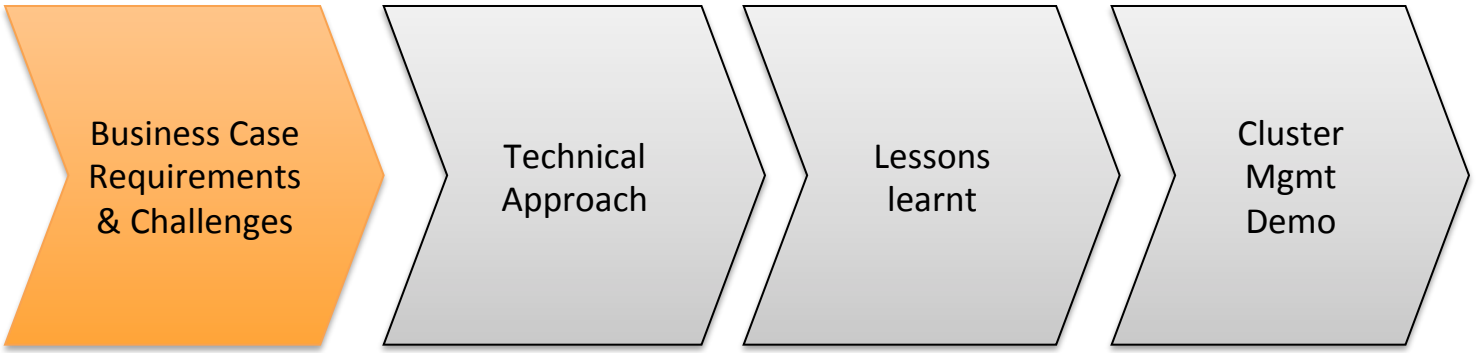
Scalable on-demand Risk and P&L

Raj Subramani
Christoph Leinemann

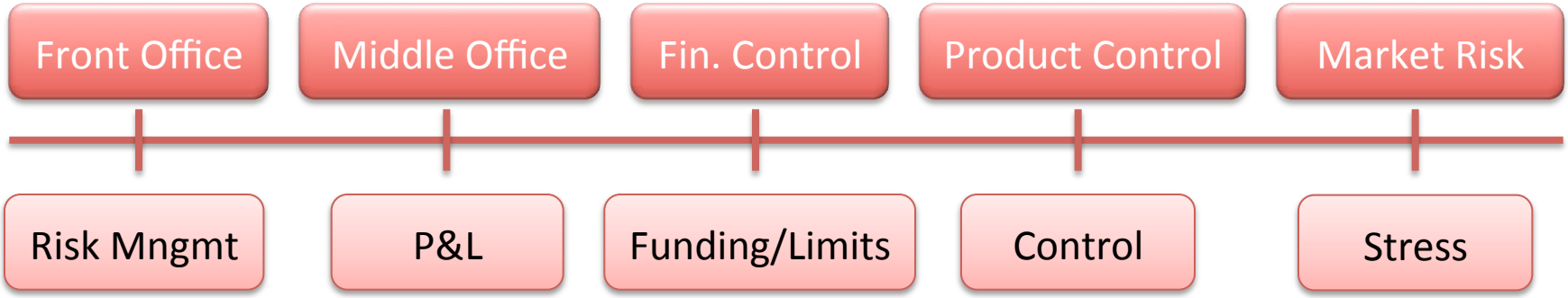
Overview

- Business Case, Requirements & Challenges
- Technical Approach
- Main Pitfalls & Lessons Learnt
- An approach to cluster management



The Business Case

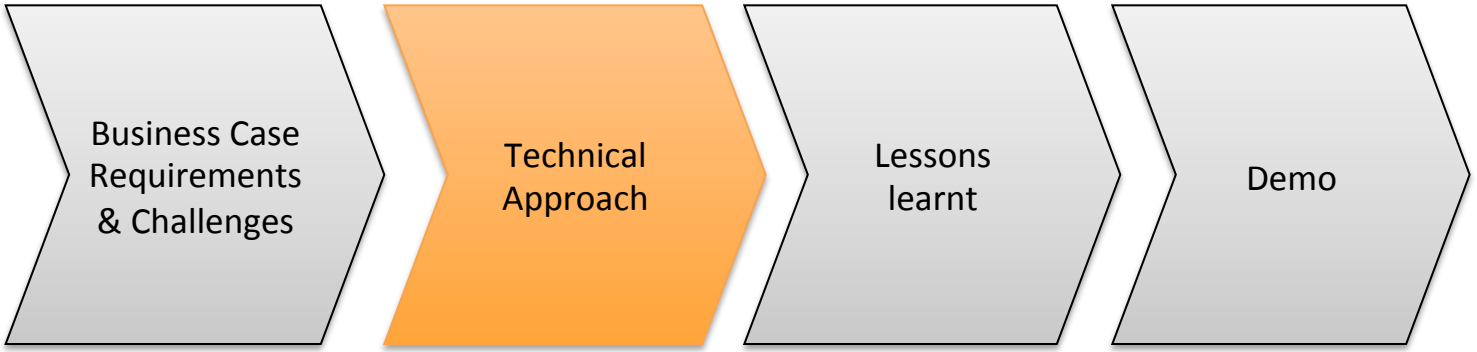
- Consistent view of data across business functions



