
ORACLE®

Agenda

1. Intro
2. Overview of architecture
3. JMeter
4. OEP
5. ELK
6. Demo
7. Q&A

Intro

Jon Hall
Oracle Consulting
littlegrid developer and maintainer

Ivan Cikic
Oracle Consulting

Mini-me architecture

- Load test tool – JMeter
- Queues – ActiveMQ
- Event processing – OEP
- Caching – Coherence, of course :-)
- Data/log analysis – ELK (ElasticSearch, LogStash and Kibana)

Mini-me architecture (continued)

- JMeter – reads source data from a CSV and publishes it to a topic
- ActiveMQ – provides the demo messaging infrastructure
- OEP – subscribes to the topic, parses the payload and performs event processing
- Coherence – acts as in-memory storage/data grid (could be queried)
- ELK – visualisation of requests

JMeter - concepts/terminology

- Test plan – your test creation area
- Workbench – temporary 'paste' area, also used to configure recording proxy
- Sampler (request type): Web - HTTP(S), SOAP, FTP, JDBC, JMS, TCP, others
- Logic controllers (if, loop etc.)
- Config Element (e.g. HTTP Request Details)
- Timers
- Pre/pro processing
- Assertions (response checks)
- Listener (reporting, summaries, graphs, write to file etc.)

JMeter - example

- Building an example:
  - Reading source data from CSV
  - JMS publish and subscribe
  - Looking at output

- Toggle is your friend :-)

JMeter – avoiding hard-coding

- User defined variables:
  - Name: numberOfMessages
  - Value: ${__P(numberOfMessages, 20)}
- Loop count: ${numberOfMessages}
- jmeter -JnumberOfMessages=25
- Description field as temporary paste area

JMeter summary

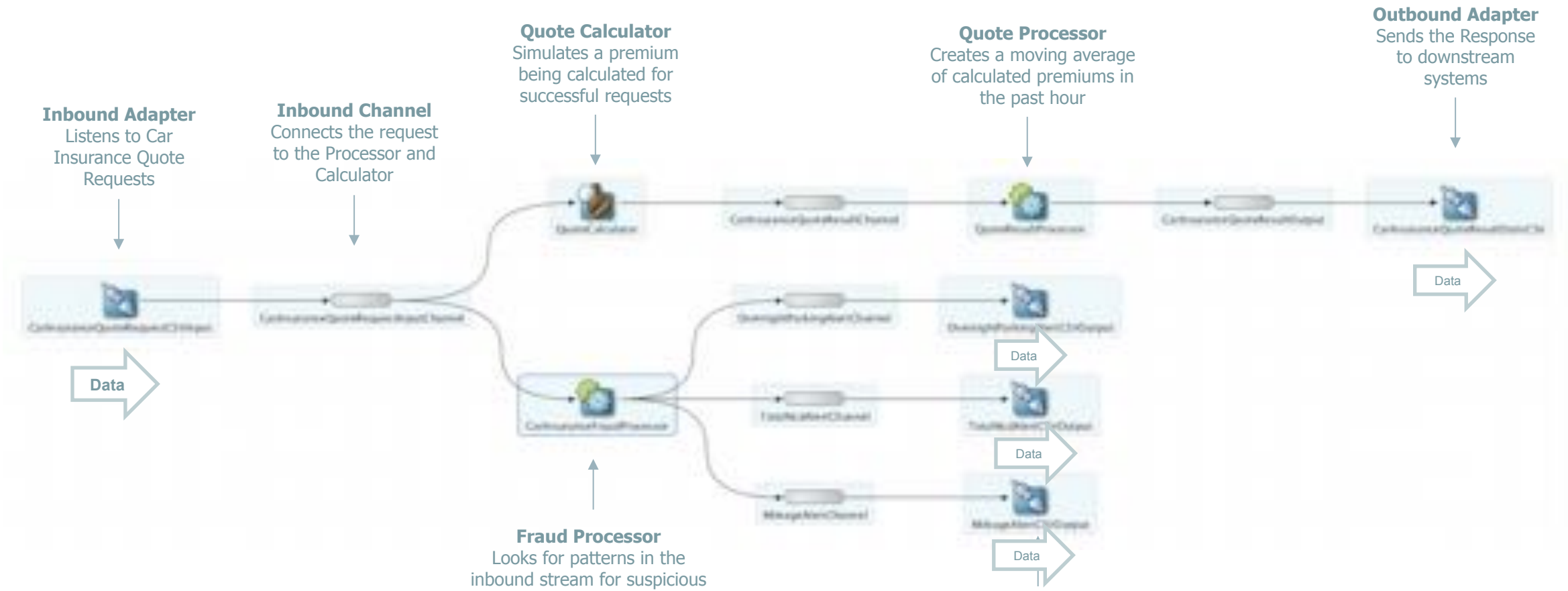
- Distributed mode
- Lots of other features not covered here
- BeanShell is useful

