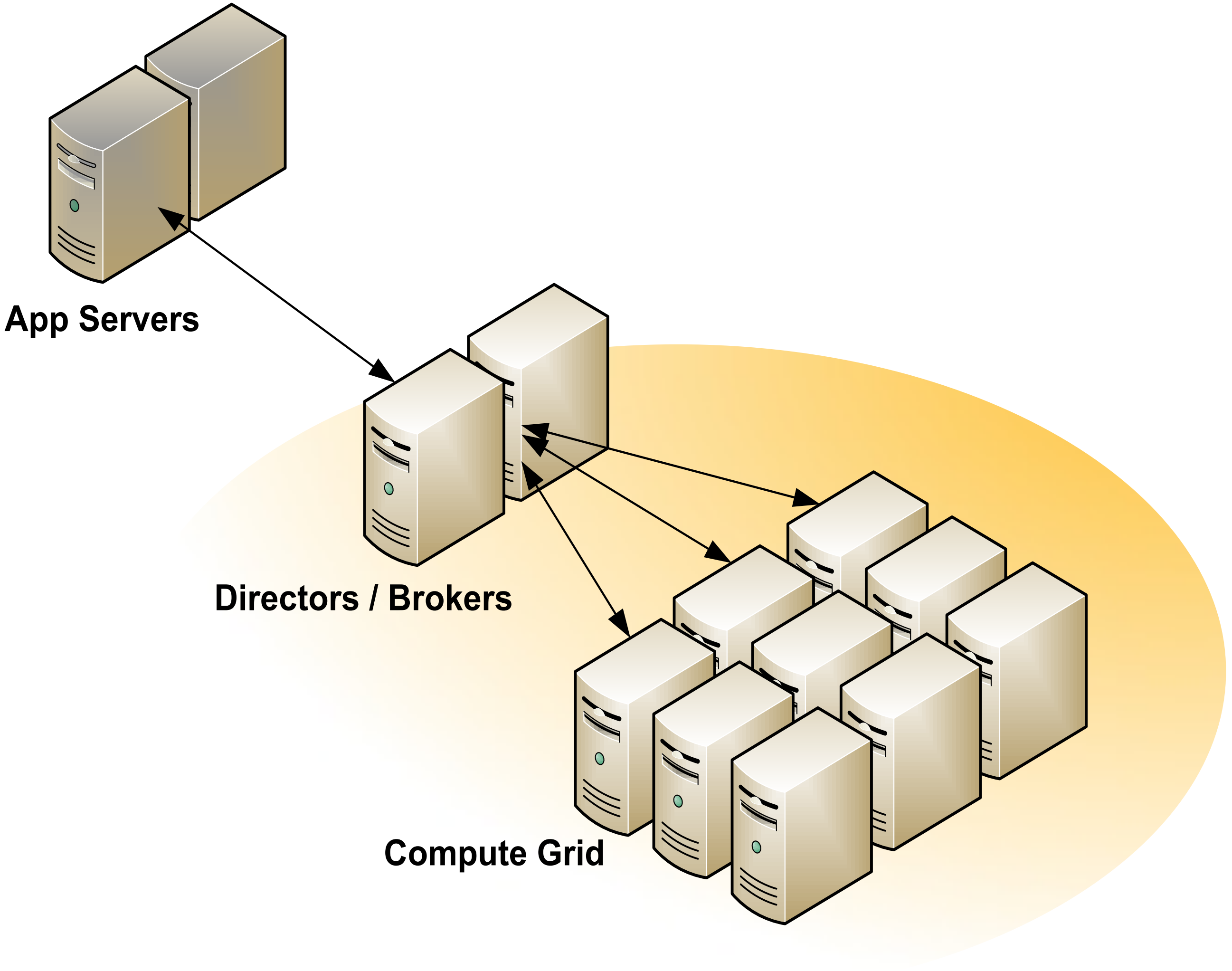
# How much Compute?

| Single Step Simulation | | Multi Step Simulation |
|---|---|---|
| < 4,000 compute hours | → | ~ 20,000 compute hours |
| < 1,200 CPU cores | → | ~ 5,000 CPU cores |



**Massive growth in Computations**

# How do we handle that?




**Standard Compute Grid Architecture**

# Huge difference in required Data Volumes

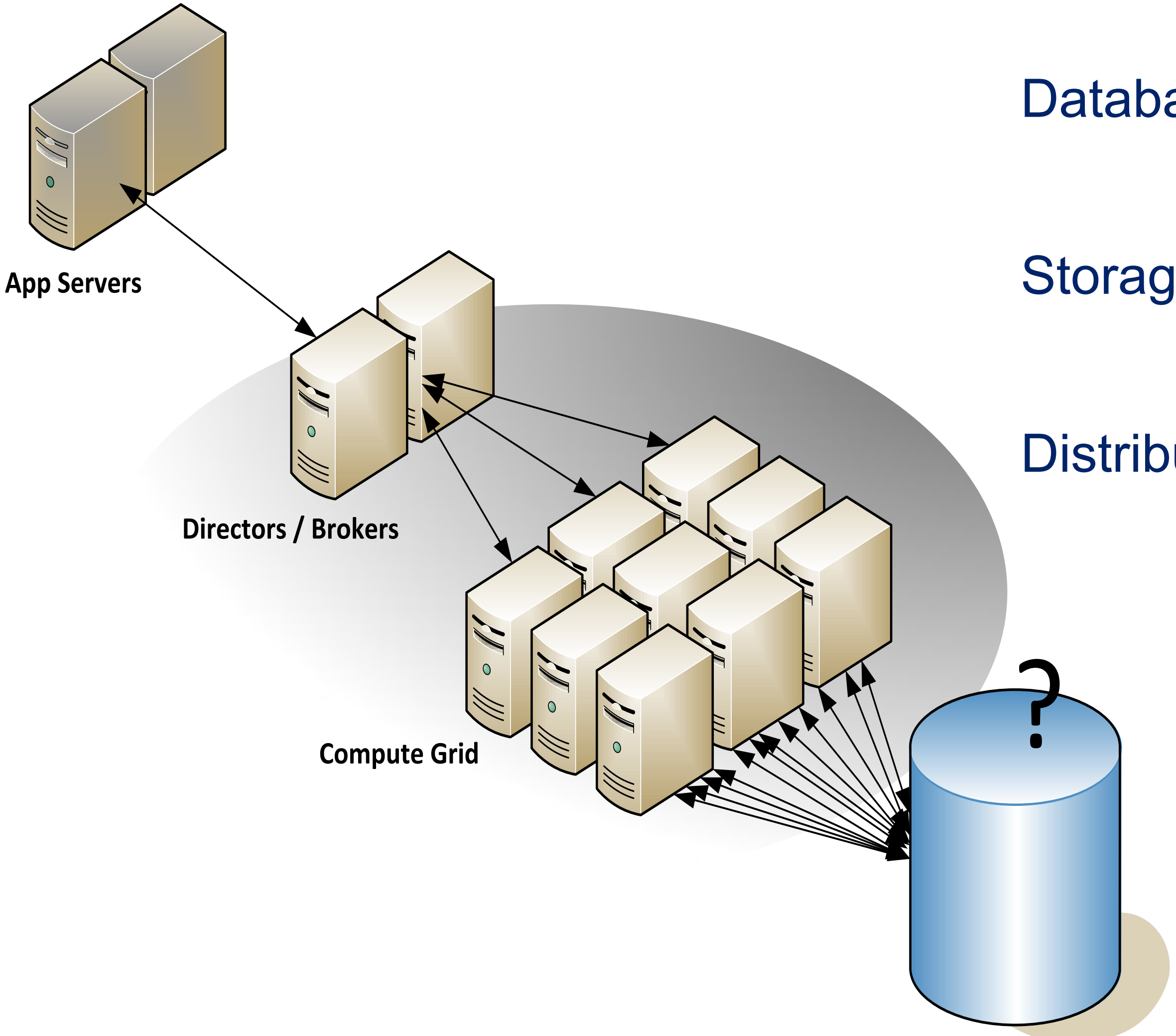
Single Step Simulation

Multi Step Simulation

**~ 2 MB** → **~ 10 TB**

**How will this data be accessed?**

# Storage ....




Database?

Storage Appliance?

Distributed Filesystem?

**What's the right solution?**

# Requirements

| | |
|---|---|
| Speed | Concurrent Access Bandwidth<br>Transaction Throughput |
| Scalability | Concurrent Access<br>Rack awareness |
| Robustness | Resilience and Recovery<br>Guaranteed Correctness |
| Interoperability | Access from both Linux and Windows Server Grid Engines. |
| Support Infrastructure | Logging and Performance Monitoring<br>Support Infrastructure |

