

High Performance Risk Calculations with Coherence

Oliver Buckley-Salmon
Fixed Income Solution Architecture Manager
&
Dhileep Paski
Fixed Income Solution Architect

oliver.salmon@lchclearnet.com

Contents

- About LCH.Clearnet
- About Value at Risk (VaR)
- A Model For Implementing a VaR Calculator in Coherence
- Performance
- Summary

About LCH.Clearnet

- LCH.Clearnet is the leading independent clearing house, serving major international exchanges & platforms, as well as a range of OTC markets
- It clears a broad range of asset classes including: securities, exchange traded derivatives, energy, freight, foreign exchange derivatives, interbank interest rate swaps, credit default swaps & euro & sterling denominated bonds & repos
- As a clearing house, LCH.Clearnet sits in the middle of a trade, assuming the counterparty risk involved when two parties (or members) trade. When the trade is registered with LCH.Clearnet, it becomes the legal counterparty to the trade, ensuring the financial performance; if one of the parties fails, LCH.Clearnet steps in

About LCH.Clearnet – Cont.

- LCH.Clearnet is systemically important to the functioning of global markets
- It is legal counterparty to around 50% of the global interest rates swaps market managing positions in excess of \$280 trillion notional, some of which extend out 50 years
- It clears €13 trillion of cash bonds & repo trades per month
- LCH.Clearnet clears for the LSE, LME, LIFFE & Euronext markets in cash & listed financial & commodity derivatives
- It also clears cash equities & CFDs for various MTFs such as BATS, Turquoise, Chi-X etc
- On a day to day basis it sits on between €50-100 billion of margin

About Value at Risk (VaR)

- Value at Risk (VaR) is a widely used risk measure of the risk of loss on a specific portfolio of financial assets
- Formally VaR can be defined as
 - Given a confidence level $\alpha \in (0, 1)$, the VaR of the portfolio at the confidence level is given by the smallest number l such that the probability that the loss L exceeds l is at most $(1 - \alpha)$
 - Mathematically, if L is the loss of a portfolio, then $\text{VaR}_\alpha(L)$ is the level α -quantile
- In plain English, for a given portfolio, probability & time horizon, VaR is defined as a threshold value such that the probability that the mark-to-market loss on the portfolio over the given time horizon exceeds this value is the given probability level
- There are many ways to calculate VaR, such as Delta Gamma, Historical Simulation, Variance/Co-Variance, Boot Strapped & Monte Carlo

About Value at Risk (VaR) Cont.

- All the methods give the loss (or expected loss) at a given probability level but the methods to get to that result have advantages & disadvantages
- Parametric VaR models such as Delta Gamma & Variance/Co-variance have the advantage of a fast calculation time, but at a sacrifice of accuracy
- Historical Simulation can be computationally expensive if the number of scenarios is high & requires large amounts of historical market data
- Monte Carlo Simulation is computationally very expensive but doesn't require any market data
- Bootstrap VaR has the honour of being computationally very expensive & requiring large amounts of historical market data