- Consistent view across management functions

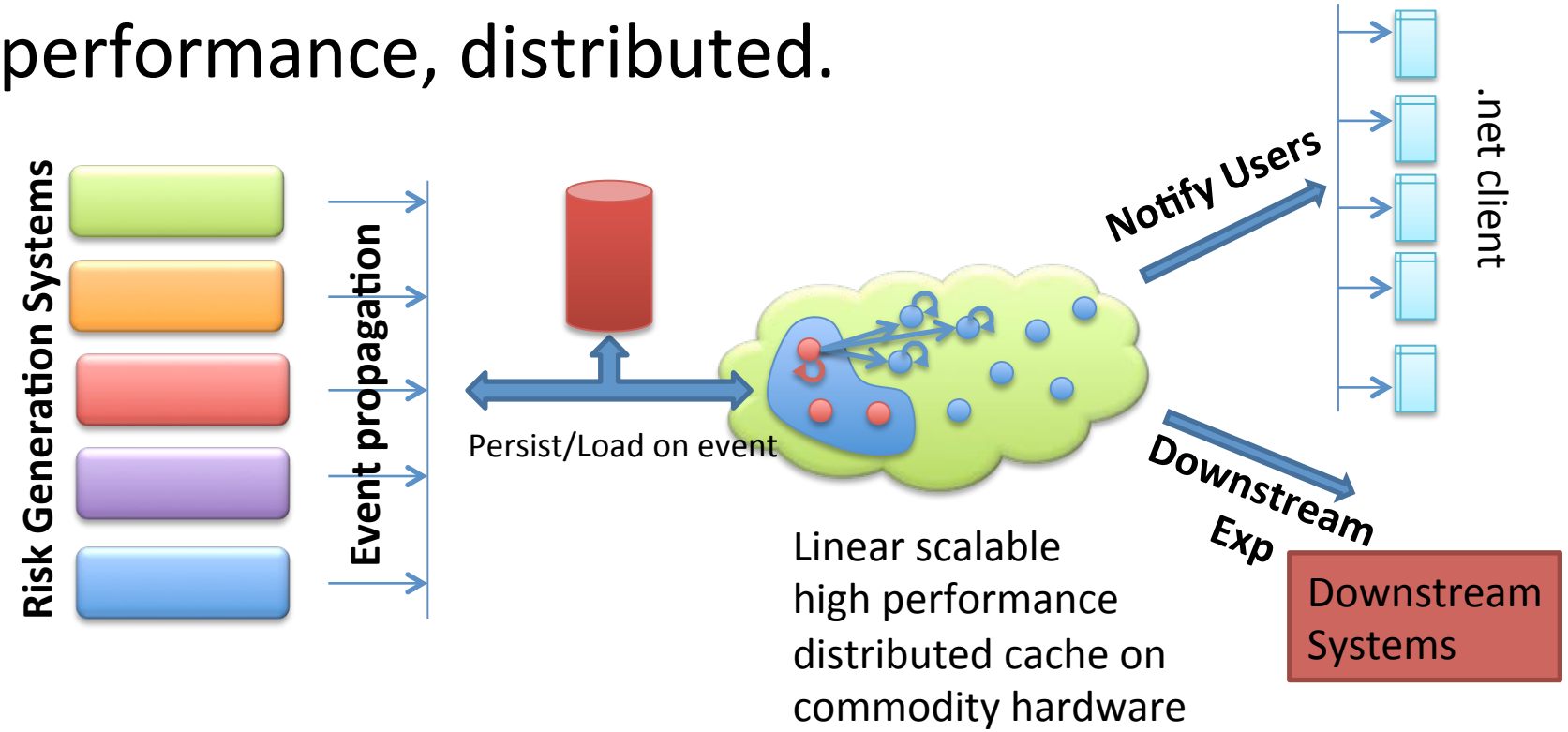
Business Requirements

- Near Real-time reporting
- Data completeness (confidence)
- Trade Level reporting and Historical Coverage
- Cross Asset compliant
- Time to market on new/changing metrics
- Complex reports (mkdata, quant, range)
- Workflow sign-off, adjustments, locking of PnL
- Automated export to downstream system


Business Case
Requirements
& Challenges
Technical
Approach
Lessons
learnt
Demo

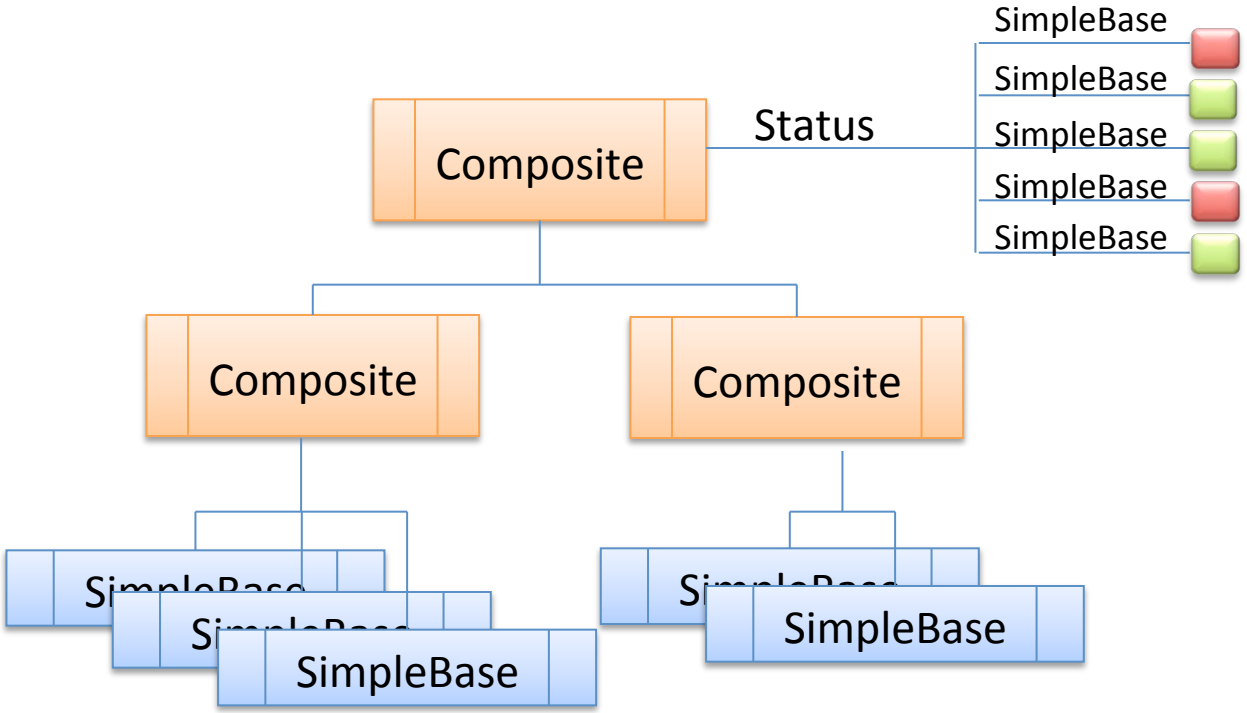
Technical Approach

- Near Real-time reporting
- Event driven data flow, scalable at constant performance, distributed.