ORACLE
FUSION MIDDLEWARE

Oracle Event Processing

- Light-weight Java Application Server (embeddable)
- Based on three simple concepts:
  - Event Adapters – Inbound and outbound external connections
  - Event Channels – To connect things together
  - Event Processors – To process information in real-time
- These components connect to form an Event Processing Network (EPN)
- The heart of processing is Continuous Query Language (CQL)
- Good integration with Oracle Coherence

Event Processing Network (EPN)



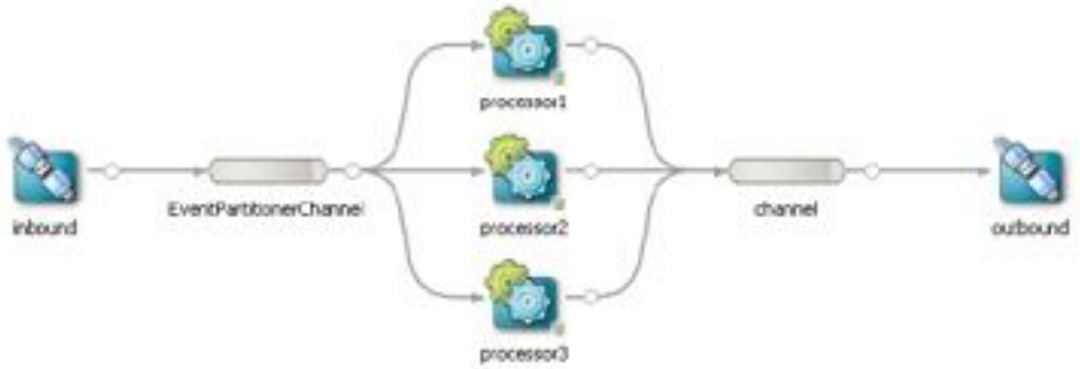
ORACLE
FUSION MIDDLEWARE

Event Adapters

- Adapters manage data entering and leaving the EPN
- A number of different inbound / outbound adapters is provided OOTB:
  - JMS
  - REST
  - EDN
  - CSV
  - HTTP Publish-Subscribe
  - High Availability Adapters
  - Write your own Adapter

Event Channels

- A channel represents the logical conduit through which events flow between other types of components
- Channels provide buffering, queuing and concurrency
- Event partitioning
- Channel selector



Event Processors

- Processes incoming events from various input channels and other data sources
- Processors use Oracle Continuous Query Language (CQL) to write the business logic in the form of continuous queries

```
<query id="findPriceBySymbol"><![CDATA[
    select price from StockStreamChannel where symbol = "ORCL"
]]></query>
```

Continuous Query Language (CQL)

- CQL supports:
  - Filtering, Aggregation, Projection
  - Time and Count based windows
  - Slides
  - Joining streams
  - Pattern matching (with MATCH_RECOGNIZE)
    - Top / Bottom N
    - Up / Down Trend
    - Fluctuation
    - Eliminate / Detect Duplicates
    - Detect Missing Event
    - W / Inverse W

```
<query id="avgPricePerSymbol"><![CDATA[
    select avg(price), symbol
    from StockStreamChannel [PARTITION BY symbol ROWS 20]
    group by symbol
]]></query>
```

```
<query id="joinFeeds"><![CDATA[
   select s1.price as price1, s2.price as price2, name
   from ReutersMarketFeed[RANGE 60 SECONDS] as s1,
        BloombergFeed[RANGE 60 SECONDS] as s2,
        TickerListing as r1
   where s1.symbol = s2.symbol and s2.symbol = r1.tickerSymbol
]]></query>
```