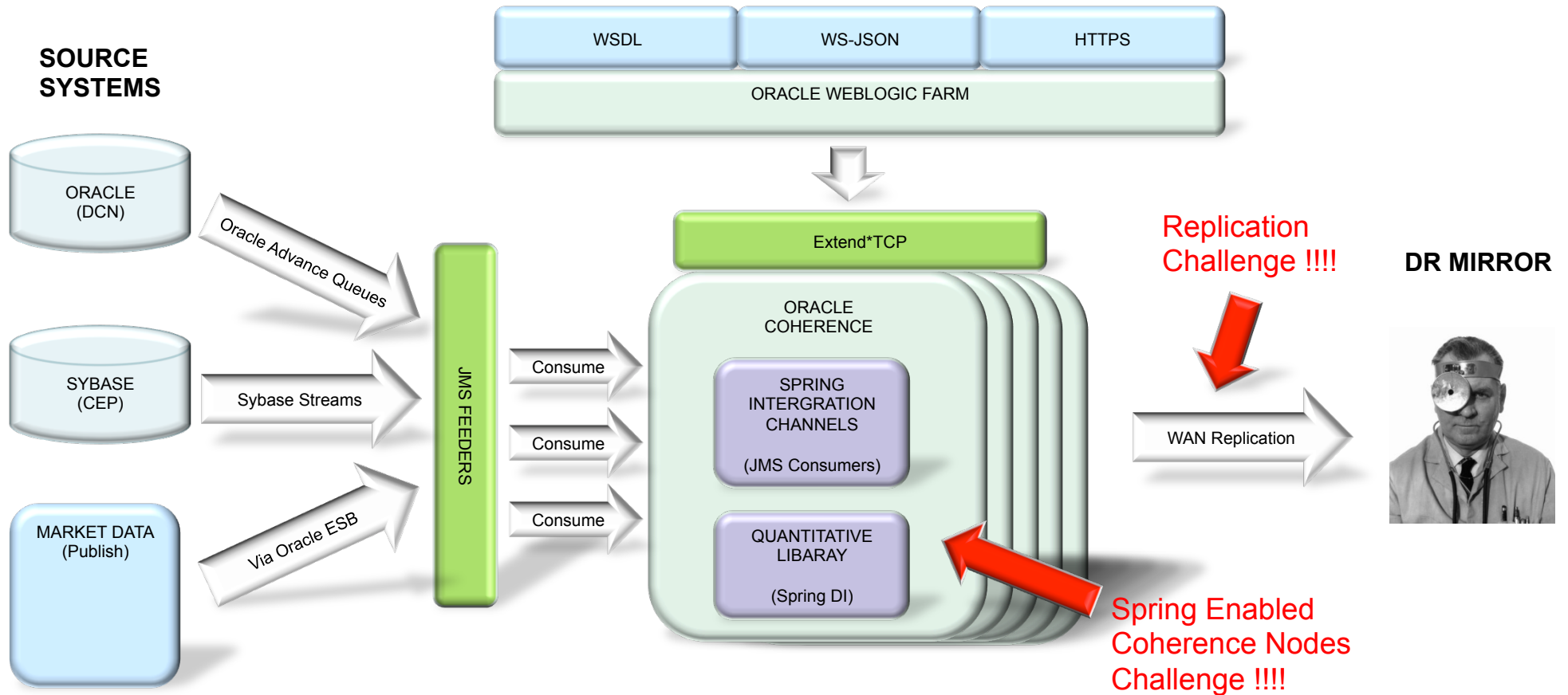
About Value at Risk (VaR) Cont.

- For this presentation we will be considering Filtered Historical Simulation as the method to get to the VaR, it is calculated via the following steps
 - Set the VaR parameters
 - Number of scenarios (e.g. 2500)
 - Holding period (e.g. 5 days)
 - Confidence level (e.g. 99.5%)
 - Acquire Position/Trade Data, Market Data (Prices, Curves etc) & Reference Data (Instrument reference data, Portfolio hierarchies etc)
 - Generate the scenarios from the historical market data
 - Re-value the positions/trades using the scenario shifts and store the results
 - Organise the portfolio losses into a distribution curve and store it
 - Statistically calculate (i.e. via standard deviation) the loss at the required confidence level (VaR)
 - Store & report VaR
 - Make intermediate results available for analysis

About Value at Risk (VaR) Summary

- As can be seen, these steps require software functionality that is conveniently exactly the same as Coherence offers
- Very fast access to large amounts of stored data
- Parallel execution of arbitrary code against an entire grid of data
- Parallel aggregation across an entire grid data
- An event driven programming model






Implementing a VaR Calculator in Coherence



Implementing a VaR Calculator in Coherence

- Bootstrap Mode
 - Uses Aggregators and Processors
 - Used when the cache is initially seeded with data
 - Disable backing map listeners
- Realtime Mode
 - Notion of “Hierarchical” caches. (Pyramids)
 - Used to propagate trade update and market data update calculations to “cache stack”
 - Uses backing map listeners

