INFINITE
REDPOTENTIAL
INSPIRE.CONNECT.PERFORM.

Experiences from testing a large Coherence application on Exalogic

Eugen Gendelman

Coherence consultant

Agenda

- Customer Use Case – large coherence application
- Why considering Oracle's Engineered Systems?
- Planning for Exalogic POC
- First impression of running on the Exastack
- Improving Performance of a large Coherence cluster
- Coherence filtering techniques
- What did we achieve?
- Q&A

Customer Use Case - platform

- Global investment Bank's platform provides critical data processing platform for regulatory reporting
- Failure to comply can result in unlimited fines, prison sentences or loss of banking licence
- Coherence based solution consumes trade data from several trading systems
- Consolidated trade data is transformed, enriched and reports generated in near real time
- Reports are sent to the appropriate regulatory body, whether this is to the US for Dodd Frank, Hong Kong for HKMA or other various regulatory bodies around the world
- Central view of trades across the Bank for all asset classes
- System failure leads to stop of trading = millions in lost profits

Customer Use Case - continued

- ~ 200 storage enabled Coherence nodes
- ~ 2 terabytes trade and reference data stored in Coherence
- 48 servers in 8 racks between Live and DR sites
- Trade data must be synchronously persisted for DR recovery – write-through for every mutation
- Heavy trade data and reference data querying
- Heavy use of Drools and XSLTs for transformation and enrichment
- Distributed and scalable state machine implemented on top of Coherence

Why considering Oracle's Engineered Systems?

- Legislative and regulatory rules put significant pressure on the availability and performance of the system
- No noticeable down time
- Fast time to recovery
- Zero data loss
- Lack of consistency with internal build/networking/patching
- Need to process 10x trades per second
- Expansion plans beyond trade reporting

Exastack POC – planning

- Test scenarios include:
  - Kill several Coherence nodes
  - Panic physical servers
  - Kill and recover the whole Coherence cluster
  - Kill Exalogic Network switch

- Over 100 destructive tests runs in 3 weeks
- All this while processing **1 Million trades!**
- **In addition to achieving 1000% performance improvement on half of the servers**

Exastack POC - preparation

- Create production like test data - 1 million trades
- Simulators for Trading Systems and Regulators
- Fast multi-threaded export/import mechanism
- Attempts to improve performance on the commodity kit

- *Focus on repeatability*

POC setup

Live Exalogic
half rack
12 blades of 16

4 logical racks

JDBC

Live Exadata
RAC
4 blades

Data Guard

DR Exalogic
half rack
12 blades of 16

4 logical racks

JDBC

DR Exadata
RAC
4 blades

First 1M trades run on Exalogic

- Using LightMessageBus - the network is no longer the bottleneck – RDMA (Remote Direct Memory Access)
- Starting up 140 nodes simultaneously is fast and it works
- Exalogic was too fast for our code!
- Exposes thread safety issues as though under an x-ray

```
//This line produced lots of errors but worked in prod for over a year
private static SimpleDateFormat dateFormat = new SimpleDateFormat("...");

//This line in authorized-hosts filter caused dead lock on node startup
private static final NamedCache hosts =
        new ContinuousQueryCache(namedCache, AlwaysFilter.INSTANCE, false);
```