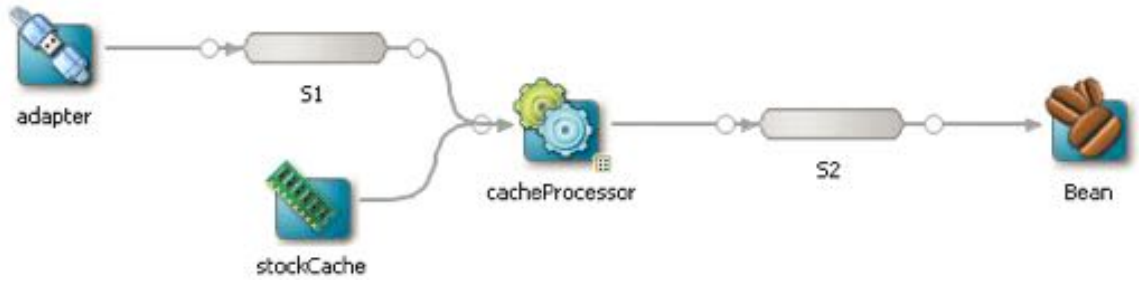
ORACLE
FUSION MIDDLEWARE

Continuous Query Language (CQL)

- CQL supports joining with external sources
  - Coherence cache
  - Database table
- Extendable via data cartridges
  - JDBC
  - Hadoop
  - Oracle NoSQL
  - Java

Oracle Coherence and OEP

- Access to non-streaming data, event enrichment for example



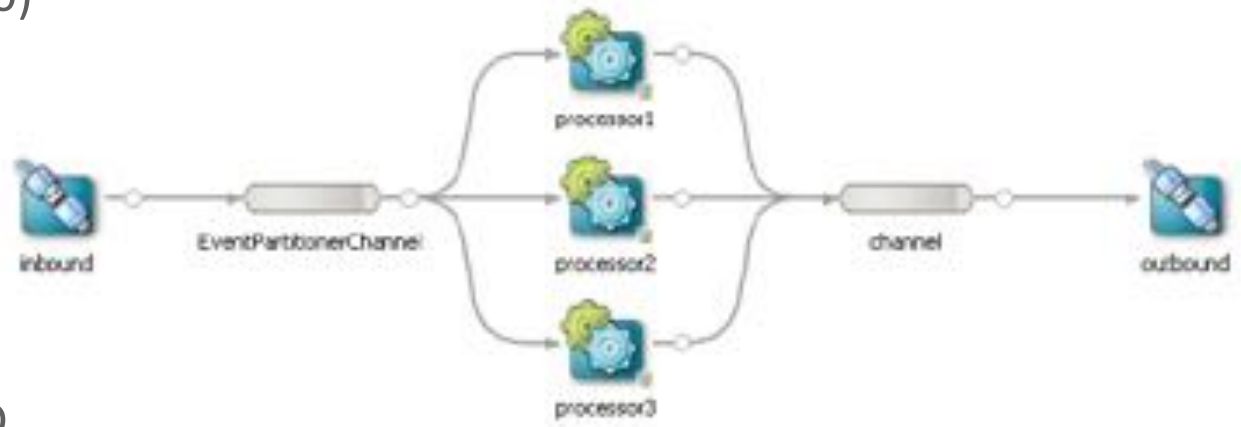
- Cache as event sink

- Cache as event source (pull or push model)
- Perform data grid operations within Event Beans