# Distributed File Systems

# Selection of Top Users

1-100 nodes, many clusters

532 nodes – 5.3 PB storage

1100 nodes, 8800 cores, 12 PB raw storage

120,  580,  1200 nodes

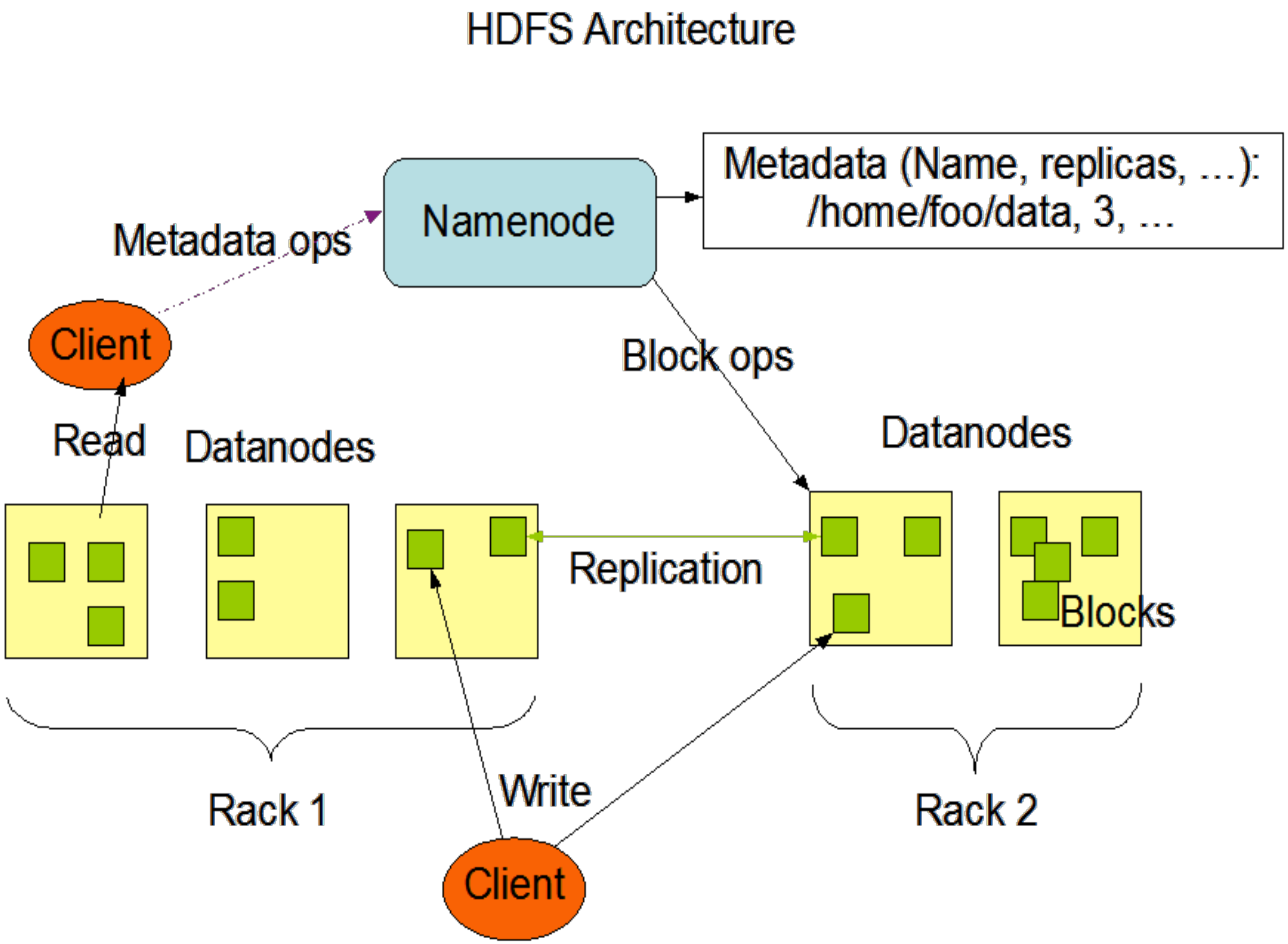
40,000 nodes

**Hadoop scales and is mainstream**



RBS™

# Quick dip into HDFS

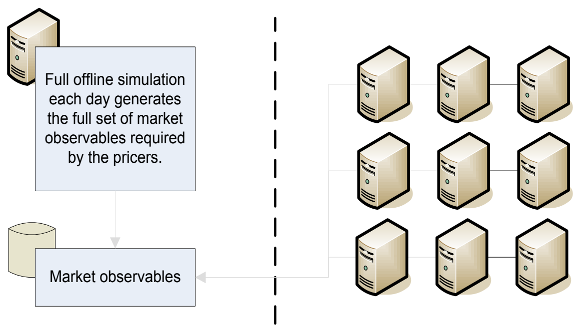





RBS™

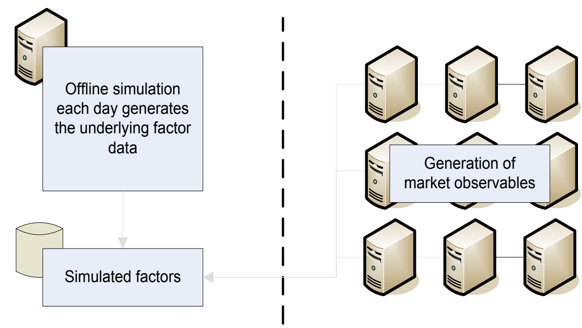
# We tried several architectures

## Option 1



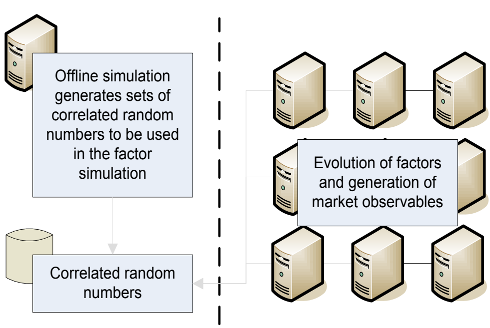

Distributed calculation nodes access the market observables directly on demand. The storage and remote access technology is not defined, but the following could be used:
Disk based storage with access over HTTP
iRODS server with storage in HDF5

## Option 2



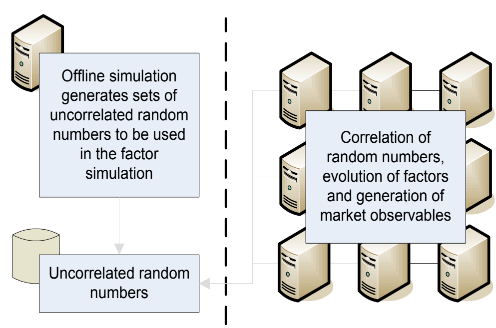

Distributed calculation nodes access the underlying simulated factors and then generate the market observables as required. The storage and remote access technology is not defined, but the nature of the simulated factors may lend itself to a technology that supports some type of sub-element access.

## Option 3a



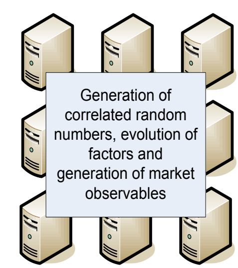

Distributed calculation nodes access the correlated random numbers on demand. The factors are then evolved and the market observables generated. The storage and remote access technology is not defined, but the nature of the correlated random numbers may lend itself to a technology that supports some type of sub-element access.

## Option 3b




Distributed calculation nodes access the correlated random numbers on demand. The factors are then evolved and the market observables generated. The storage and remote access technology is not defined, but the nature of the correlated random numbers may lend itself to a technology that supports some type of sub-element access.

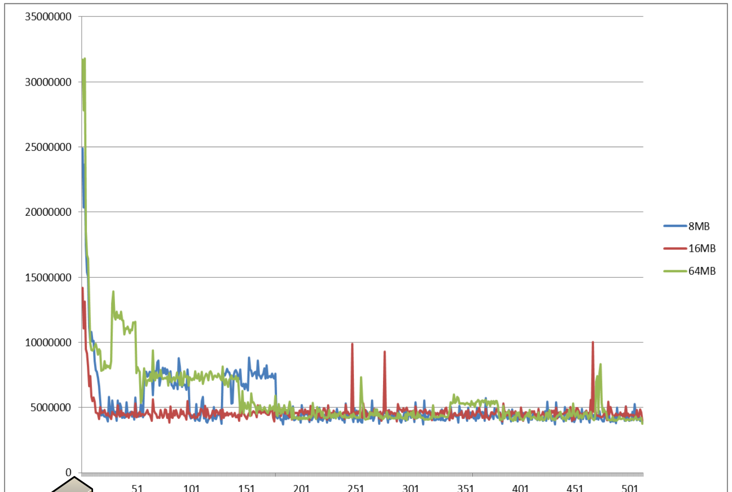
## Option 4

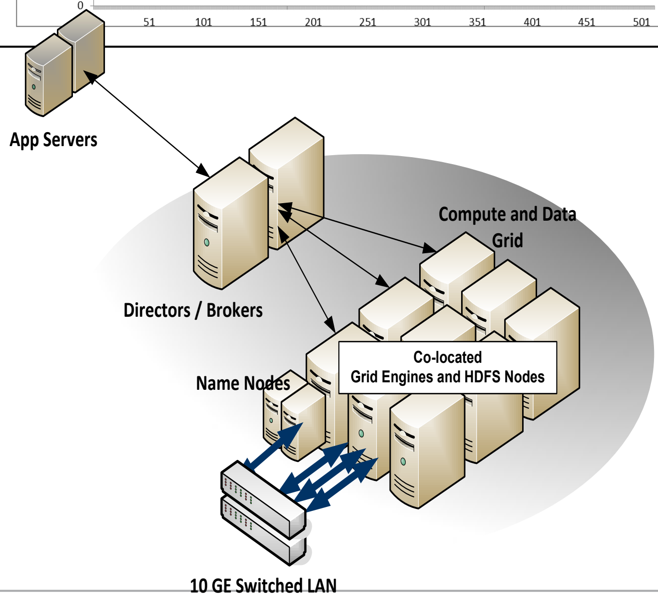



Distributed calculation nodes generate the correlated random numbers, the factor evolution and the market observables as required.

# And carried out a lot of analysis …

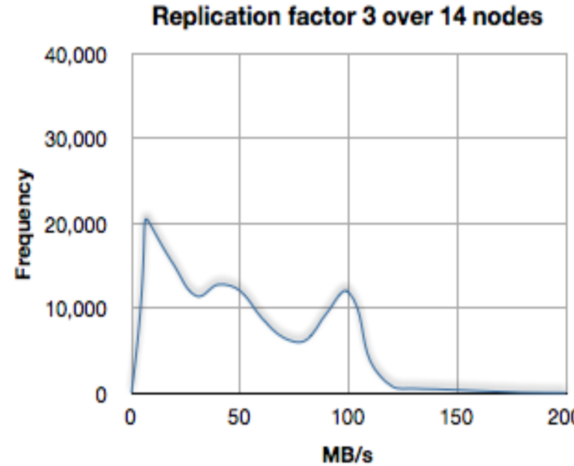
**Transfer speeds as a function of HDFS block size**

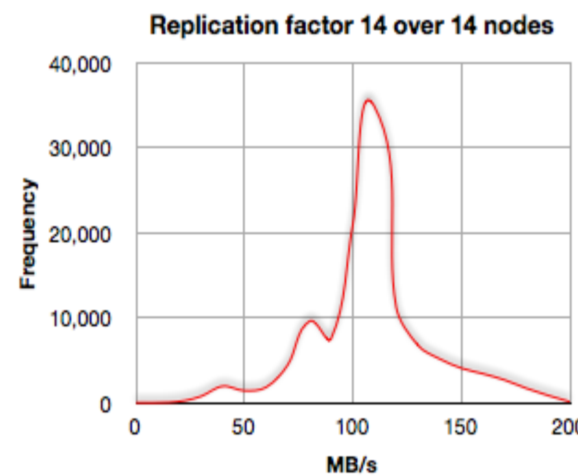


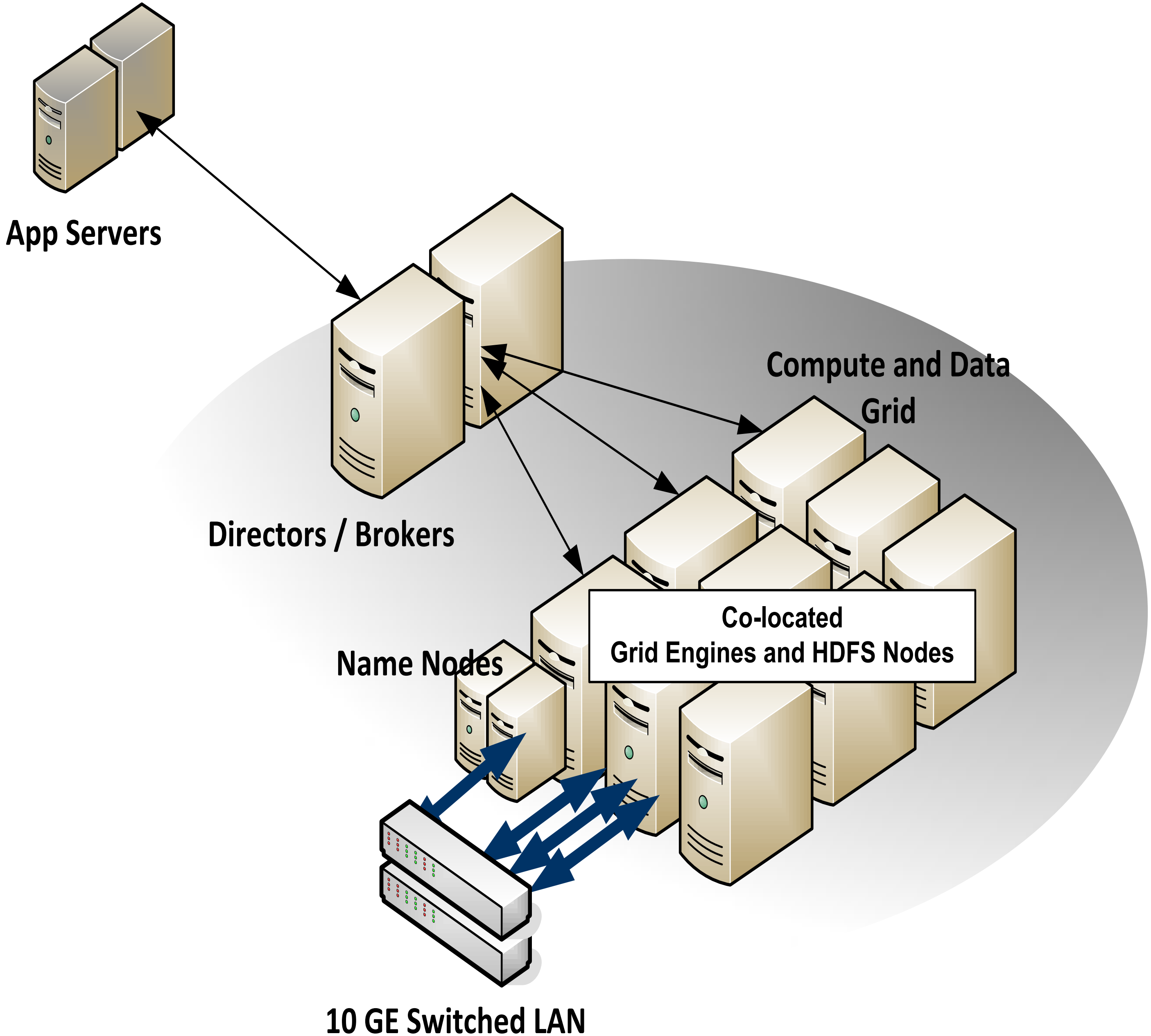




**Varying replication count significantly affects average read speed:**

**3 of 14 = ~ 50MB/s** **14 of 14 = ~ 100MB/s**




App Servers
Directors / Brokers
Compute and Data Grid
Co-located
Grid Engines and HDFS Nodes
Name Nodes
10 GE Switched LAN

# Where Next?

## Software Optimisations

- Data Structures
- Distribution Algorithms
- Uncorrelated and Correlated Random number generation

## Data Compression

## Novel Hardware Architectures



££ / Compute

**Optimisations are being carried out continuously to reduce costs**



RBS

# Summary

Risk is challenging and interesting!