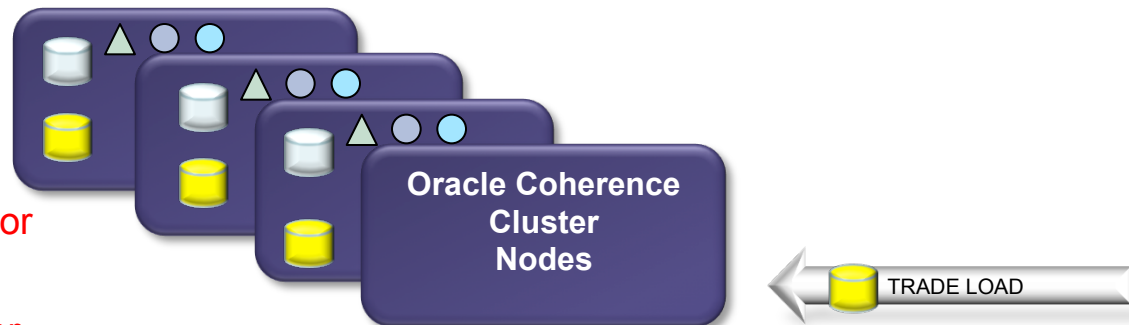
Bootstrap Mode – Aggregate and Process

- Aggregate data as required for processing
 - Collate data from  (Trade cache) into  (PortfolioRiskBucket cache).
 - Use a processor (locking) to facilitate this. 
- Send process to co-located data for calculation
 - Invoke the mark to market  (MTM) calculation on PortfolioRiskBucket cache entries and populate MTM cache.
 - Invoke the VaR  calculation on the MTM cache entries to populate VaR cache.