First 1M trades run on Exalogic
continued

- Drools (cached in Coherence) thread safety issues resulted in infinite loop on initialization when hit by multiple threads – had to implement a single threaded “warm-up” mechanism

```
Thread[TradeViewCacheWorker:5,5,TradeViewCache]
        java.util.WeakHashMap.get(WeakHashMap.java:470)
        org.mvel2.util.ParseTools.getBestCandidate(ParseTools.java:246)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.getMethod(ReflectiveAccessorOptimizer.java:1037)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.getMethod(ReflectiveAccessorOptimizer.java:982)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.compileGetChain(ReflectiveAccessorOptimizer.java:375)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.optimizeAccessor(ReflectiveAccessorOptimizer.java:141)
        org.mvel2.ast.ASTNode.optimize(ASTNode.java:157)
        org.mvel2.ast.ASTNode.getReducedValueAccelerated(ASTNode.java:113)
        org.mvel2.ast.BinaryOperation.getReducedValueAccelerated(BinaryOperation.java:116)
        org.mvel2.MVELRuntime.execute(MVELRuntime.java:87)
        org.mvel2.compiler.CompiledExpression.getValue(CompiledExpression.java:122)
        org.mvel2.compiler.CompiledExpression.getValue(CompiledExpression.java:115)
        ....

Thread[TradeViewCacheWorker:40,5,TradeViewCache]
        java.util.WeakHashMap.get(WeakHashMap.java:470)
        org.mvel2.util.ParseTools.getBestCandidate(ParseTools.java:246)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.getMethod(ReflectiveAccessorOptimizer.java:1037)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.getMethod(ReflectiveAccessorOptimizer.java:982)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.compileGetChain(ReflectiveAccessorOptimizer.java:375)
        org.mvel2.optimizers.impl.refl.ReflectiveAccessorOptimizer.optimizeAccessor(ReflectiveAccessorOptimizer.java:141)
        org.mvel2.ast.ASTNode.optimize(ASTNode.java:157)
        org.mvel2.ast.ASTNode.getReducedValueAccelerated(ASTNode.java:113)
        org.mvel2.ast.BinaryOperation.getReducedValueAccelerated(BinaryOperation.java:116)
        org.mvel2.MVELRuntime.execute(MVELRuntime.java:87)
        org.mvel2.compiler.CompiledExpression.getValue(CompiledExpression.java:122)
        org.mvel2.MVEL.executeExpression(MVEL.java:930)
        org.drools.base.mvel.MVELPredicateExpression.evaluate(MVELPredicateExpression.java:100)
        org.drools.rule.PredicateConstraint.isAllowed(PredicateConstraint.java:291)
        org.drools.reteoo.AlphaNode.assertObject(AlphaNode.java:130)
        org.drools.reteoo.SingleObjectSinkAdapter.propagateAssertObject(SingleObjectSinkAdapter.java:59)
        org.drools.reteoo.AlphaNode.assertObject(AlphaNode.java:134)
        org.drools.reteoo.CompositeObjectSinkAdapter.doPropagateAssertObject(CompositeObjectSinkAdapter.java:458)
        org.drools.reteoo.CompositeObjectSinkAdapter.propagateAssertObject(CompositeObjectSinkAdapter.java:386)
        .....
```

Improving write-through performance

- POF objects stored as BLOBs in the database
- Average 30 database updates per trade
- Target 1,000 trades per second = 30,000 blob writes per second – a small volume for Exadata
- But need to reduce redo logs contention
- **Coherence cache store bundling doubled database throughput**

```
<operation-bundling>
    <bundle-config>
        <operation-name>store</operation-name>
        <preferred-size>4</preferred-size>
        <delay-millis>10</delay-millis>
        <thread-threshold>4</thread-threshold>
        <auto-adjust>false</auto-adjust>
    </bundle-config>
</operation-bundling>
```

Nested AND filters

- What is the cost of executing nested AND filters, assuming all extractors are indexed?

```
Filter filter = new AndFilter(
        new AndFilter(
                new AndFilter(new EqualsFilter(TRADE_ID_EXTRACTOR, tradeId),
                              new EqualsFilter(TRADE_VERSION_EXTRACTOR, version)),
                new AndFilter(new EqualsFilter(ASSET_CLASS_EXTRACTOR, assetClass),
                              new EqualsFilter(SOURCE_SYSTEM_EXTRACTOR, sourceSytem))),
        new AndFilter(
                new AndFilter(new EqualsFilter(REPORT_TYPE_EXTRACTOR, reportType),
                              new EqualsFilter(REGION_EXTRACTOR, region)),
                new AndFilter(new EqualsFilter(STATUS_EXTRACTOR, notInvalidStatus),
                              new EqualsFilter(IS_ERROR_EXTRACTOR, false))));
```

- 7 key sets intersections – N operation
- Low Cardinality filters may intersect very large sets
- Performance degradation as cache size increases
- Can easily take hundreds of milliseconds on each node
- Unless KeyAssociatedFilter used, ALL nodes will do similar work event where trade is not present
- High CPU impact