Performance Tuning and Scalability

Scale Up

- Channel threading and buffering
  - Pass Through (max-threads=0, max-size=0). Event ordering preserved.
  - Synchronous Handoff (max-threads>0, max-size=0)
  - Concurrent Queue (max-threads>0, max-size>0)
- Event Partitioning Channel
- Batching channel
- Parallel CQL execution
  - ORDERED, UNORDERED, PARTITION_ORDERED
- Parallel execution of adapters and event beans (work manager threads)

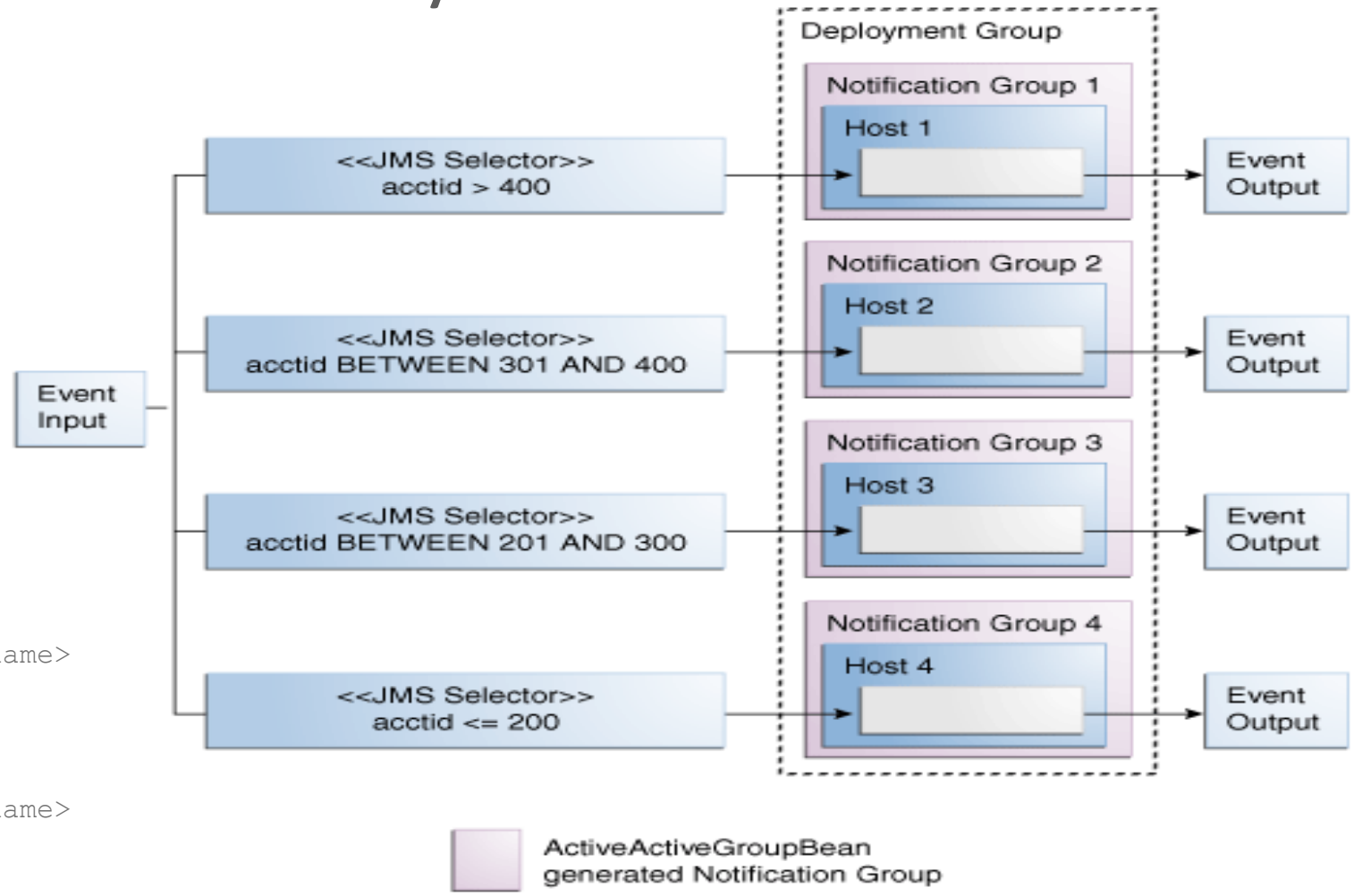


Performance Tuning and Scalability

Scale Out