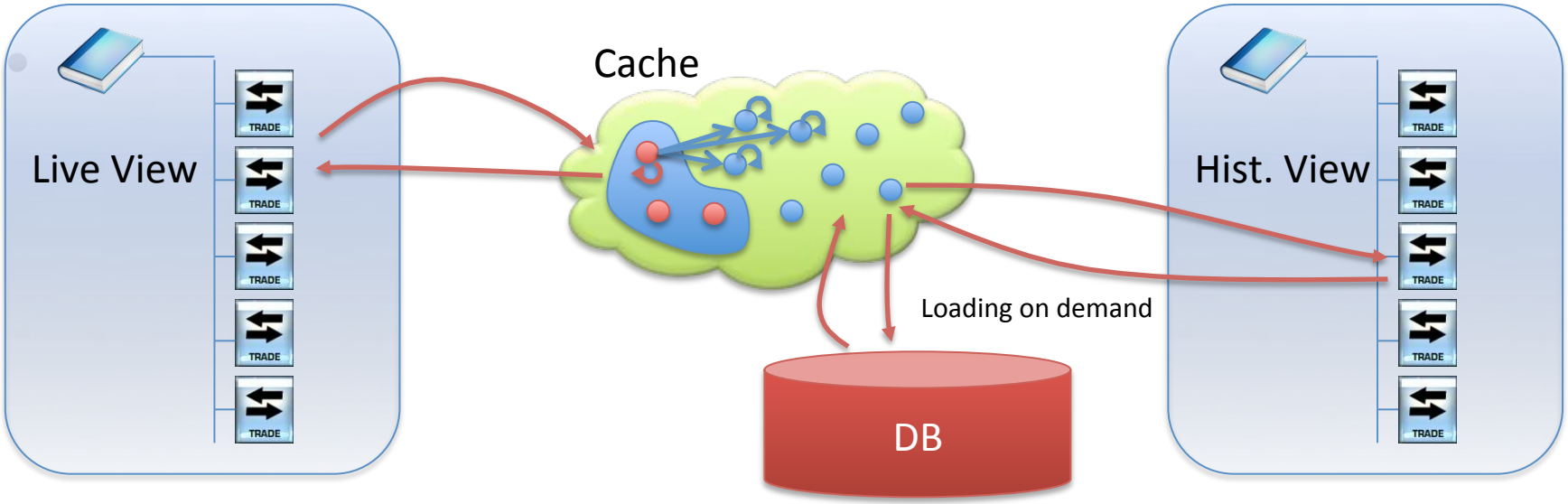
Technical Approach

- Data completeness (confidence)
- Reporting Framework. Consistent distributed tree for status handling based on hierarchical (reusable) components



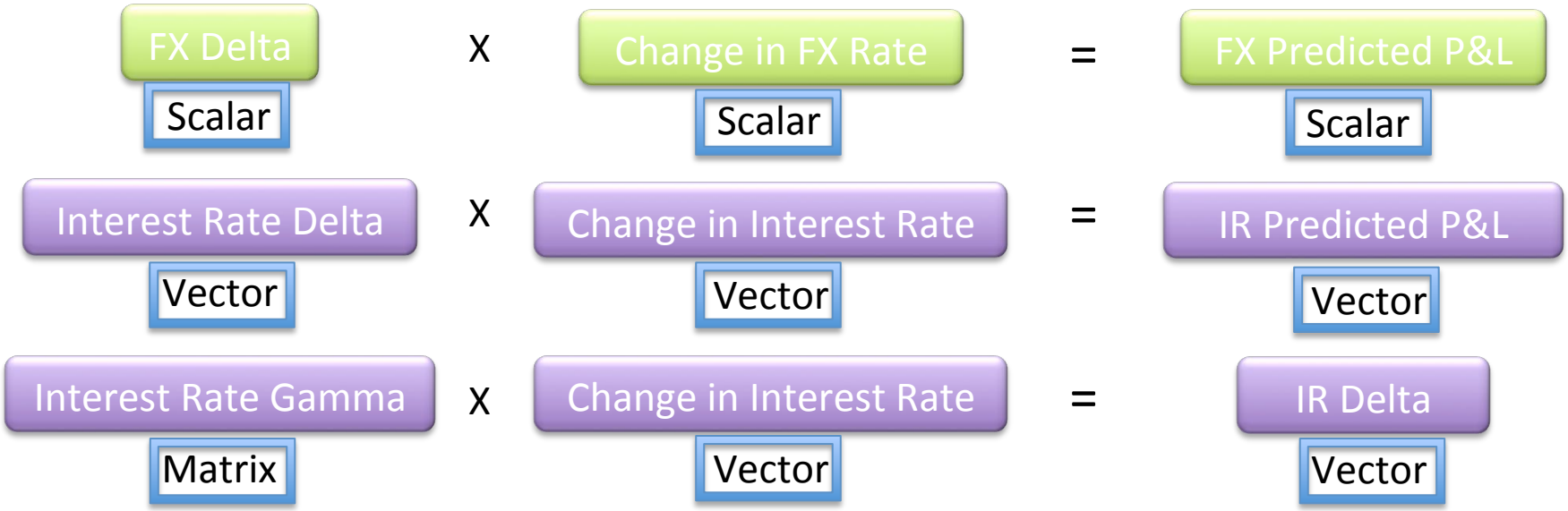
Technical Approach

- Trade Level reporting and Historical Coverage
- Same report hierarchy works at aggregated or trade level
- Same report hierarchy works on historical data
- Support for paging if result sets become too big