All Filter

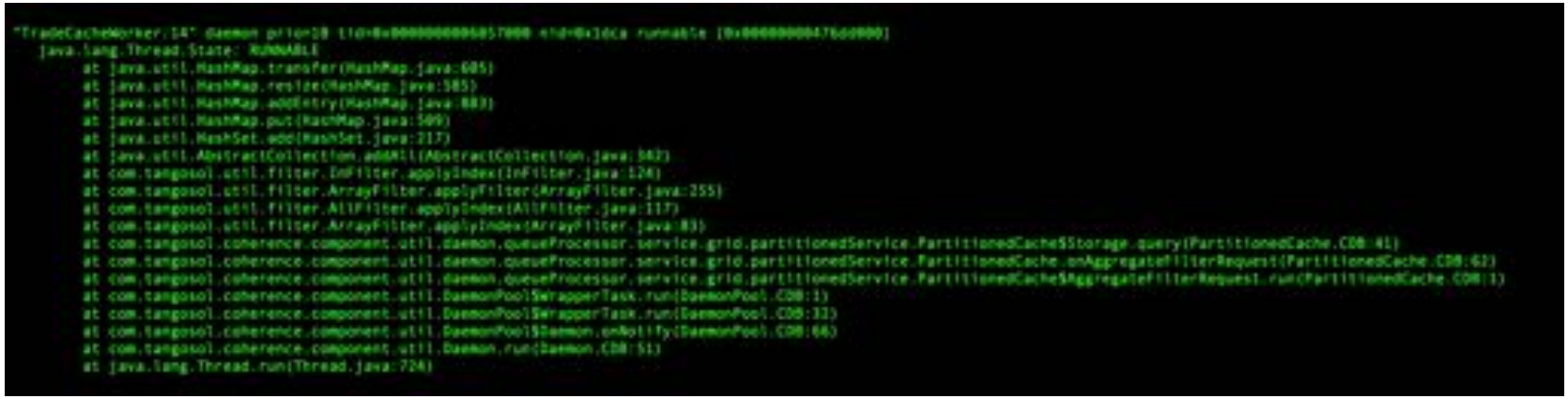
This filter runs ~300 times faster on our cluster

```
Filter allFilter = new AllFilter(new Filter[] {
            new EqualsFilter(TRADE_ID_EXTRACTOR, tradeId),
            new EqualsFilter(TRADE_VERSION_EXTRACTOR, version),
            new EqualsFilter(ASSET_CLASS_EXTRACTOR, assetClass),
            new EqualsFilter(SOURCE_SYSTEM_EXTRACTOR, sourceSytem),
            new EqualsFilter(REPORT_TYPE_EXTRACTOR, reportType),
            new EqualsFilter(REGION_EXTRACTOR, region),
            new EqualsFilter(STATUS_EXTRACTOR, notInvalidStatus),
            new EqualsFilter(IS_ERROR_EXTRACTOR, false)
});
```

- Individual filter's results evaluated sequentially
- Always takes ~1ms assuming high cardinality (uncommon or unique) of tradeId filter
- Nodes where the tradeId is not present would do almost no work
- Not only is this faster but it also frees up CPU resources
- Order is important

InFilter behaviour

```
Filter allFilter = new AllFilter(new Filter[] {
        new EqualsFilter(TRADE_ID_EXTRACTOR, tradeId),
        new EqualsFilter(TRADE_VERSION_EXTRACTOR, version),
        new EqualsFilter(ASSET_CLASS_EXTRACTOR, assetClass),
        new EqualsFilter(SOURCE_SYSTEM_EXTRACTOR, sourceSystem),
        new EqualsFilter(REPORT_TYPE_EXTRACTOR, reportType),
        new EqualsFilter(REGION_EXTRACTOR, region),
        new InFilter(STATUS_EXTRACTOR, completeStatuses)
});
```



- InFilter will ‘union’ all keys matching selection
- Solution is to override ***applyIndex*** method making it to validate the keys passed into the filter

Coherence AllFilter optimisation

- AllFilter optimisation is based on the result of ***calculateEffectiveness*** method