- Scalability with Notification Groups
- Partition incoming JMS stream to multiple servers
- Clustering supported with Oracle Coherence

```
<cluster>
        <server-name>oep-server-1</server-name>
        <enabled>coherence</enabled>
        <groups>LondonGroup_1</groups>
</cluster>
<cluster>
        <server-name>oep-server-2</server-name>
        <enabled>coherence</enabled>
        <groups>LondonGroup_2</groups>
</cluster>
```

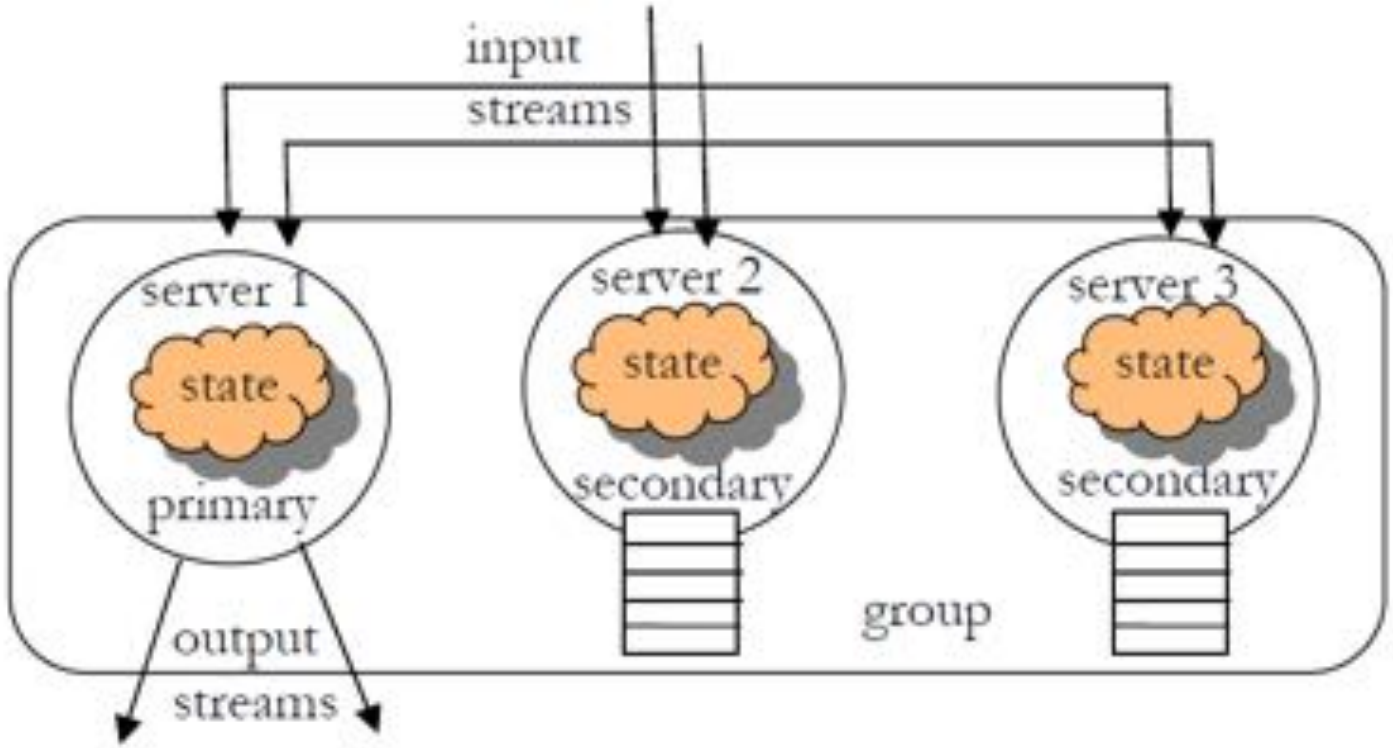


Types of High Availability (in context of CEP systems)