Our focus is accuracy, speed and cost

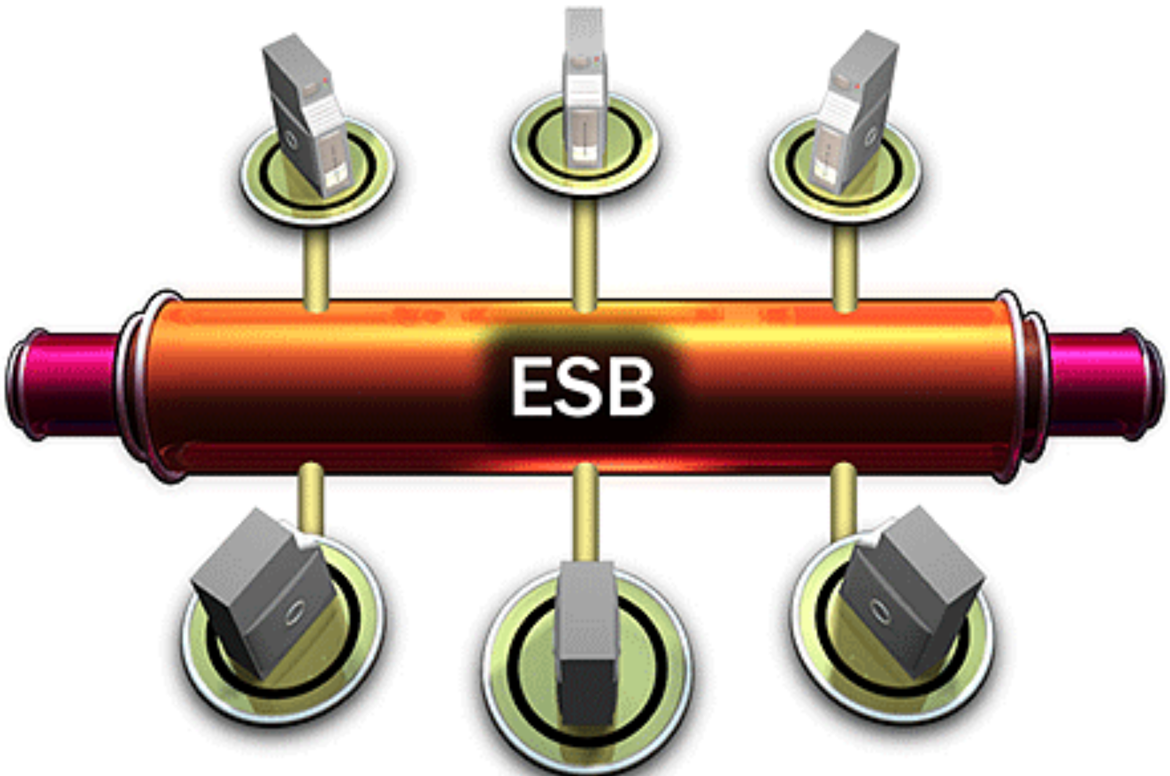
We are innovating methodologies, architecture & engineering

**Thank you**

# ODC

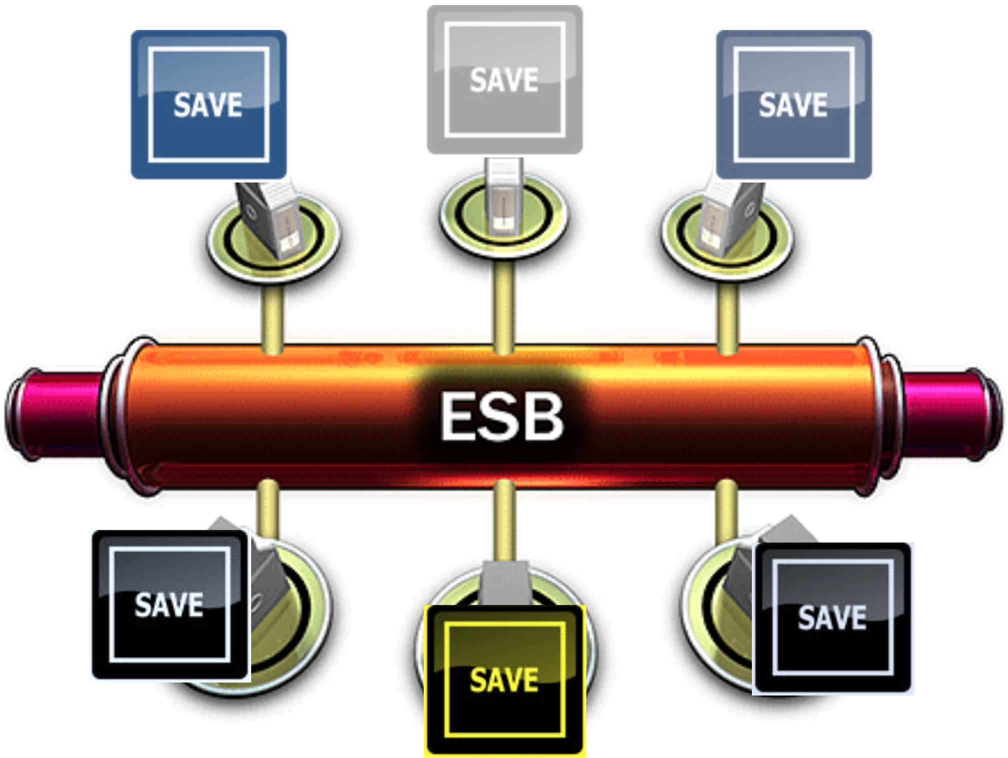
## One Version of the Truth For Everyone

# Messaging allows us to disseminate facts

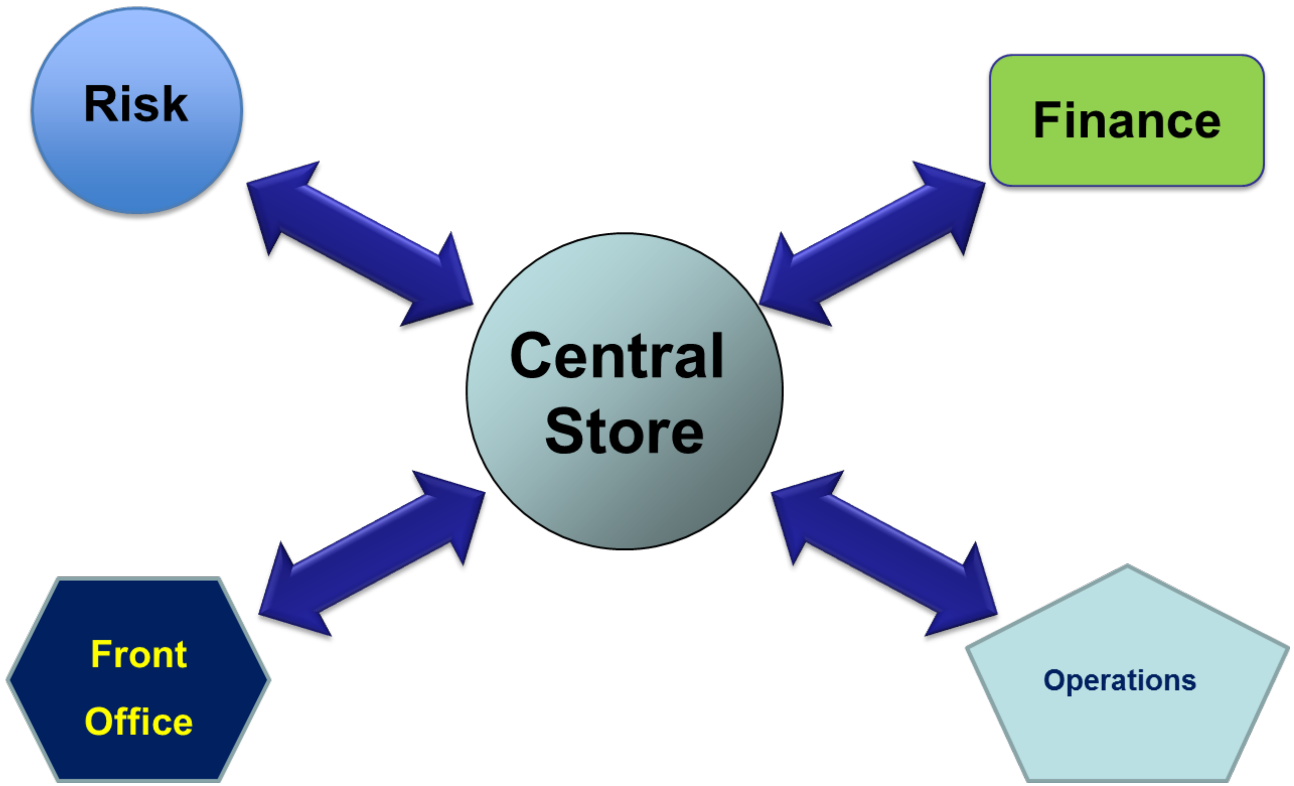


# …but messaging has its limits

# It leaves interpretation of different facts to the various consumers



# Centralising data gets all eyes on a single version of the truth

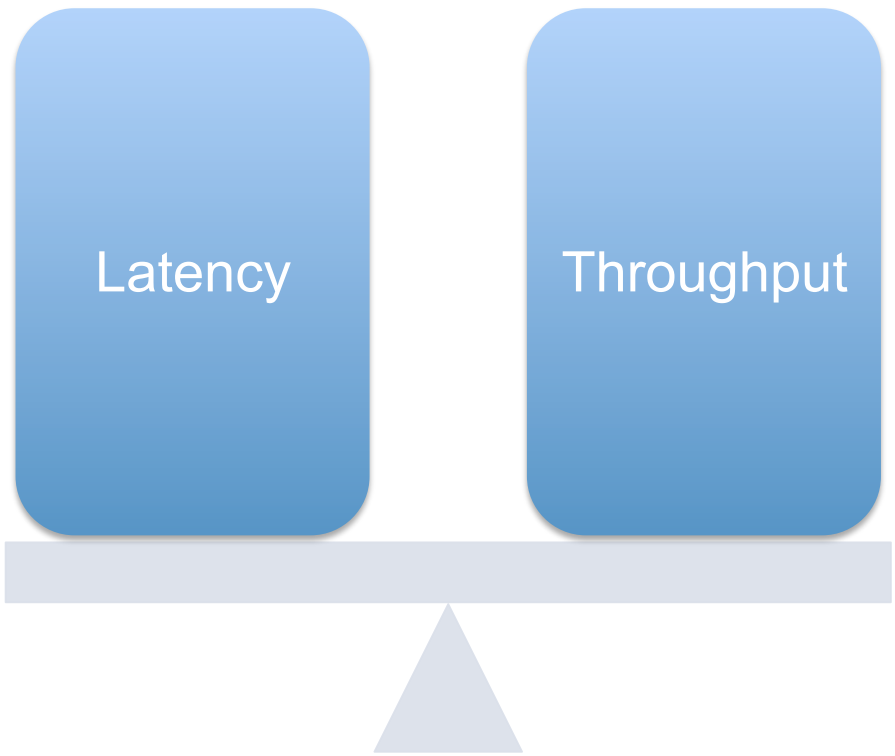


# Originating from concept of a central ‘brain’ within the bank:

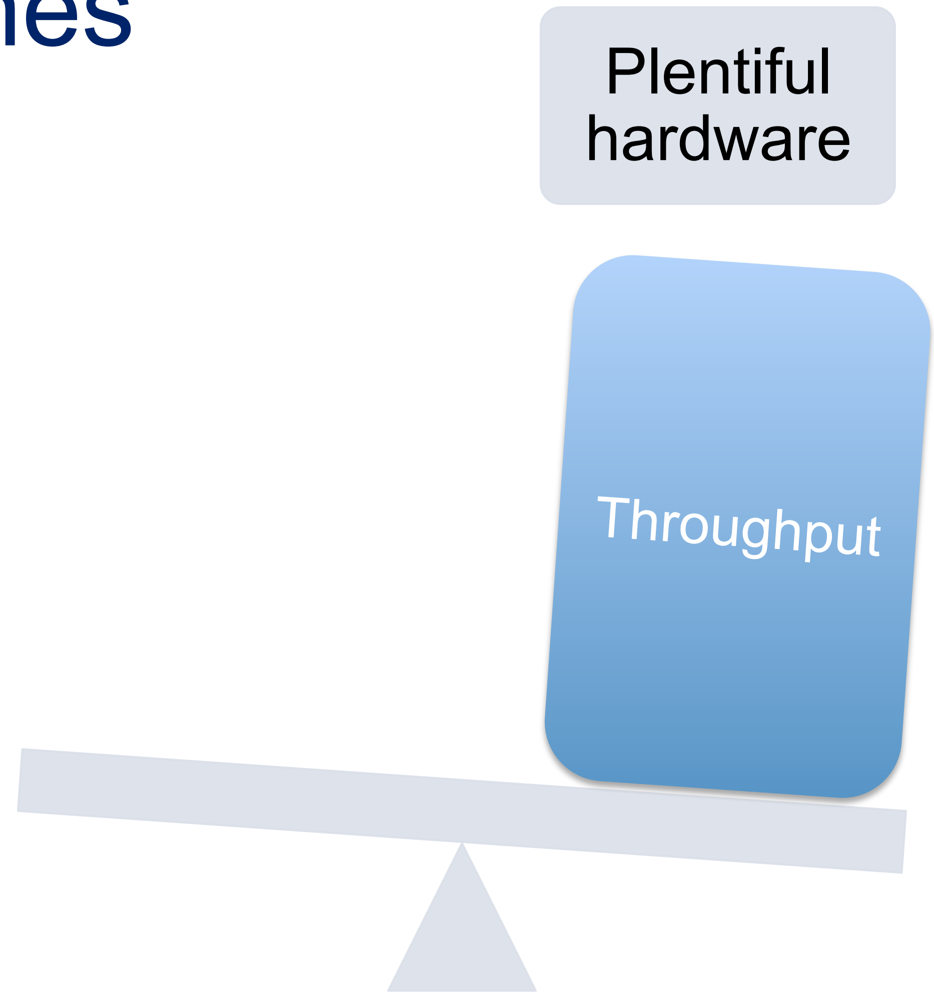
*“The copying of data lies at the route of many of the bank’s problems. By supplying a single real-time view that all systems can interface with we remove the need for reconciliation and promote the concept of truly shared services” - Scott Marcar (Head of Risk and Finance Technology)*