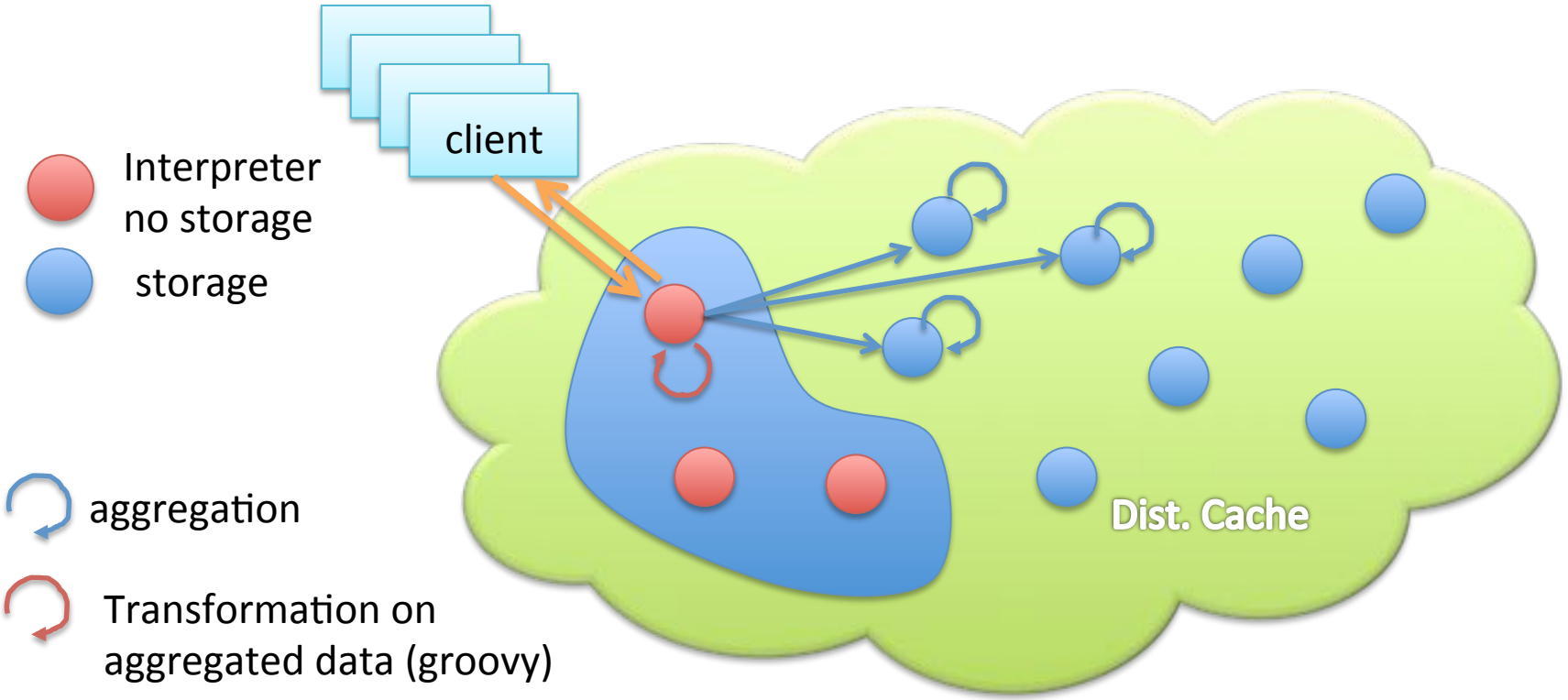
Technical Approach

- Cross Asset compliant
- Generic data model allows coverage across all asset classes (Equity, FX, Fixed Income and others)



Technical Approach

- Time to market on new/changing metrics
- Reporting module based on interpreted language and separated from the aggregations module



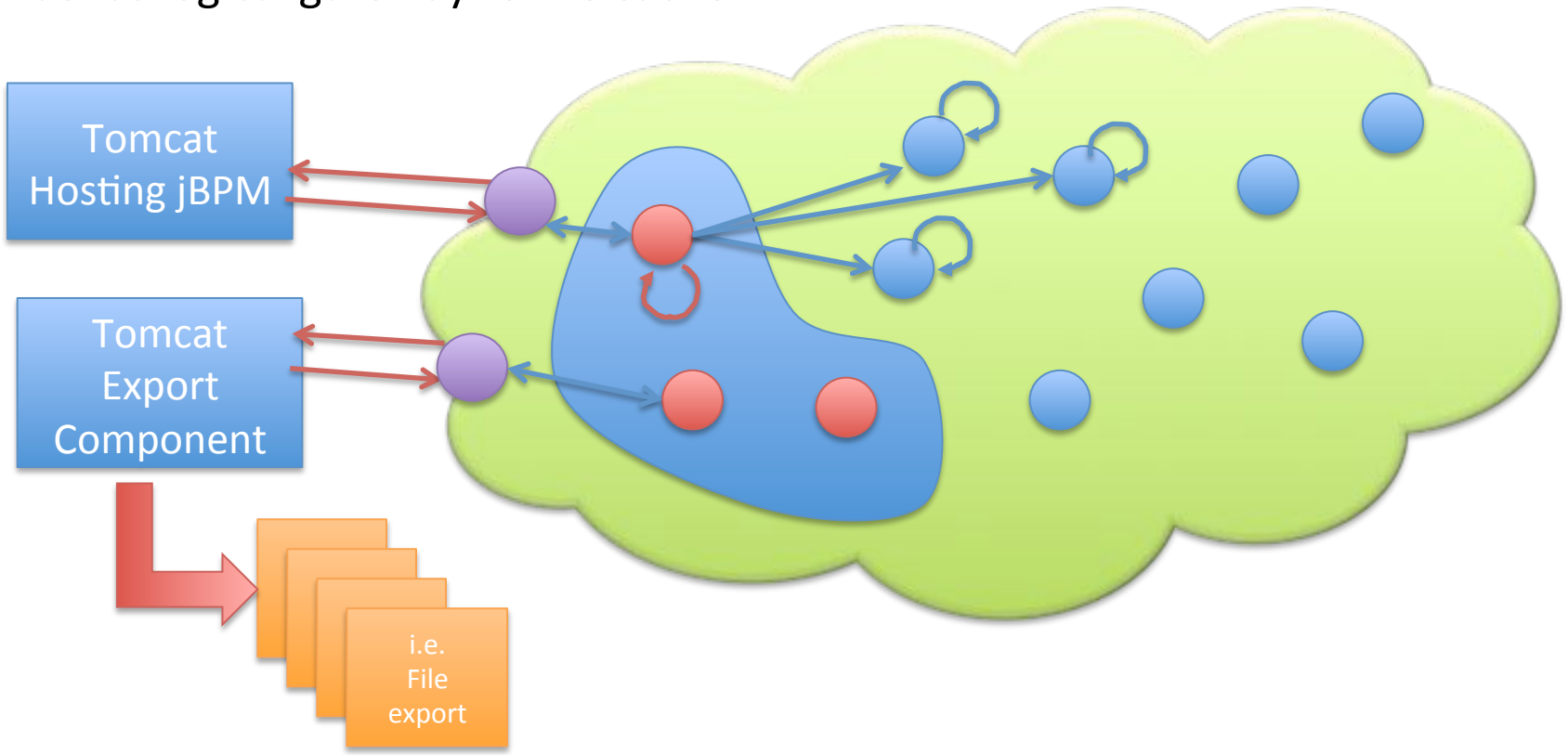
Technical Approach

- Complex reports
- Dedicated Reporting language
- XML based
- Transformations/Operations described in groovy
- Support for Market data variables
- Support for simple quantative operations
- interpreted

Technical Approach

- Workflow sign-off, adjustments, locking of PnL
- Event driven export to downstream system

Satellite Applications which integrate with other Enterprise Applications and act as logical gateway to the cache.