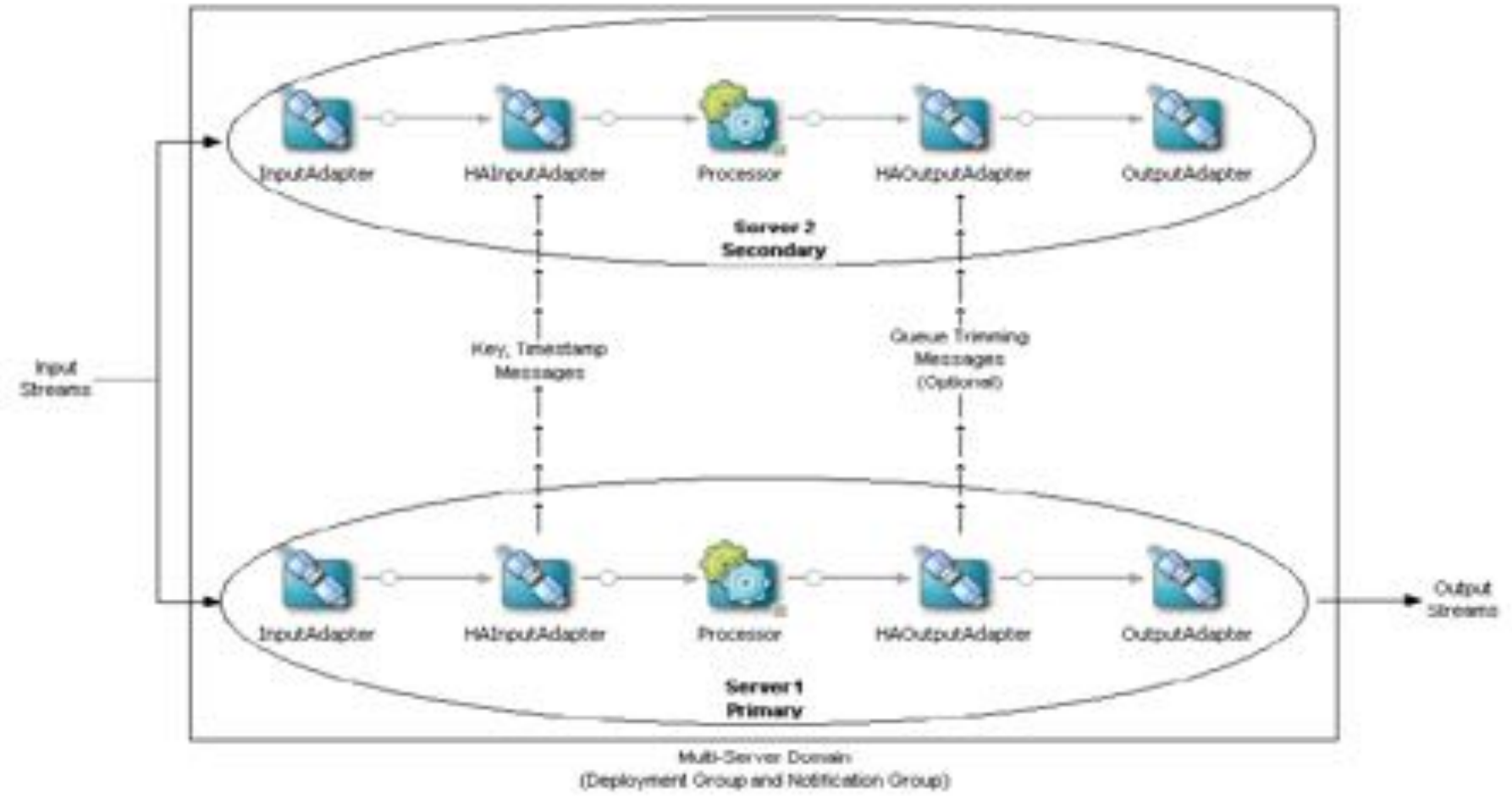
- CEP systems are characterised by very dynamic, constantly changing data
- CEP systems are often highly stateful
- Typical solutions to statefulness problem:
  - Active/active – replicate the behaviour of the system
  - Active/passive – replicate the state of the system
  - Upstream backup – saving the stream of events that produced the state so that it can be rebuilt