# Nice idea, but how do you do it?

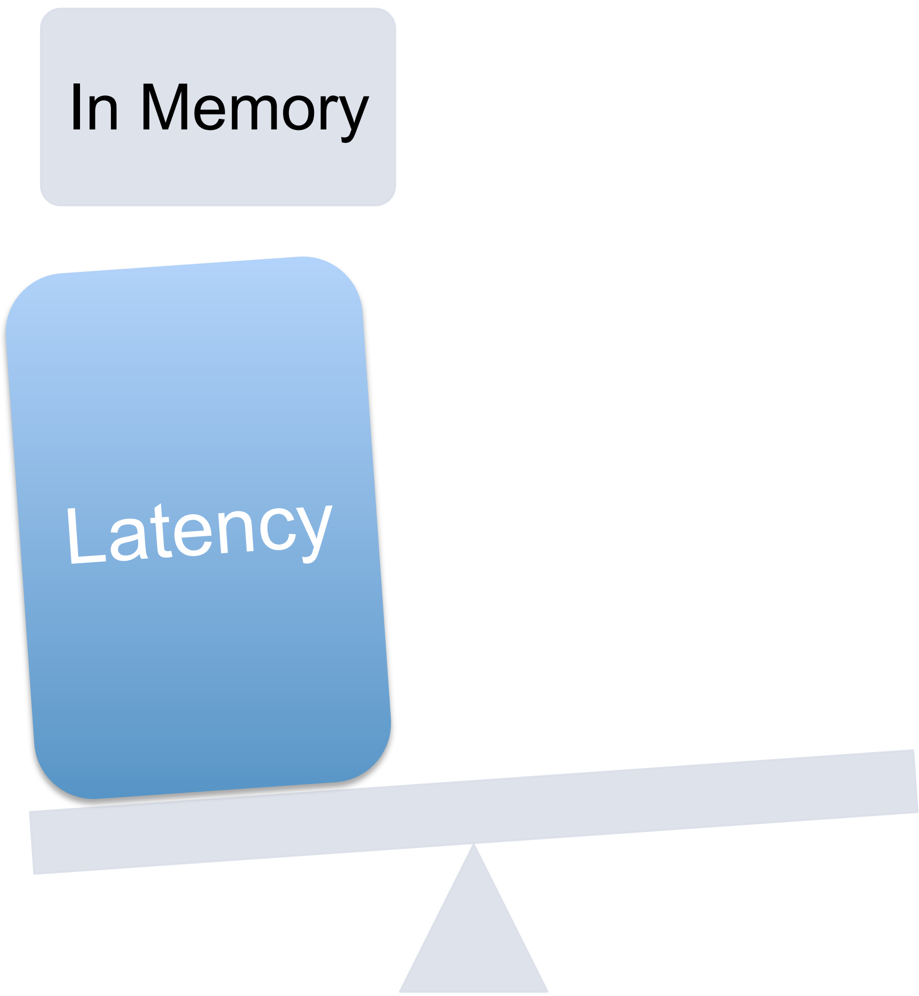
# Needs a balance of low latency and high throughput



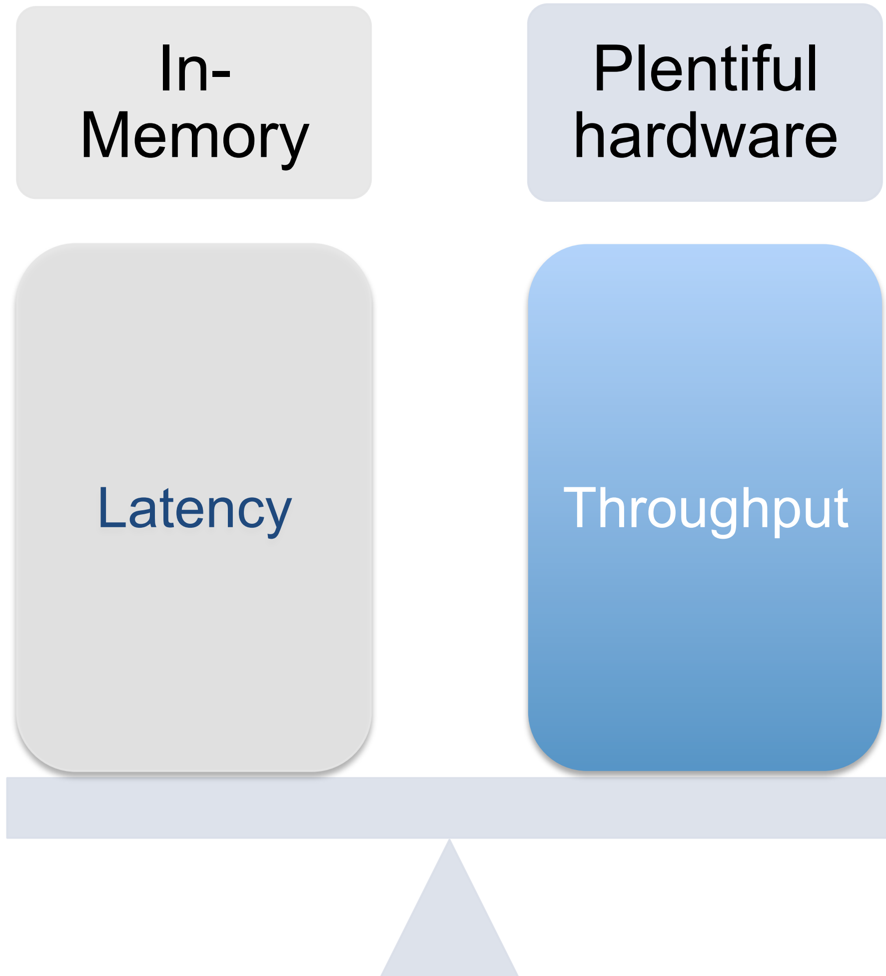
# Throughput generally equates to lots of machines



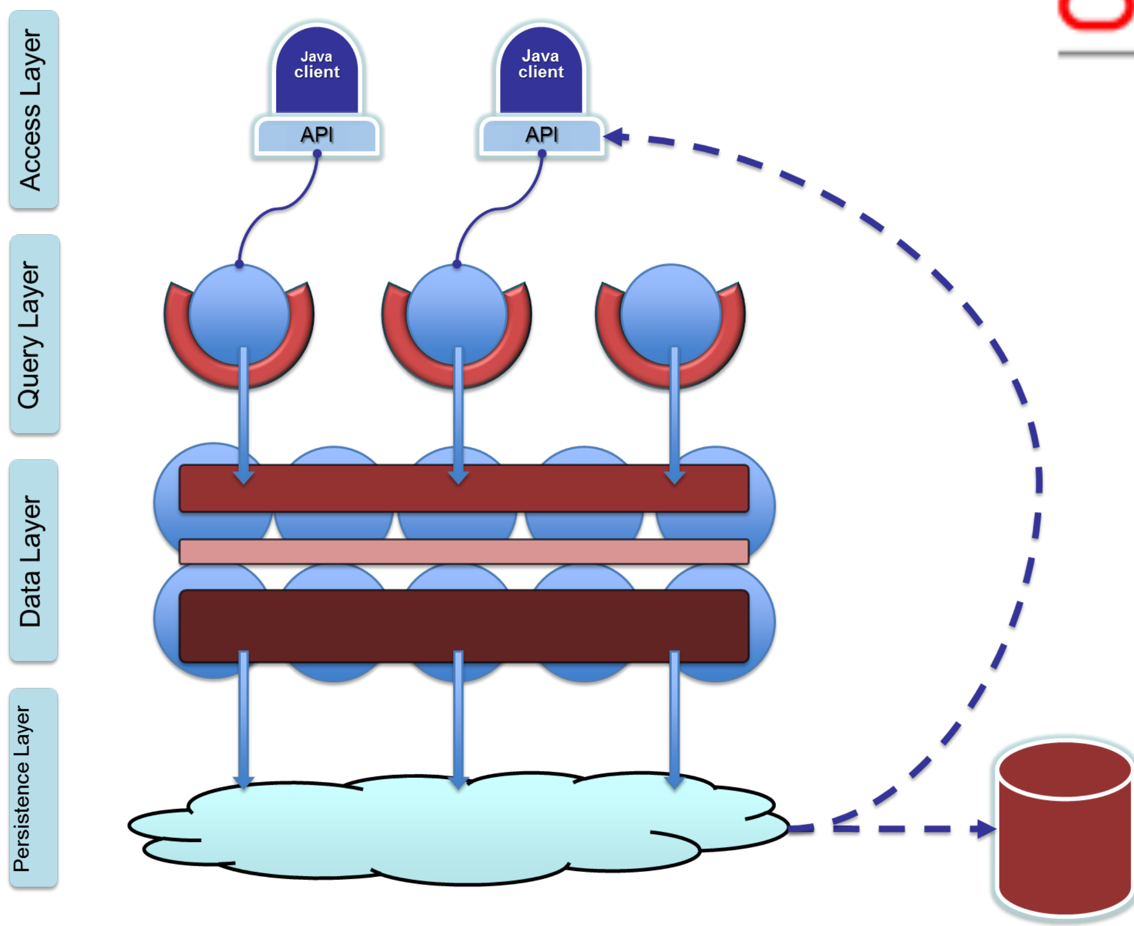
# Low latency = in-memory / keyed access