 • Trade Processor

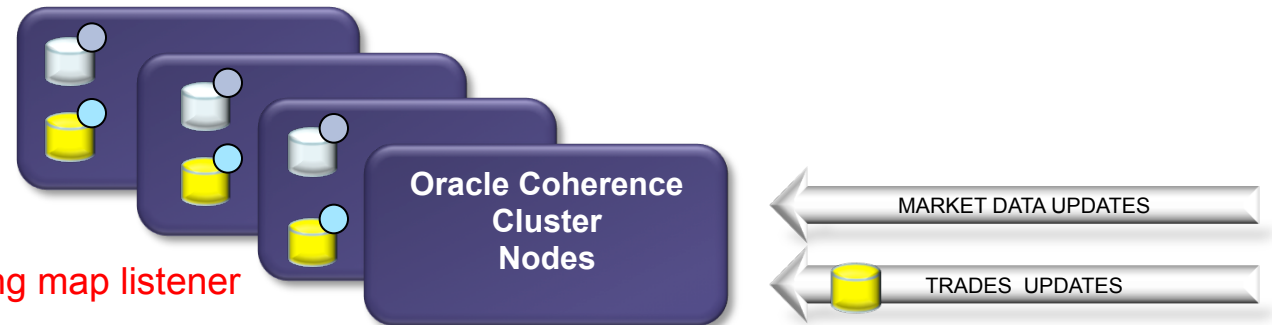
 • MTM Calculator Aggregator



 • VaR Calculator Aggregator



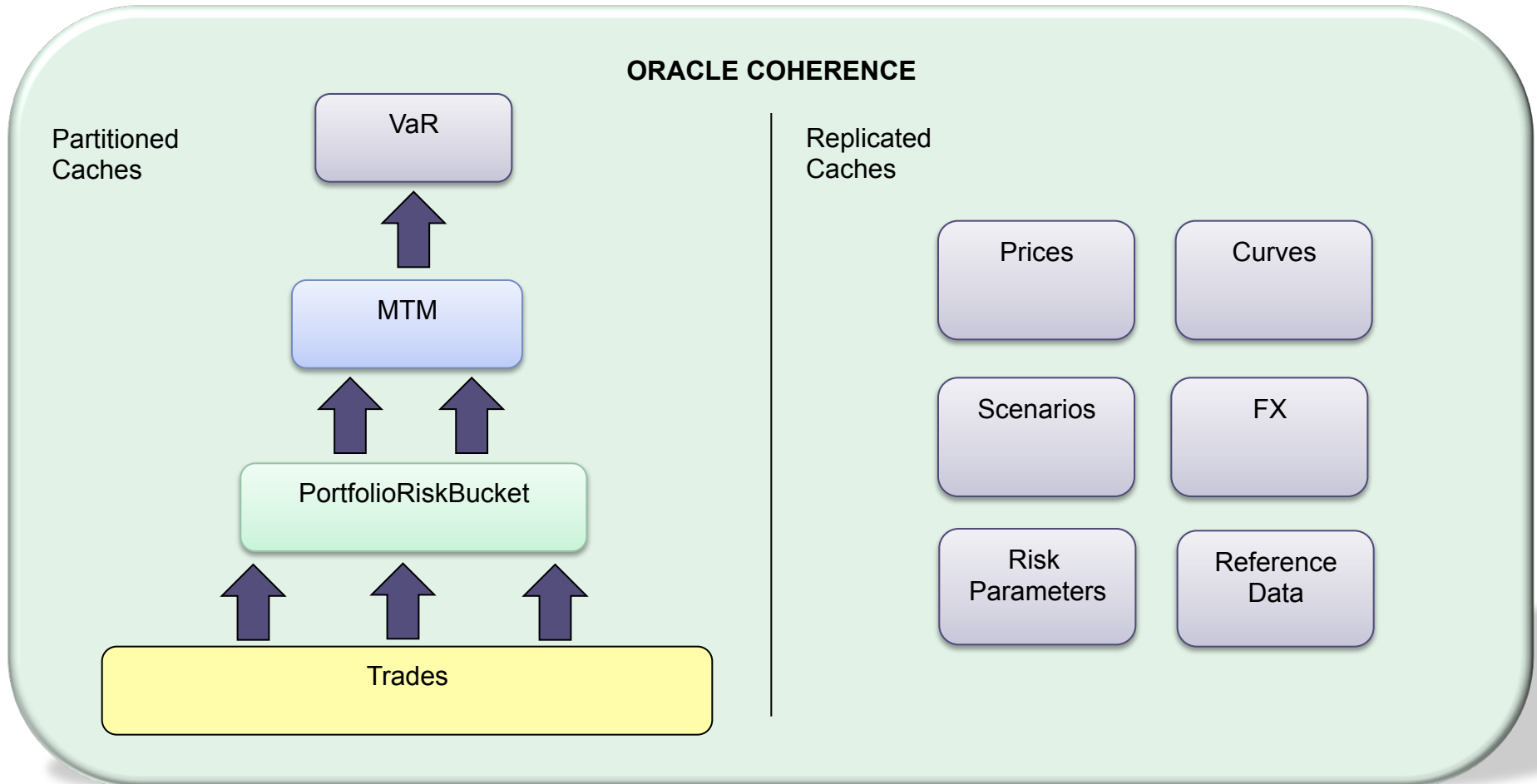
Realtime Mode – Hierarchical Cache

- Aggregate data “On The Fly”
 - Listeners on the Trade Cache  will collate and place data in the PortfolioRiskBucket  cache in real time.
 - No callers required. (Event driven)
- Calculate VaR in realtime
 - Listeners on the PortfolioRiskBucket cache  will perform the MTM calculations and update the MTM cache on PortfolioRiskBucket map events. (Insert,Delete,Update)
 - And similarly, The VaR cache will be updated on MTM cache events.



-  • Portfolio cache backing map listener
-  • Trade Cache backing map listener

Realtime Mode – Hierarchical Cache



Spring Enabled Coherence Nodes

- The Challenge

- How to load a Spring Context inside each Coherence Node ?
- How do we consume JMS message from a coherence agnostic JMS Queue ?
- How do we load the dependency injection artefacts for the Quantitative Library (DataProviders) ?

- Solution

- Use the Coherence-Common **Incubator** project
- <http://coherence.oracle.com/display/INCUBATOR/Coherence+Common>
- Extend the `coherence-cache-config.xml` namespace to include your very own cache config xml tags using a `AbstractNamespaceContentHandler`
- Use `com.oracle.coherence.environment.extensible.ExtensibleEnvironment`

Spring Enabled Coherence Nodes

```
package my.own.namespace.ext;

import java.net.URI;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.oracle.coherence.environment.extensible.ConfigurationContext;
import com.oracle.coherence.environment.extensible.ElementContentHandler;
import com.oracle.coherence.environment.extensible.namespaces.AbstractNamespaceContentHandler;
import com.tangosol.run.xml.XmlElement;

public class MyNamespaceContentHandler extends AbstractNamespaceContentHandler
{

    public void onStartScope(ConfigurationContext context,
                             String prefix,
                             URI uri)
    {
        registerContentHandler("MyXmlTag", new ElementContentHandler() {
            @Override
            public Object onElement(ConfigurationContext context, XmlElement xmlElement) {

                // LOAD SPRING CONTEXT HERE
                new ClassPathXmlApplicationContext((String) xmlElement.getValue(),String.class);

                return xmlElement;
            }
        });
    }
}
```