Oracle Event Processing HA

- Relies on Oracle Coherence
- Supports active/active architecture
- Primary instance responsible for sending events
- Secondary instances buffer output events
- High Availability Adapters acting as proxies



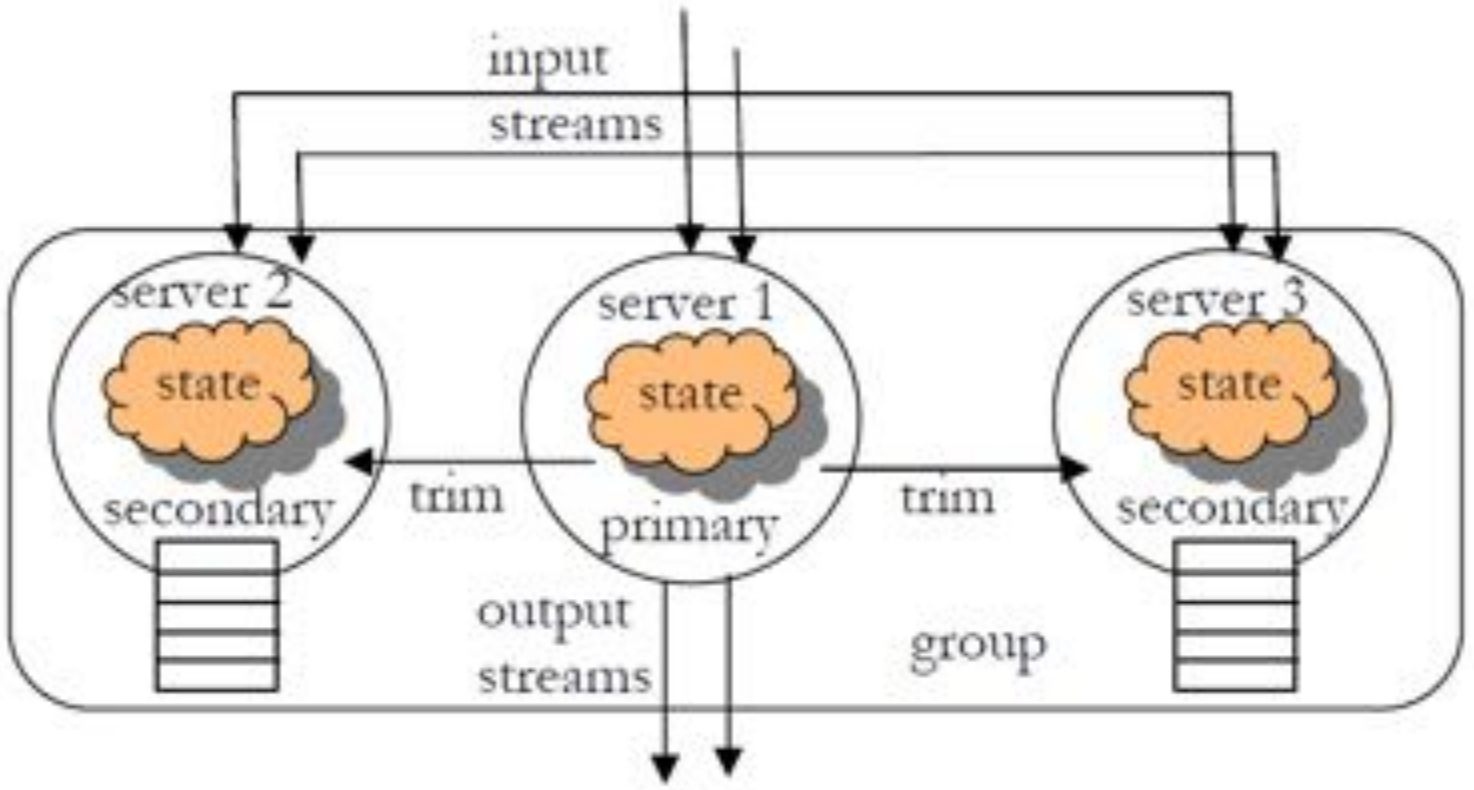
High Availability Adapters



High Availability Quality of Service

Quality of Service	Missed Events	Duplicate Events	Performance Overhead
Simple Failover	Yes (many)	Yes (few)	Negligible
Simple Failover with Buffering	Yes (less)	Yes (many)	Low
Light-Weight Queue Trimming	No	Yes (few)	Medium
Precise Recovery with JMS	No	No	High

Light-Weight Queue Trimming



ORACLE
FUSION MIDDLEWARE

High Availability Design Patterns

- Select the minimum high availability your application can tolerate
- Limit the application state
- Ensure applications are idempotent
- Make events universally identifiable
- Understand the importance of event ordering (queue trimming)
- Prefer deterministic behaviour
- Prefer monotonic event identifiers
- Plan for server recovery

ELK

- ElasticSearch – stores data in-memory for querying
- Logstash – agent, captures and uploads data to ElasticSearch
- Kibana – frontend GUI to ElasticSearch
- ELK? Or should it be LEK?
  - But that doesn't sound as cool :-p

Logstash

- Tool for receiving, processing and outputting logs.
- Written in JRuby.
- Inputs
  - Over 40 types, e.g. file, stdin, JMX, Log4J, etc.
- Filters
  - Over 50 types, e.g. Grok, etc.
- Outputs
  - Over 55 types, e.g. ElasticSearch, file, stdout, etc.

Logstash configuration

- Configuration string
  - logstash -e "input { stdin{} } output { stdout{} }"
- Configuration file: logstash -f simple.conf

```
input { stdin{} }
filter { grok { match => [ "message", "%{NUMBER:quote:float}" ] } }
output {
   stdout { codec => json }