1. Match filters executed first – effectiveness 1

   EqualsFilter, ContainsFilter, IsNullFilter, NotEqualsFilter, IsNotNullFilter

2. Range filters

   GreaterFilter, GreaterEqualsFilter, LessEqualsFilter, LessFilter

3. Iterator filters

   *InFilter, ContainsAllFilter, ContainsAnyFilter, LikeFilter**

4. Unindexed last

InFilter effectiveness

How to make Coherence execute InFilter first?

```
Filter allFilter = new AllFilter(new Filter[] {
        new InFilter(TRADE_ID_EXTRACTOR, tradeIds),
        new EqualsFilter(ASSET_CLASS_EXTRACTOR, assetClass),
        new EqualsFilter(SOURCE_SYSTEM_EXTRACTOR, sourceSystem),
        new EqualsFilter(REPORT_TYPE_EXTRACTOR, reportType),
        new EqualsFilter(REGION_EXTRACTOR, region),
        new EqualsFilter(STATUS_EXTRACTOR, notInvalidStatus),
        new EqualsFilter(IS_ERROR_EXTRACTOR, false)
});
```

```
public class EffectiveInFilter extends InFilter {

    @Override
    public int calculateEffectiveness(Map mapIndexes, Set setKeys) {
        MapIndex index = (MapIndex) mapIndexes.get(getValueExtractor());
        if (index == null) {
            return calculateIteratorEffectiveness(setKeys.size());
        }
        return 1;
    }

}
```

Getting it right on large scale project

- Challenge
  - About 800 places in the code where filters constructed
  - Global team - not all devs fully understand production data set
  - Similar filters use different extractors – indexing nightmare
  - Do I need to use KeyAssociatedFilter?
  - One un-optimised filter can effect stability of the whole cluster
  - Very difficult to troubleshoot

- Solution is to abstract the complexity into a Filter Builder

```
Filter filter = FilterBuilder.newInstance().equalsIsError(false)
        .equalsAssetClass(assetClass).includeTradeIds(tradeIds)
        .equalsTradeVersion(version).equalsSourceSystem(sourceSystem)
        .equalsRegion(region).equalsReportType(reportType).build();
```

How does Filter Builder work?

- Fluent API easy to use
- AllFilter constructed automatically based on cardinality
- KeyAssociatedFilter will be used automatically where possible
- Implements logging for slow filters – above threshold
- Can track the which component constructed the “offending” filter
- QueryRecorder can be used on a specific node using Invokable

```
private TreeMap<Integer, List<Filter>> filterMap = null;

private enum Cardinality {
    tradeIdEquals, includeTradeIds, equalsTradeVersion, equalsAssetClass, equalsSourceSystem,
    equalsReportType, equalsRegion,
    equalsStatus, equalsIsError
}

public FilterBuilder includeTradeIds(Set<String> tradeIds) {
    return add(Cardinality.includeTradeIds.ordinal(), new InFilter(TRADE_ID_EXTRACTOR, tradeIds));
}
```

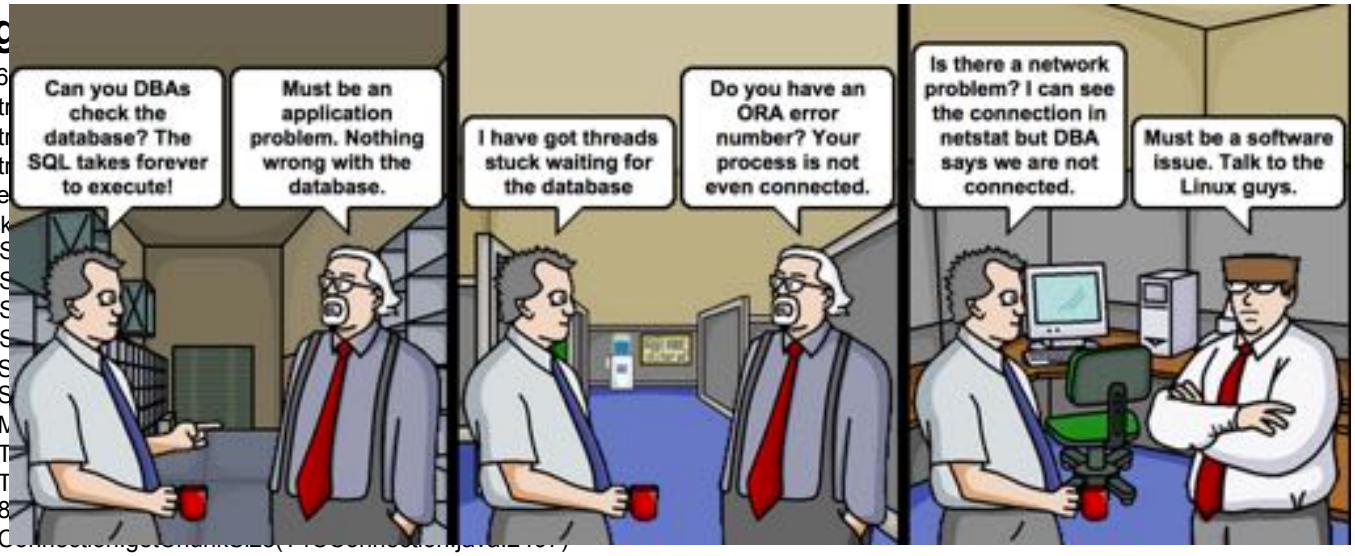
What did we achieve?