Performance Metrics

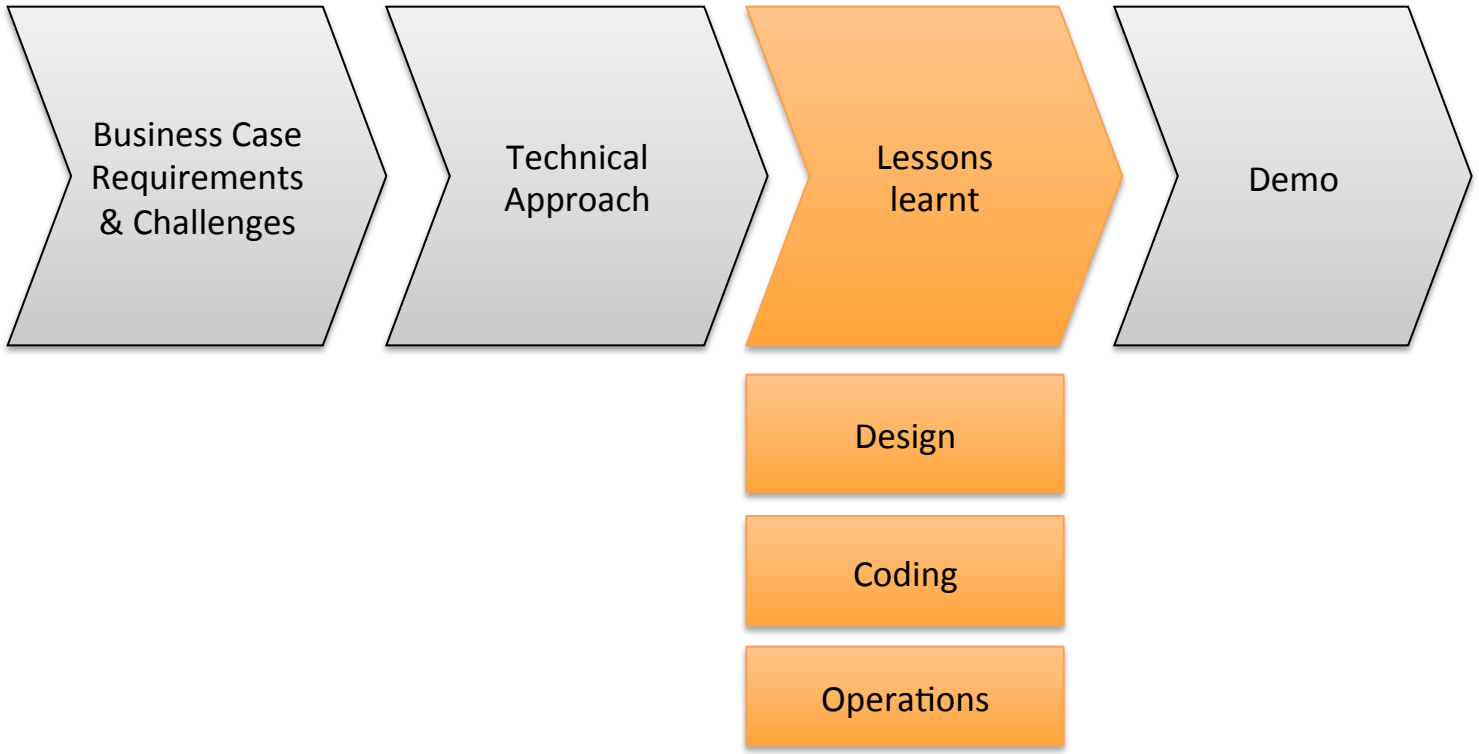
- Official T+1 PNL report
  - Comprises of 1300 underlying (composite) report components
  - Evaluates 80 (SimpleBase) measures
  - Reports across 30 dimensions
  - Generates a sliceable cube across 55000 trades in 8 seconds (included rendering time in the GUI)
  - Individual (SimpleBase) aggregation takes around 200 ms