   elasticsearch { cluster => testcluster  index => quotes  codedec => json}
}
```

Logstash configuration

```
input {
jmx {
    path => "myjmx"
    polling_frequency => 30
    type => "jmx"
    nb_thread => 4
  }
}
output {
  stdout {}
  elasticsearch{
     cluster => "testcluster"
  }
}
```

ElasticSearch overview

- Built on Lucene (high performance text search engine).
- Stores data in JSON
- REST API for putting, querying and deleting data
- Scale out with HA
- Document oriented, schema-free

ElasticSearch terminology

- Index is like a database, examples:
  - quotes
  - logstash-2015.03.13
  - Logstash-2015.03.17
- You can have as many as you want
- Can be created dynamically or upfront
- Type is a bit like a table
  - Such as 'logs'
  - http://localhost:9200/quotes/_all/_mapping?pretty=true

ElasticSearch types

```
{ "quotes" : {
    "mappings" : {
      "logs" : {
        "properties" : {
          "@timestamp" : { "type" : "date", "format" : "dateOptionalTime"  },
          "@version" : { "type" : "string" },
          "host" : { "type" : "string" },
          "message" : { "type" : "string" },
          "quote" : { "type" : "double" },
          "tags" : { "type" : "string" }
        }
}}}}
```

ElasticSearch documents

- Document is like a row in a table


```
{"_index": "quotes",
    "_type": "logs",
    "_id": "ASWcW8RoShaOjWHddcc8DQ",
    "_version": 1,  "_score": 1, "_source": {
        "message": "123",
        "@version": "1",
        "@timestamp": "2015-03-18T18:29:48.405Z",
        "host": "lenovo2",
        "quote": 123
}}
```

Kibana

- Browser-based GUI for use with ElasticSearch
- Used for analysis of data (Discover)
- Used for visualization of data (Visualize)
  - Buckets
- Building cool dashboards

Kibana

Questions?

Thank you!


ORACLE®