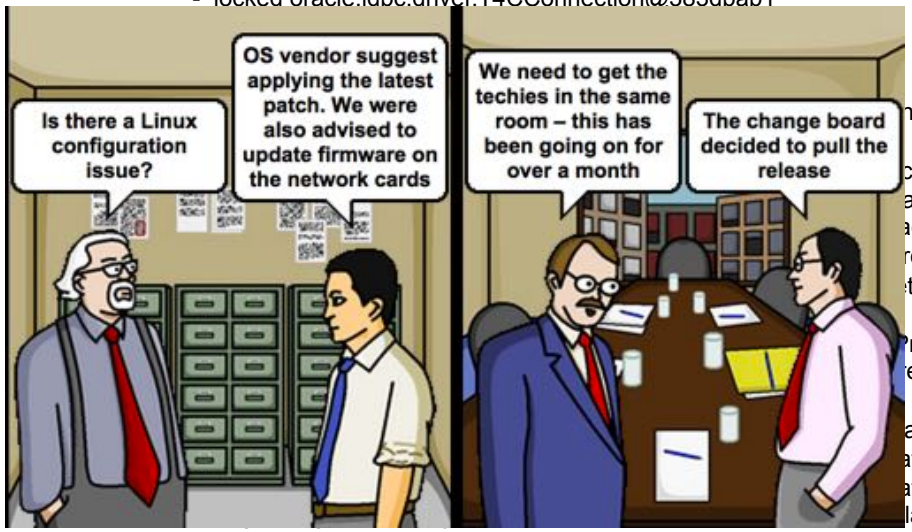
- No noticeable impact on overall processing time and zero data loss when killing
  - Coherence Nodes,
  - Rack
  - Exalogic switch
- 18 times performance improvement (target was 10)
  - Half of the improvement were due to the code optimization
  - Code deficiencies would be hard to identify on a slower network and slower hardware
- Completely automated testing process

Internal build stack reality

Troubleshooting

"TradeCacheWorker:33" id=96
at java.net.SocketInputStr
at java.net.SocketInputStr
at java.net.SocketInputStr
at oracle.net.ns.Packet.re
at oracle.net.ns.DataPack
at oracle.net.ns.NetInputS
at oracle.net.ns.NetInputS
at oracle.net.ns.NetInputS
at oracle.net.ns.NetInputS
at oracle.jdbc.driver.T4CS
at oracle.jdbc.driver.T4CS
at oracle.jdbc.driver.T4CM
at oracle.jdbc.driver.T4CT
at oracle.jdbc.driver.T4CT
at oracle.jdbc.driver.T4C8
at oracle.jdbc.driver.T4CC
- locked oracle.jdbc.driver.T4CConnection@383dbab1





ntentsForBlobCritical(OraclePreparedStatement.java:7061)
clePreparedStatement.java:11492)
aredStatement.java:11261)
aclePreparedStatementWrapper.java:560)
rce)
thodAccessorImpl.java:43)
roxyFactory.java:230)
eparedStatementProxyFactory.java:124)
acheStore.java:201)
atement(JdbcTemplate.java:893)
atement(JdbcTemplate.java:1)
late.java:587)