# ODC is a balance

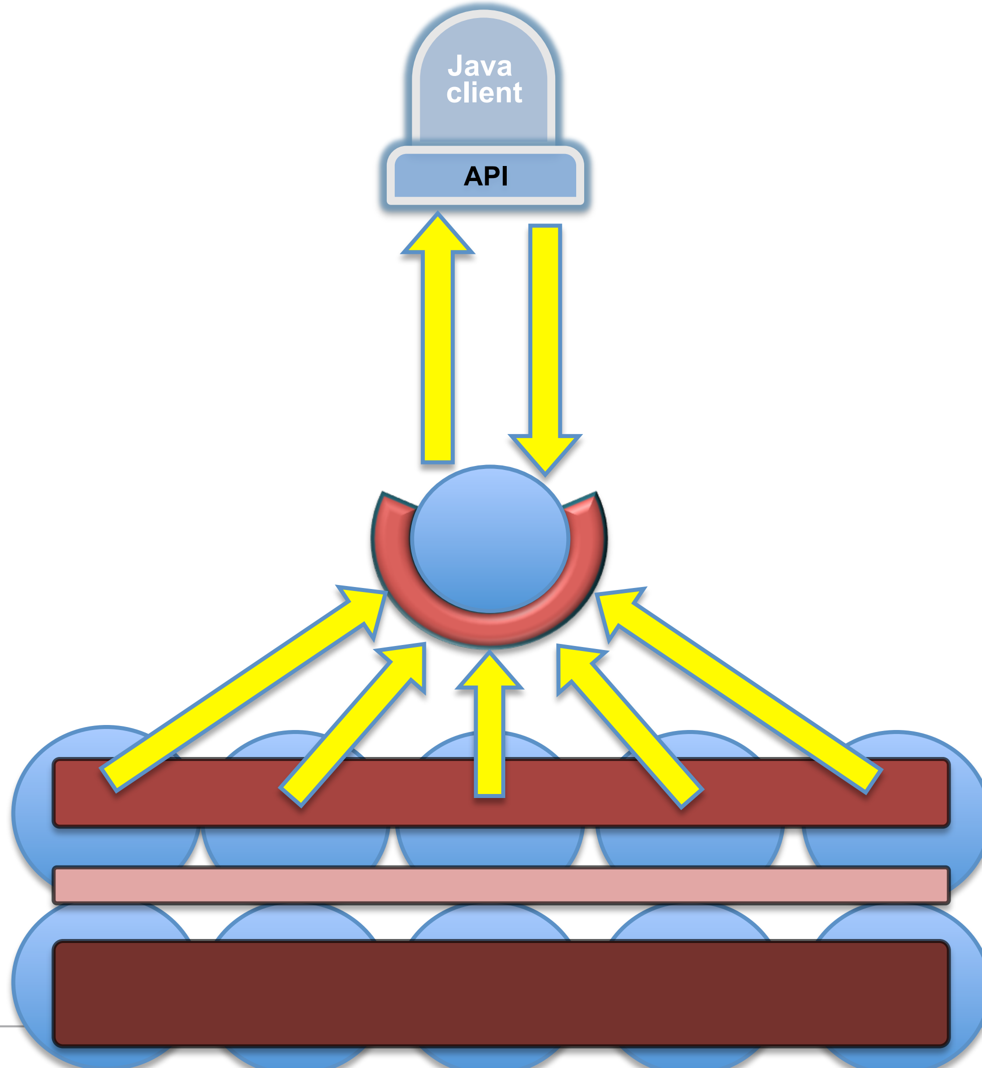


# Oracle Coherence provides the data fabric




ORACLE
Coherence

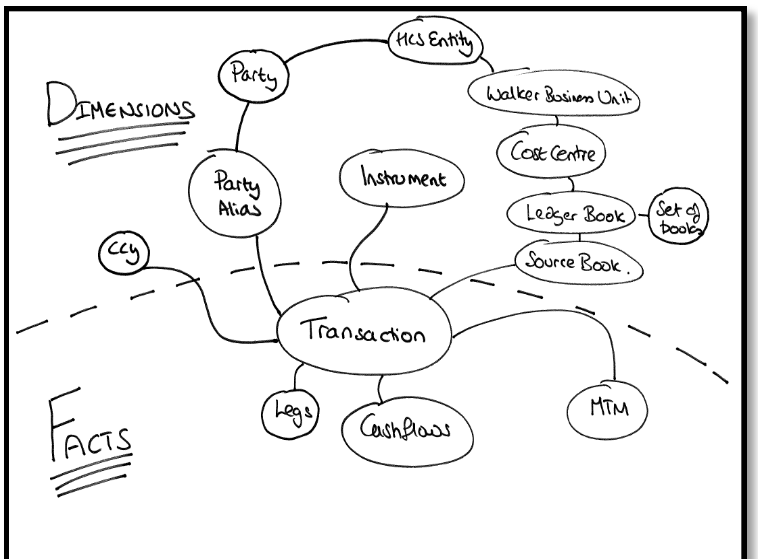
# Coherence is an In-Memory (NoSQL) caching technology with a Shared-Nothing Architecture



# …but ODC adds normalisation… …which means it does joins!

```
QueryDefinition query = odc.fromTransaction()
        .where(valueOf(transaction().getSourceBook().getName()).equalTo("CITI7061"))
        .joinToValuation().where(valueOf(valuation().).equalTo(today())).createQuery();
```

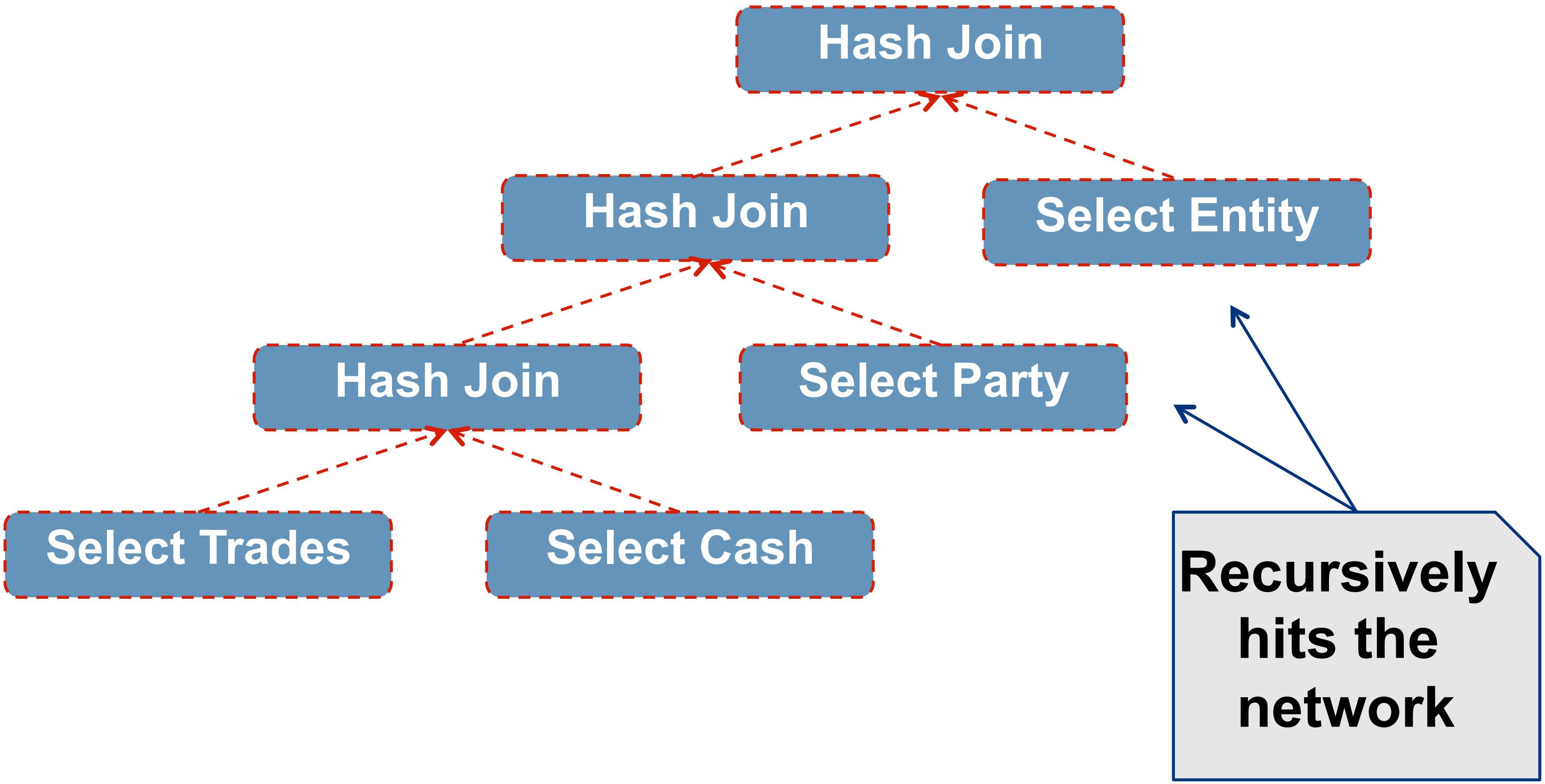
```
getBusinessCentre()                      String
getCurveSystemId()                       SystemInstanceId
getEndDateMarkedRate()                   BigDecimal
getId()                                  DailyTransactionValuationId
getNextValuationDueDate()                BusinessDate
getProjectedEndDate()                    BusinessDate
getRate()                                BigDecimal
getStartDateMarkedRate()                 BigDecimal
```



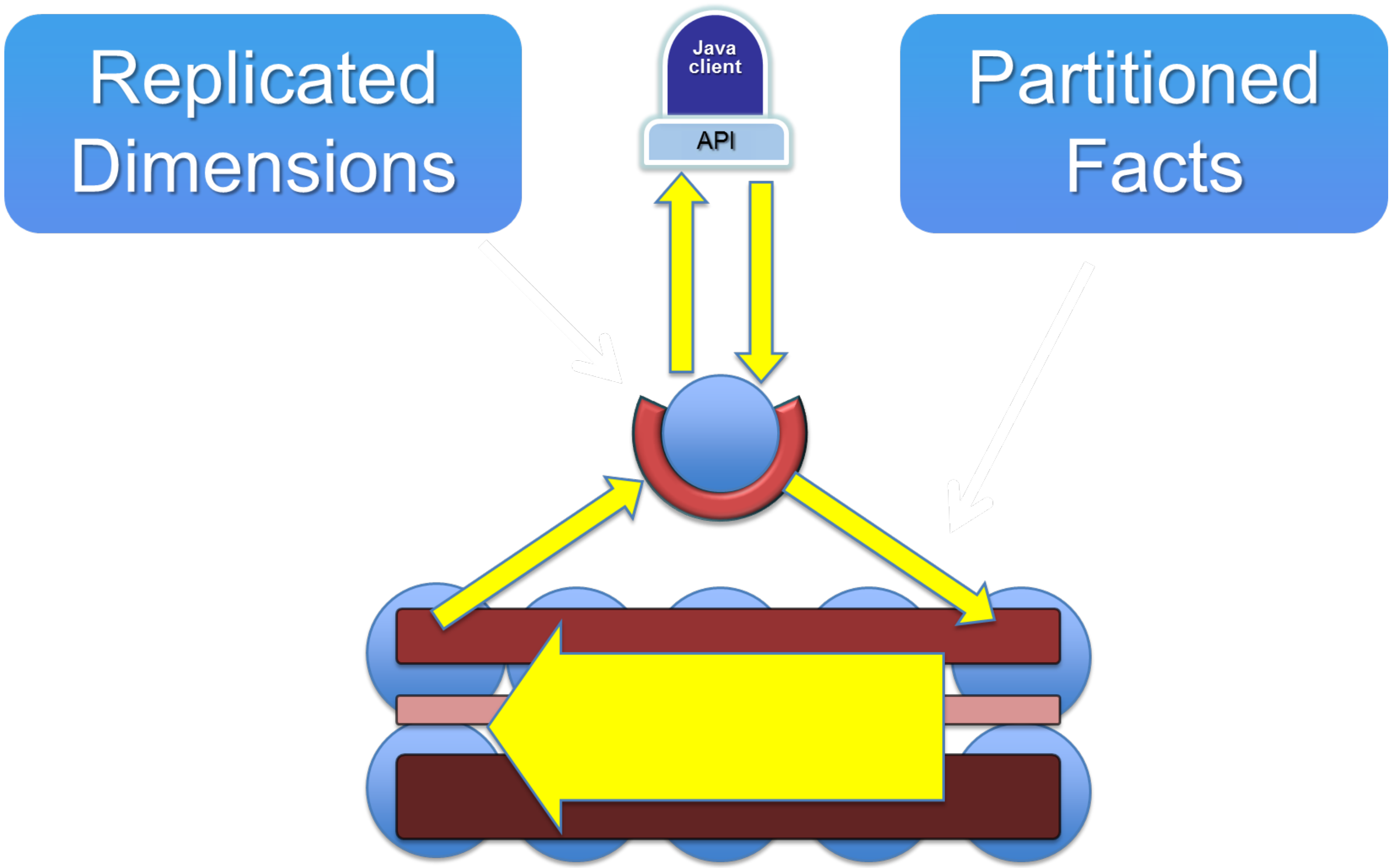
# Why Joins?