Data Metrics

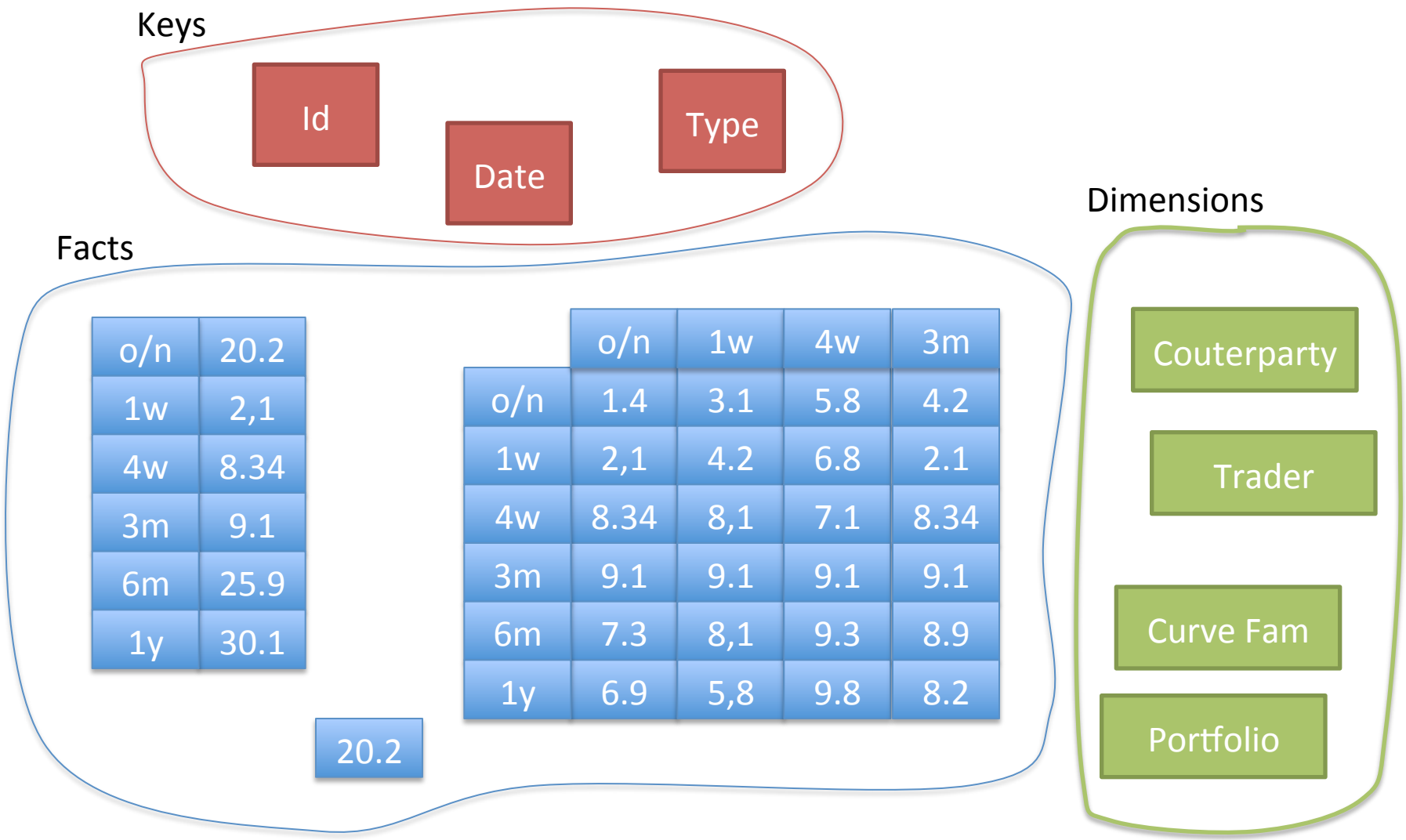
- 30 gig of data everyday
- At fingertips of Traders, Middle Office, Product Control and Downstream System
- Classical OLAP requirements
- A lot of business processes based on data
- Three production clusters
  - Around 800,000 Trades / day
  - Around 90 million risk points per day
  - Around 10 million p&l points per day
  - 20 to 70 dimensions

Summary

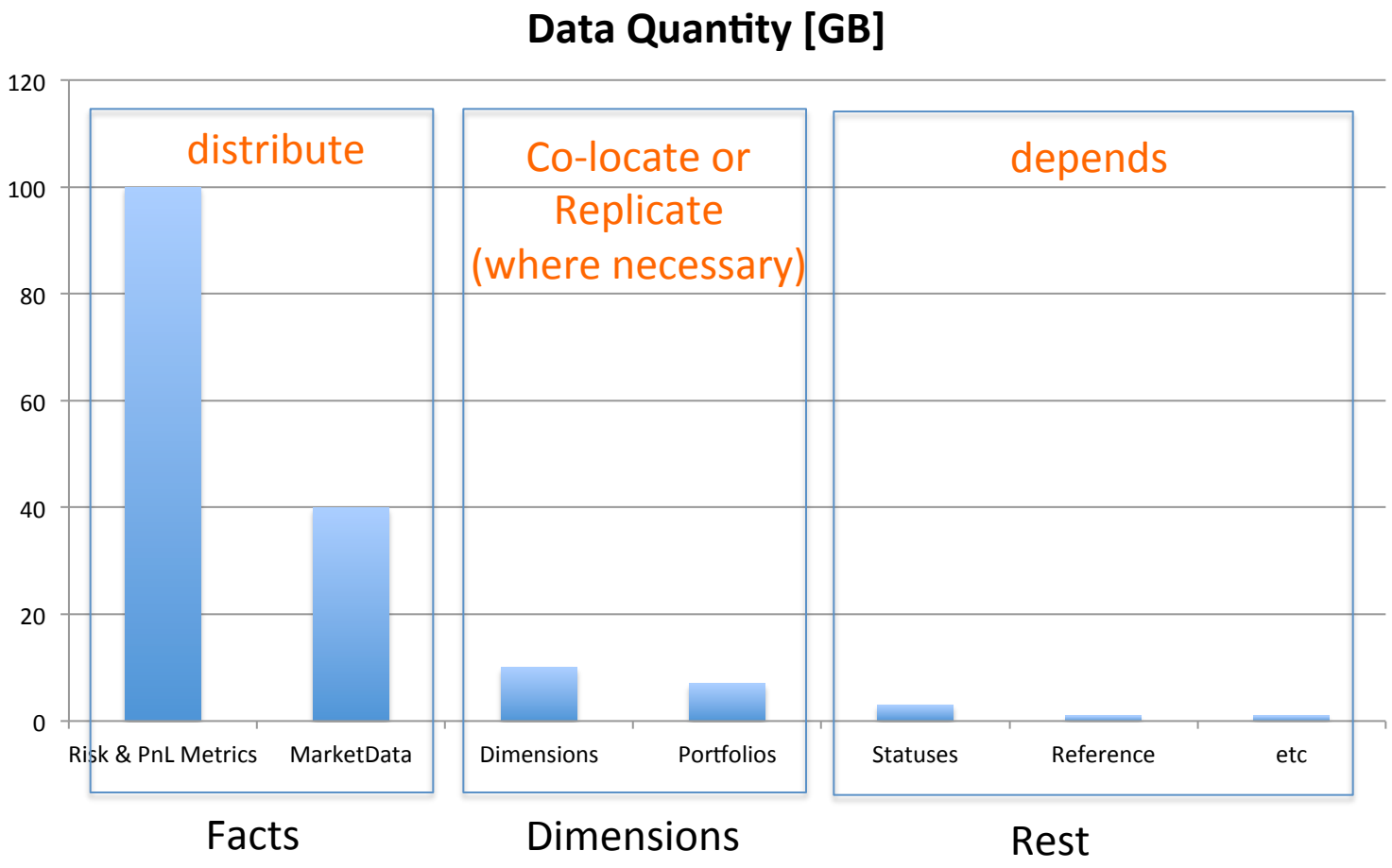
- On the fly aggregates at portfolio level
- Performant trade-level drill down
- Query Engine able to traverse the object graph
- Dimensional agnosticism
- Cross asset solution
- Multi functional reporting
- Co-location with data
- Historical and Live reporting off one framework


Business Case
Requirements
& Challenges
Technical
Approach
Lessons
learnt
Demo
Design
Coding
Operations

Design - Understanding the Data



Design - Understanding the Data



Design – how do we access data

- Primary access to cache to
  - aggregate
  - process/transform
  - compose/resolve
  - put/get
- Aggregations require flexible grouping based on dimensions

Design - Approaches to Normalization

- Near-caching to build de-normalized view. Works for aggregation and entry processors.
- Affinity and access to backing map (supported from 3.7 / unofficial before then)
- Distributed flight weight – encode your repetitive values. Be aware of your cardinality.