Spring Enabled Coherence Nodes

```
<?xml version="1.0"?>
<cache-config xmlns:MYTAG="class://my.own.namespace.etx.MyNamespaceContentHandler">

  <caching-scheme-mapping>
    <cache-mapping>
      <cache-name>Test</cache-name>
      <scheme-name>distributed</scheme-name>
    </cache-mapping>
  </caching-scheme-mapping>

  <caching-schemes>
    <distributed-scheme>
      <scheme-name>distributed</scheme-name>
      <service-name>DistributedCache</service-name>
      <backing-map-scheme>
        <local-scheme/>
      </backing-map-scheme>
      <autostart>true</autostart>
    </distributed-scheme>
  </caching-schemes>

  <MYTAG:MyXmlTag>spring-context.xml</MYTAG:MyXmlTag>
</cache-config>
```

- The Spring context will now load as part of the coherence node startup

Disaster Recovery – WAN Replication

- The Challenge

- How do we most efficiently replicate the entire cache to a secondary data centre to facilitate business continuity ?
- Do we have a cross WAN cluster ?
- Do we simply bootstrap the data in the secondary site and compromise HA ?

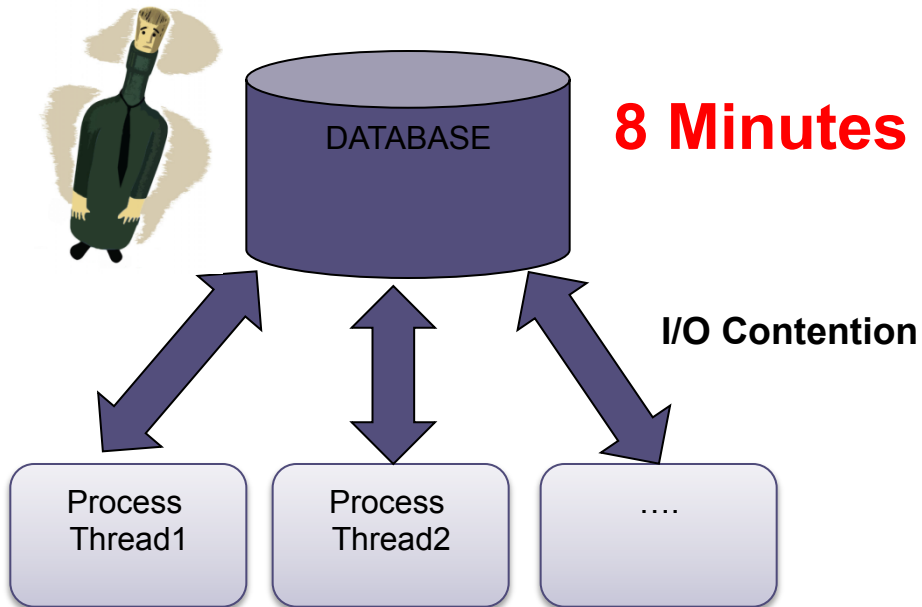
- Solution

- Use the Push Replication **Incubator** Pattern
- <http://coherence.oracle.com/display/INCUBATOR/Push+Replication+Pattern>
- Transparent
- Asynchronous (and thereby non intrusive)
- Active / Active vs Active / Passive (Hot Swap)
- Offload read only applications to other clusters. (Promote local access to Global Data)

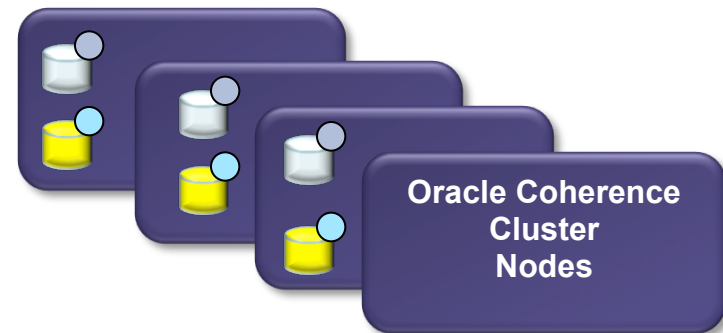
Performance

- 80,000 Trades, 100 Members, 2500 Scenarios per trade

Traditional Multi-Threaded



Compute Grid



46ms !!!!!

Summary

- Utilising Coherence's grid style code execution & event driven model combined with near instantaneous access to large volumes data gives us the ability to react to events, such as market data updates or portfolio position updates in real-time
- As we can pre-calculate many parts of VaR, such as the MTMs, the actual calculation on a request is extremely fast
- The main challenges we faced were adapting our quantitative library to be used via processors in Coherence & getting transactional data from system of record databases into the caches, both were solved by the ability to create Spring Contexts with Coherence nodes