- Objects need to be versioned as changes come from many different sources
- Recompose different slices of data (for example bi-temporally)
- Fat objects (Distributed Cache) / Documents (NoSQL) don't allow this

# But the performance of arbitrary joins in a distributed architecture tend to suck!

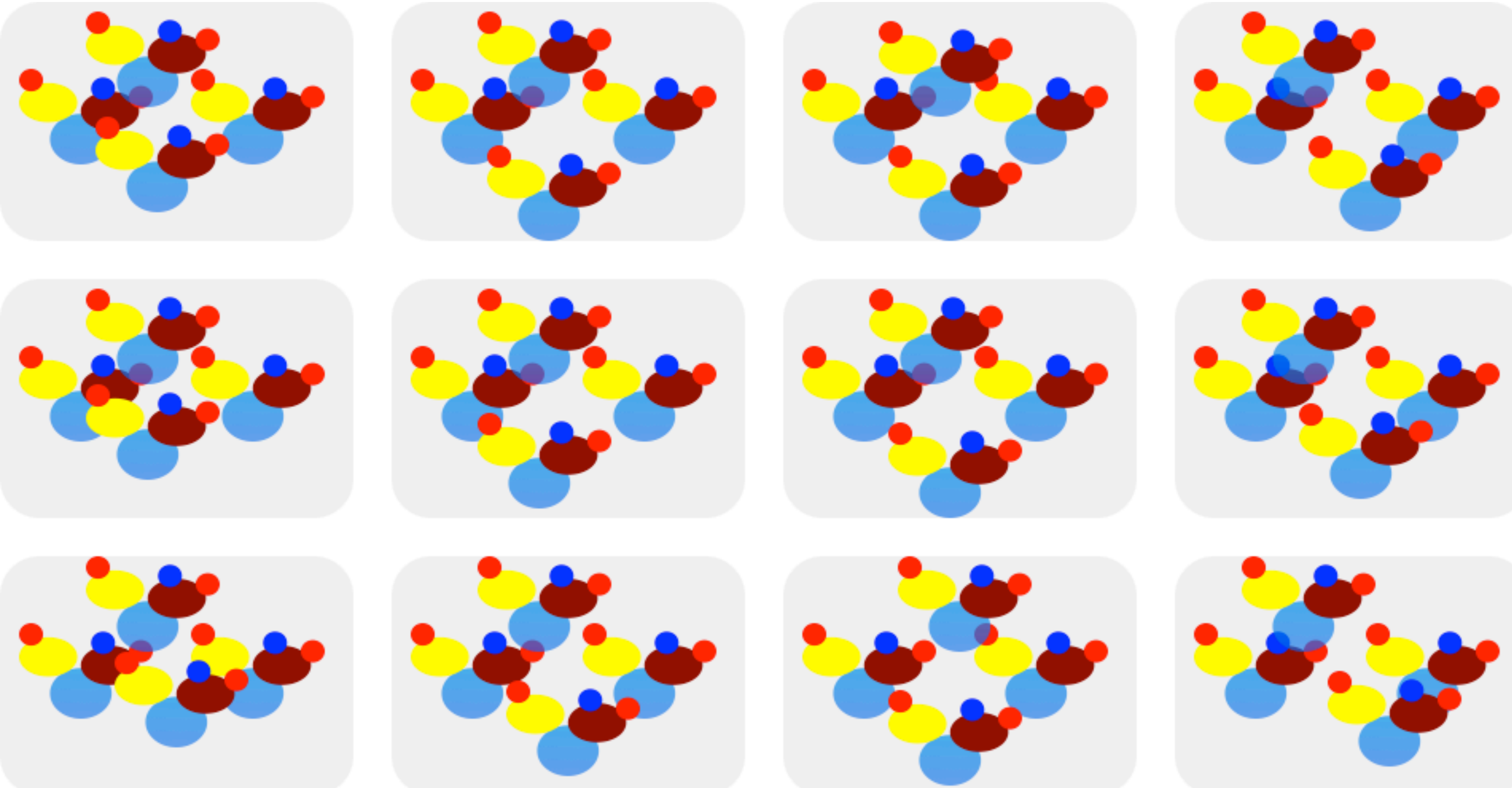


# ODC prevents this by replicating data that won’t shard with the primary key



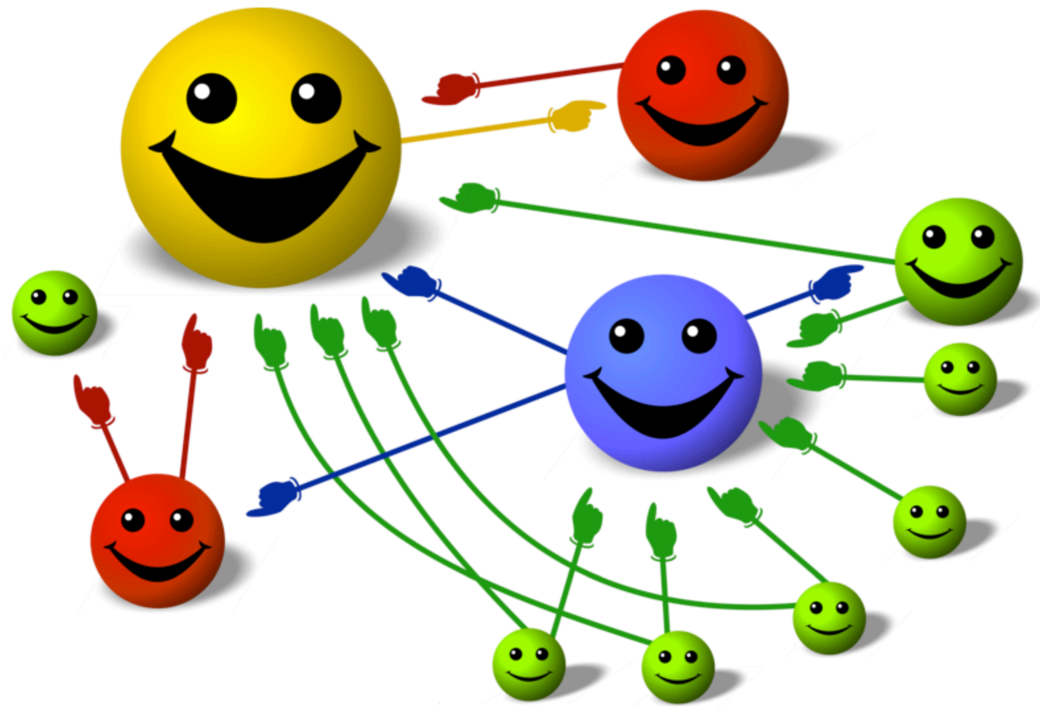
We use replication to get around the distributed join problem

# The problem with replication is that it uses up all your RAM

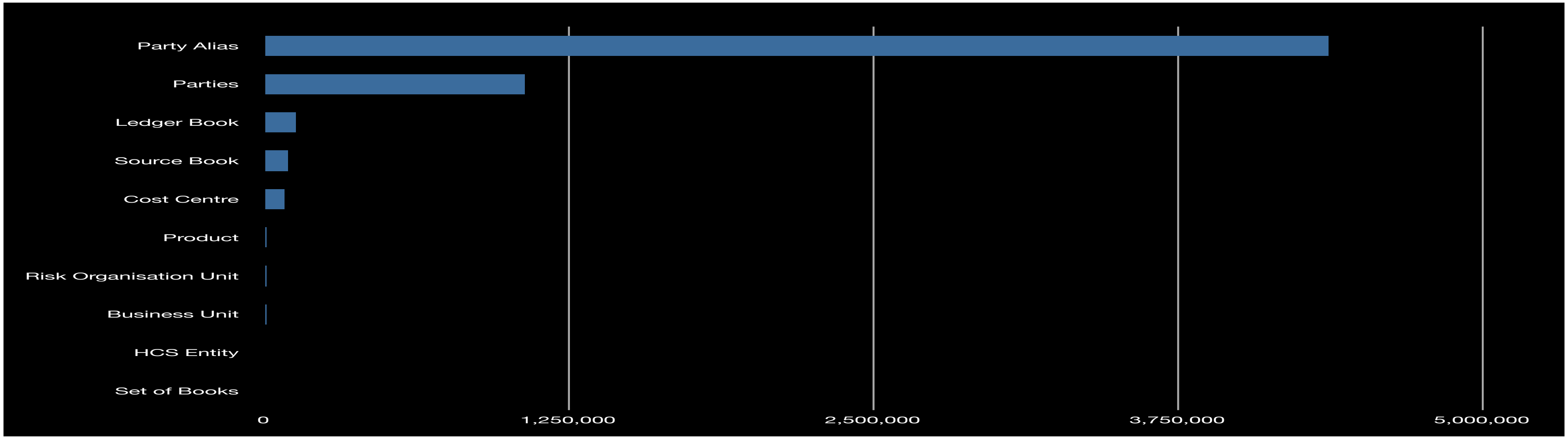


Replication eats into your storage!

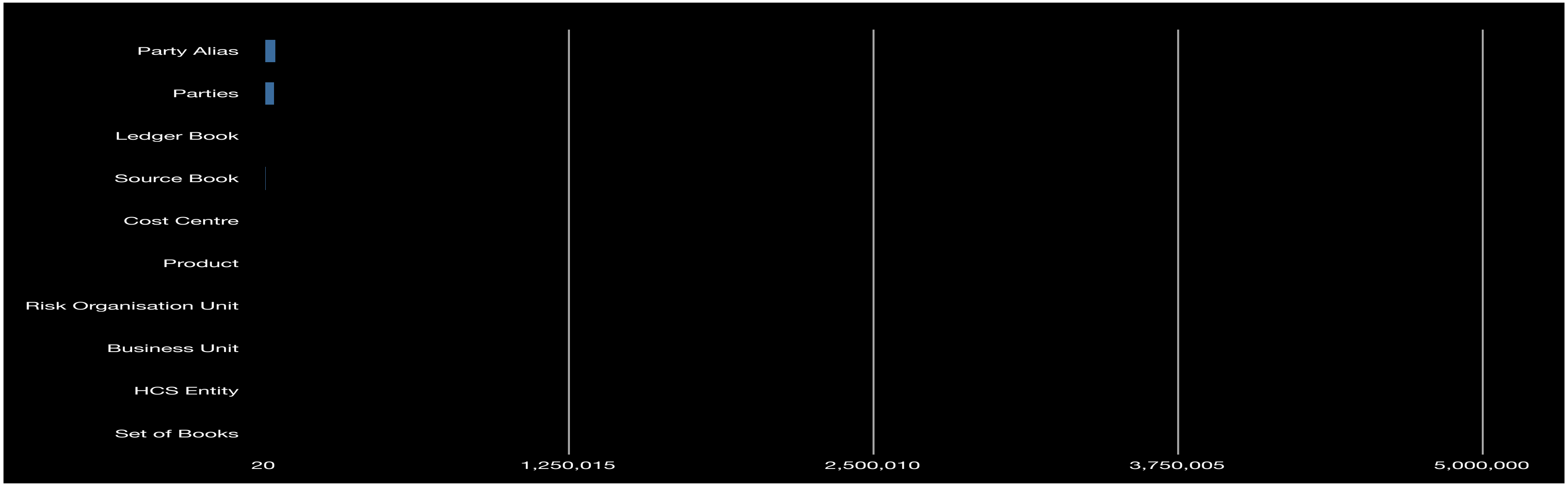
# ‘Connected Replication Pattern’ only replicate data that is referenced



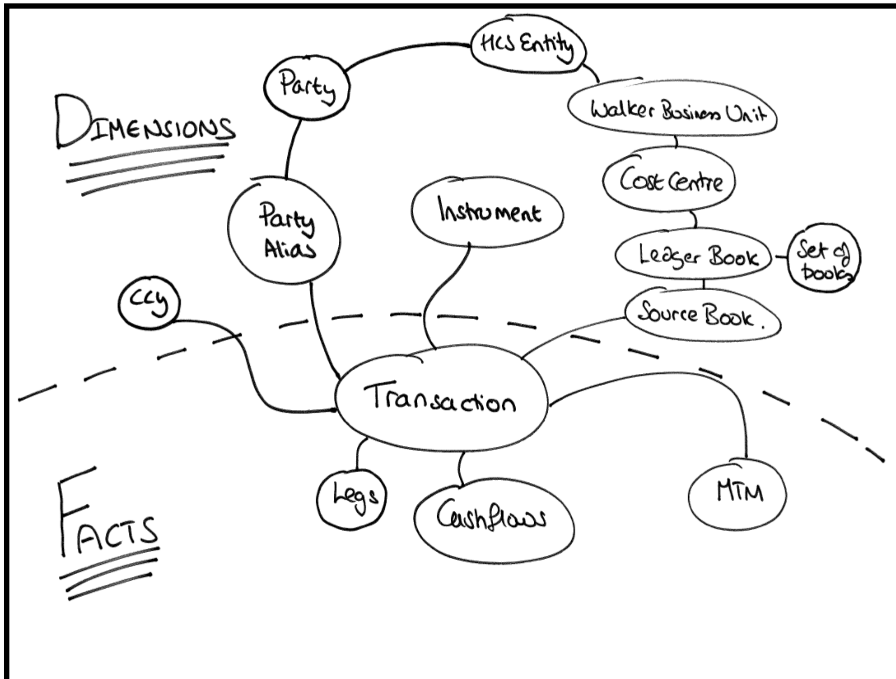
# All Data:



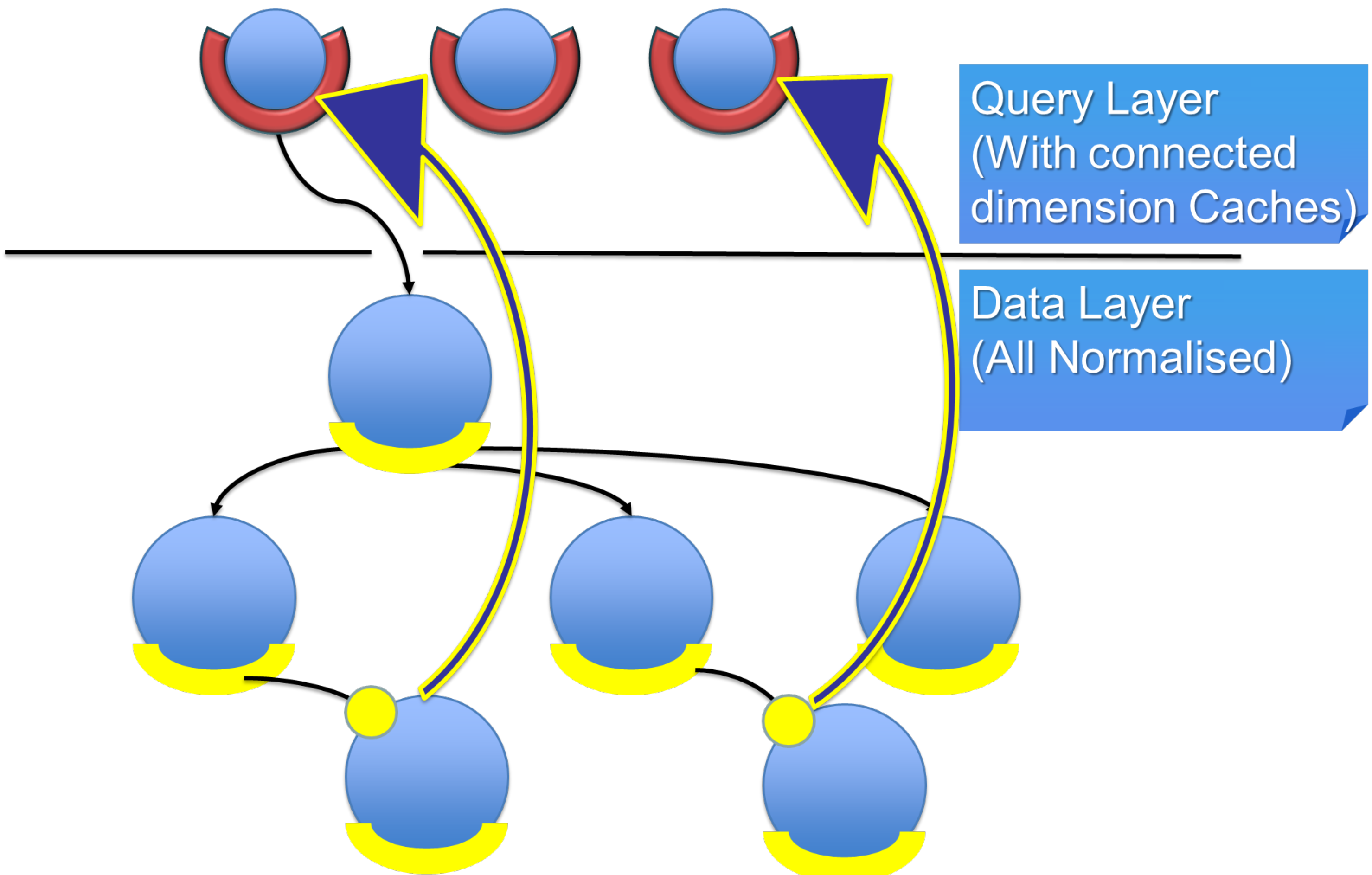

# Used Data:



# Connected Replication: Constantly track what data is used at a point in time, and only replicate this ‘used’ data

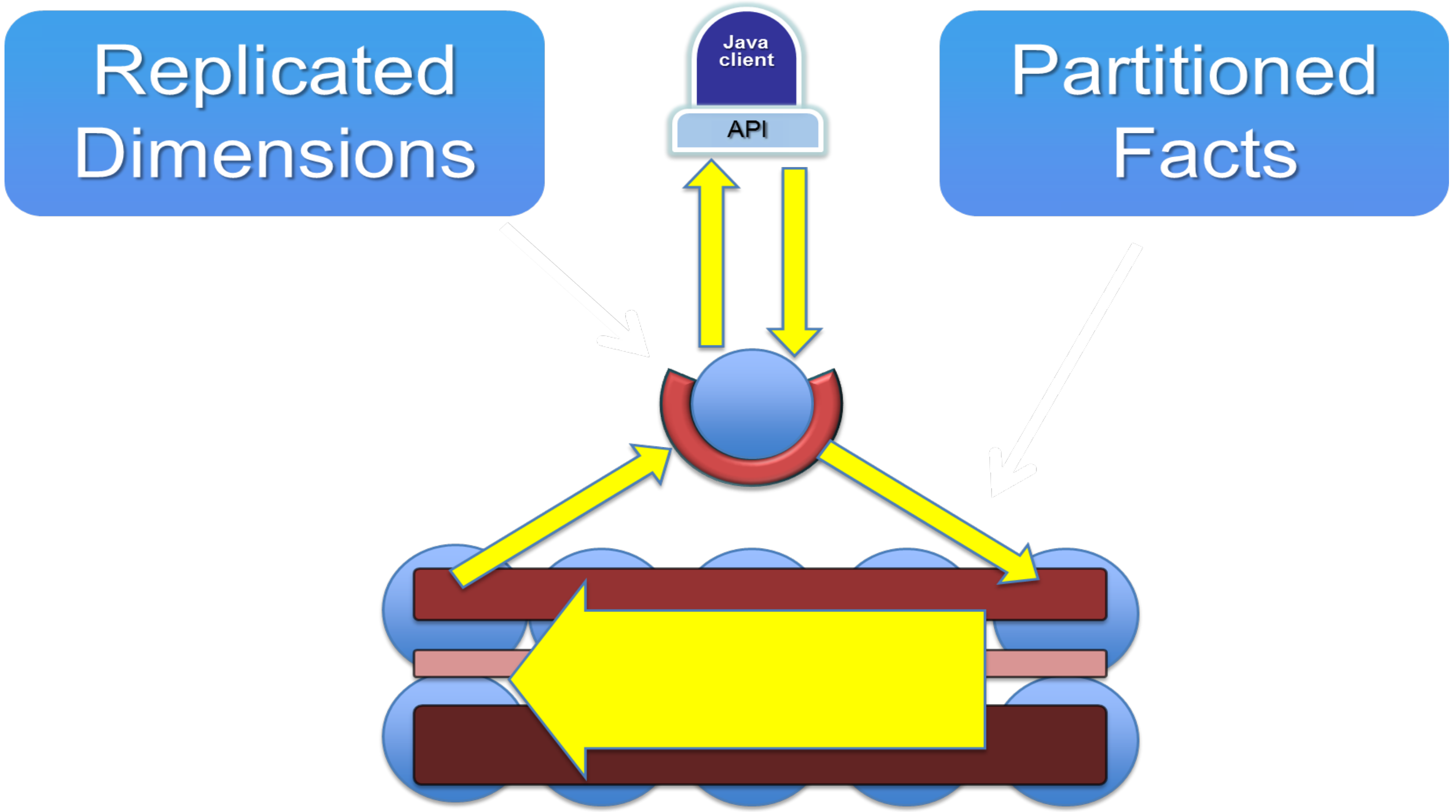


# Writes cause items to become replicated if they were previously ‘unused’



With ‘Connected Replication’ only 1/10th of the data needs to be replicated, which means we can do fast joins without eating into that much memory

# It's a two tier approach: An In-Memory database on top, with a sharded In-memory database below

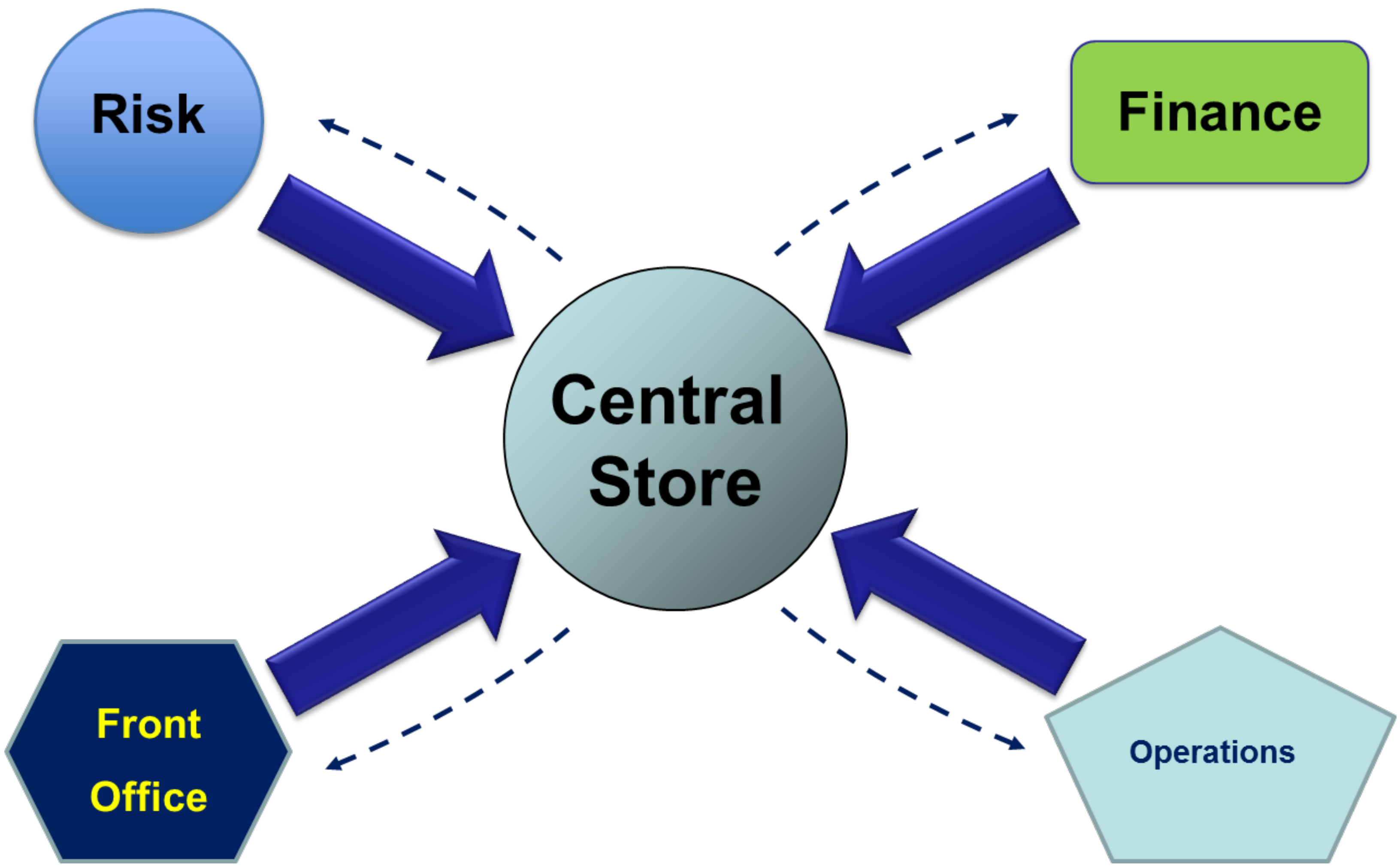


for example ODC can do a 34 join query data in a 420 node cluster in 5ms (key based) and around 20ms (broadcast)

# So ODC provides

- Scalability
- Real-time interface
- Low latency access
- High throughput (throttled)

# One centralised version of the truth!



# Data Virtualisation

Why do Data Virtualisation?

Resisting the ‘YAS – Yet Another System (…and database & feeds & code &&&)’ Syndrome.