Design - Aspects of Co-location

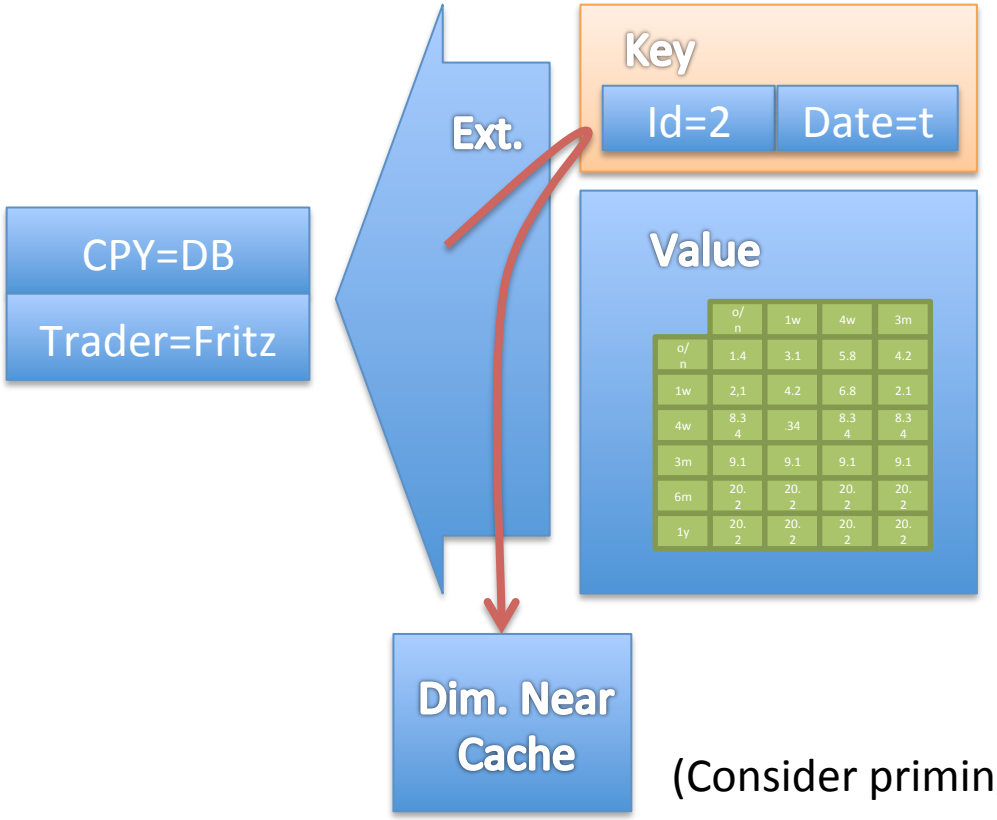
- Near Cache on storage nodes
  - Mind service dependency and invalidation policy
  - Significant communication overhead
  - Priming mechanism might be necessary
- Affinity
  - Works if part of the keys are the same
  - Mind the overall balance of your cluster
  - Referenced data still has to be de-serialized

Design– become resistant to permutations of group by criteria

- Problem
  - Indexing for GroupBy extractor
  - Access and de-serialization of dimension values.
- Solution
  - Abstraction layer to (quickly) obtain fact-related values (dimensions) on the fly
  - You do need to put mechanism in place that prevents aggregation while changing dimensions

Design – GroupBy without index

Group by Extractor uses underlying key for lookup to a near cache rather then accessing the entry



(Consider priming of near-caches)

Design – Patterns

- global counter
- distributed tree
- processing cache
- query cache
- staging cache (delta update via invokeAll)
- parallel agg/paging

Coding - Entity/Logical Code Layer

- Entity
  - Only ever abstracts one named cache
  - Use of generics
  - Massive leverage of query code and other domain specific aspects
  - Index management – don't blindly addIndex all the time as locking is involved.
  - Central point for extractable aspects of your domain objects
- Logical layer
  - one abstraction layer up from entity
  - Can reference more then one different Entity
  - Makes relationships between 'NamedCache's explicit
  - Important to understand service dependencies and designing the right topology

Coding - pitfalls

- In large clusters use Afinity and PartitionSetFilter as for some requests result set size temporarily grows proportional to number of members in your cluster
- Equals catch – equals contract of Hashmap when used as key in coherence changes. The order of serialization becomes relevant.
- Date object
- Re-entrance
- Keep your keys small
- Indexes
  - Use pof extractor where possible
  - Try to manage them in a live cycle / avoid unnecessary ‘addIndex’ calls
- When in near / replicated cache state of your objects can change
- Same applies to indexes
- Avoid temporary caches

Operational - JVM

- Less large heaps vs. more small heaps
- Minimize pause times with Concurrent Mark Sweep
- Verbose GC
- JVM lives on after OOM. It's state is unpredictable and therefore it should be killed otherwise cluster integrity is endangered
- Might want human intelligence applied before killing to avoid cascading effects.
- -XX:OnOutOfMemoryError="escalate.sh %p"
- -XX:+HeapDumpOnOutOfMemoryError

Operational - disk

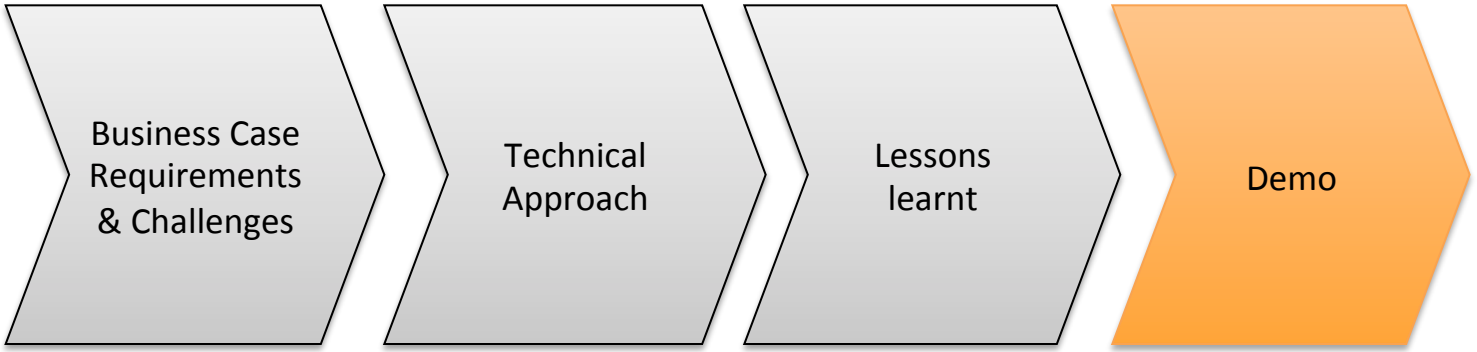
- Ideally you want shared drive for deployment and log analysis – but not at any cost as performance is crucial.
- Slow i/o can be subtle killer

```
java.lang.Thread.State: RUNNABLE
  at java.io.UnixFileSystem.getBooleanAttributes0(Native Method)
  at java.io.UnixFileSystem.getBooleanAttributes(Unknown Source)
  at java.io.File.isFile(Unknown Source)
```

- As test copy to your log/installation dir with a ceratin SLA
- Can lead to interesting conversation with infrastructure guys.
- Up until 5.x full file system could have serious effects on your cluster

Operational - MBeans

- Avoid registration through code
- Use XML config files
- Separation between
  - Utility/config beans relevant to certain node types
  - Stats and introspective beans
  - Utility Beans right-angled to node responsibilities
- One xml config per node type that is set in your launch config.
- Core jmx metrix
  - Max task backlog
  - Prune cycles
  - Hit rates
  - Communication stats

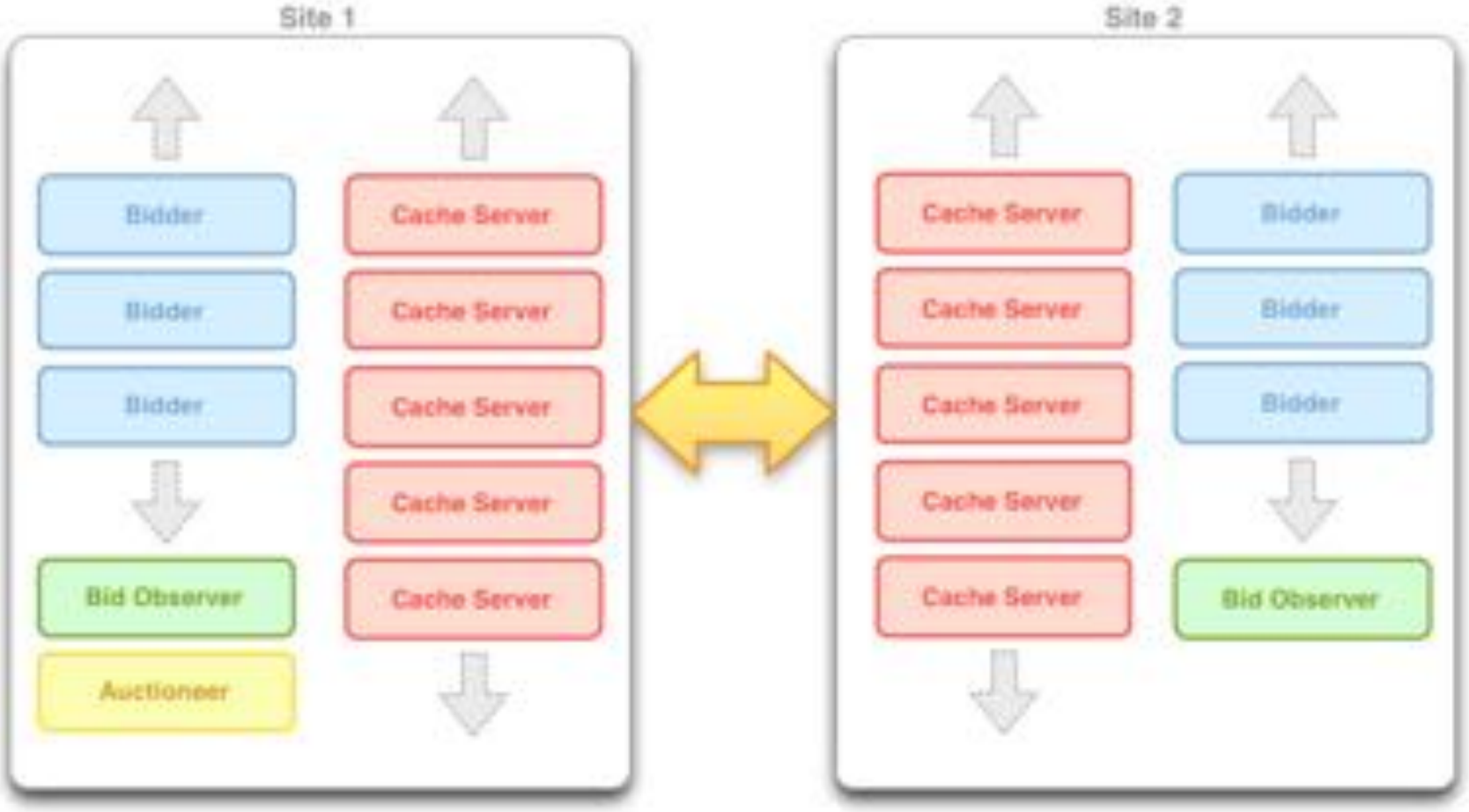

Business Case
Requirements
& Challenges
Technical
Approach
Lessons
learnt
Demo

Demo – Prototype architecture

- Separation of cluster model from actions like starting, stopping, thread dumps
- Management-cluster in parallel to managed cluster
- One node per machine
- Storing only minimal data
- Latency not an absolute priority – hence possible to run across regions.
- Remote API exposing cluster action
  - Allowing GUI implementation
  - Support for dynamic scaling
  - Other intelligent event driven cluster-actions

Example App

Incubator Auction example



Wish list

- First of all - thanks for making a great product!
- Make object local backing map part of product
- Make management easier
- Continuous aggregation
- Index declaration via config
- Generics

Thank You!

raj@subramani.com

christoph@leinemann.name