Aspirations:

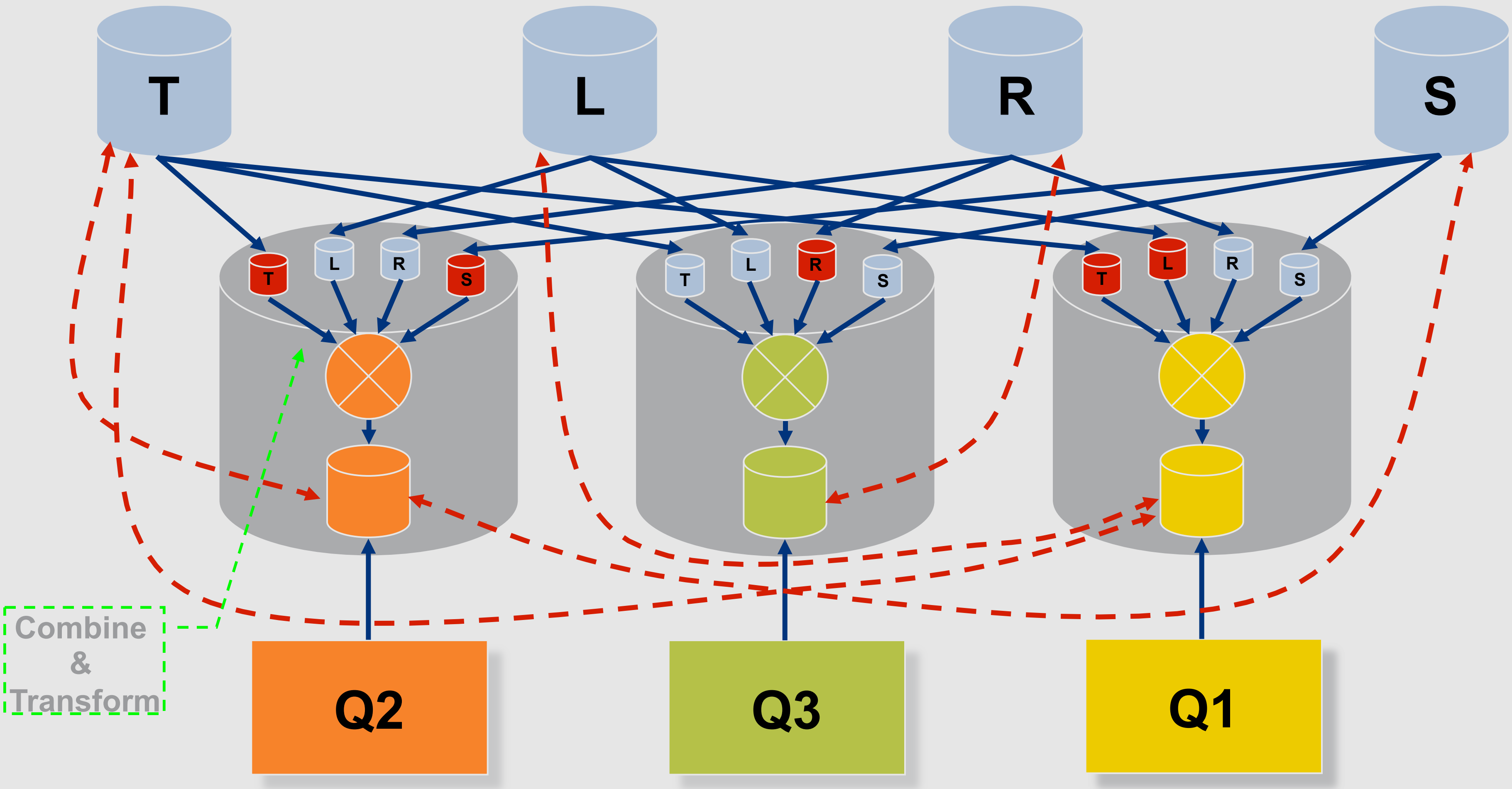
- Leave data in situ
- Define an idealized data-schema

What’s different? The DAL – is not a point-solution, it’s a strategic platform and data-source



RBS™

# YAS: Sources, Feeds, Replicas (Subsets) and Reconciliations

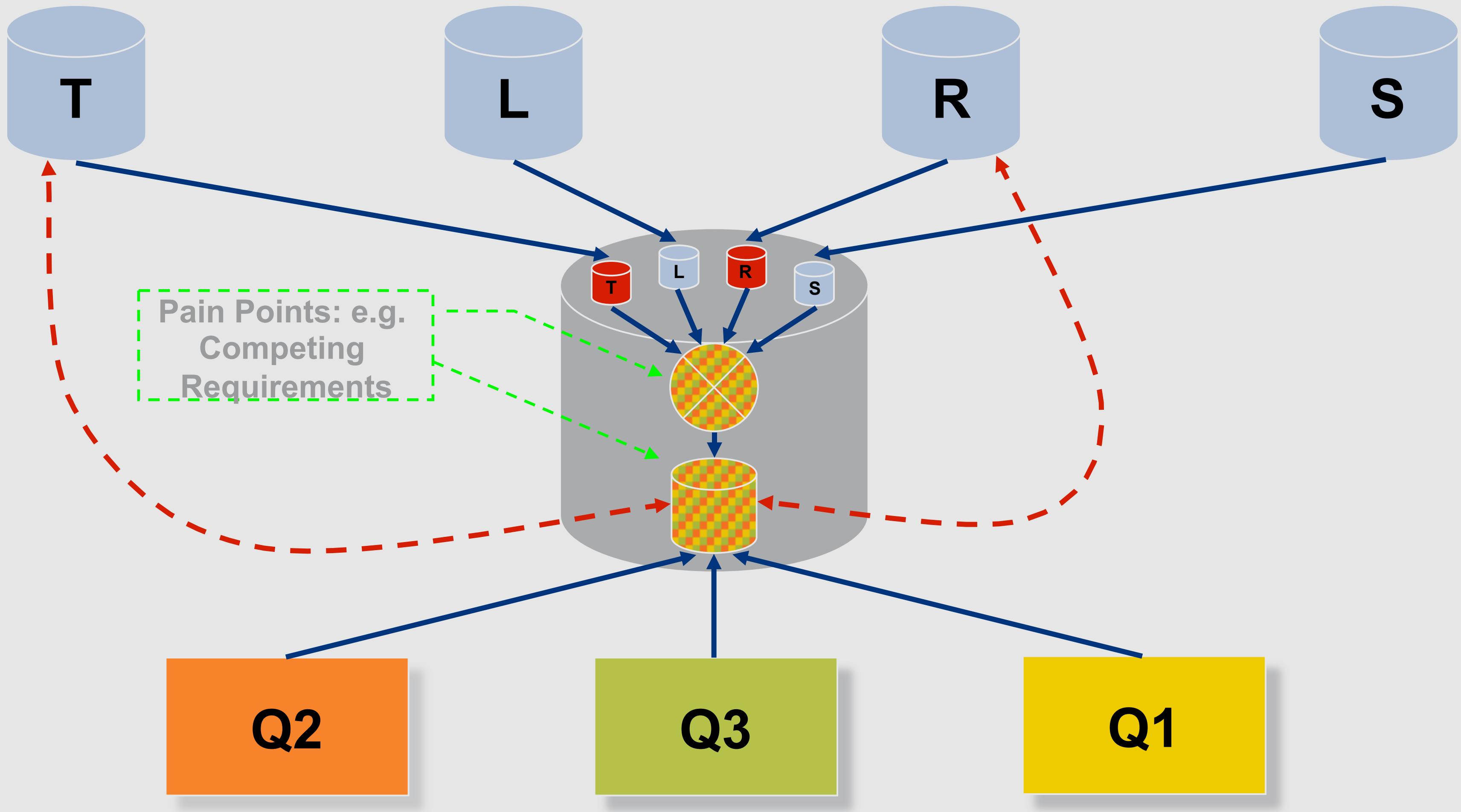





RBS™

# Does the Enterprise Data Warehouse (EDW) Help?

Co-ordination of (potentially competing) requirements & releases, reconciliations still required

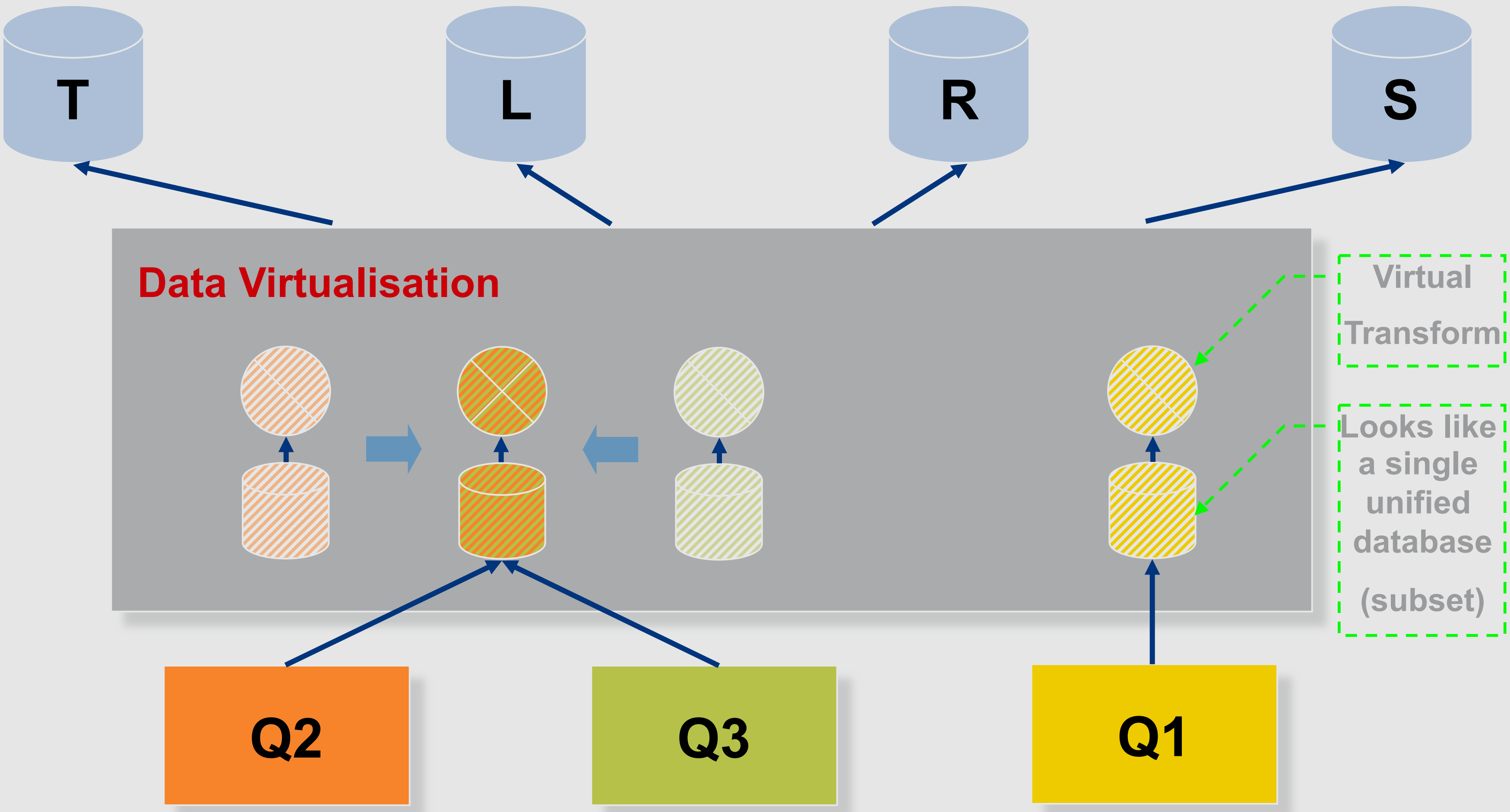





RBS™

# Data Virtualisation

…and if synergies are identified, transitions can be straightforward:



# The Opportunity to Think Differently

## However: It's not without its challenges…

**Technical:**

Distributed joins & query optimisations are getting better

Pesky physics still get in the way!

Real World™ Data Consistency and Quality

'Unusual' sources require adapter development

Some sources can't (or don't want to) be hit directly

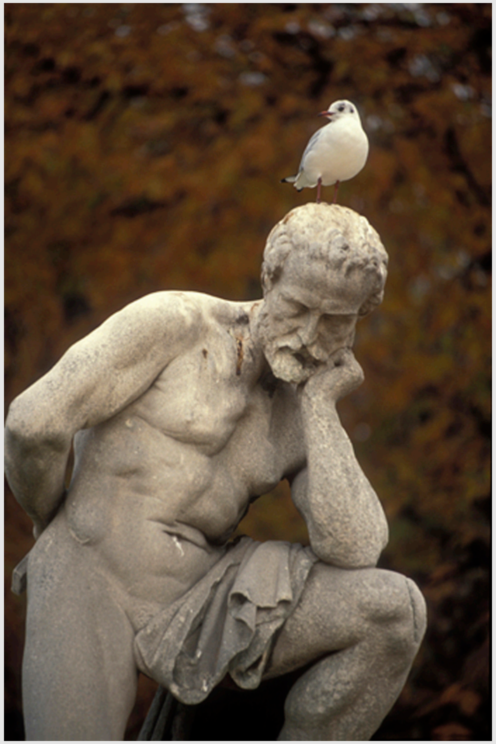
So in practice, we expect to create a hybrid **but all 'behind' the DAL**.

The developers in this space become Enterprise Data Engineers

So our aspirations are treated as our idealized design principles

**People:**

New & Unfamiliar 'Stuff' often generates its own resistance





RBS™

# Collaboration is Key

What we've learnt:

- This stuff is best shared!
- It lowers resistance to adoption.

Collaboration and Socialization is crucial:

- Guiding Principles / Thinking Frameworks
- 'Brown-Bag' Sessions
- Design & Architecture Discussions and Reviews
- Building the Network

**<u>Effort invested in these types of activity always seems to pay off!</u>**




RBS™

# Questions?