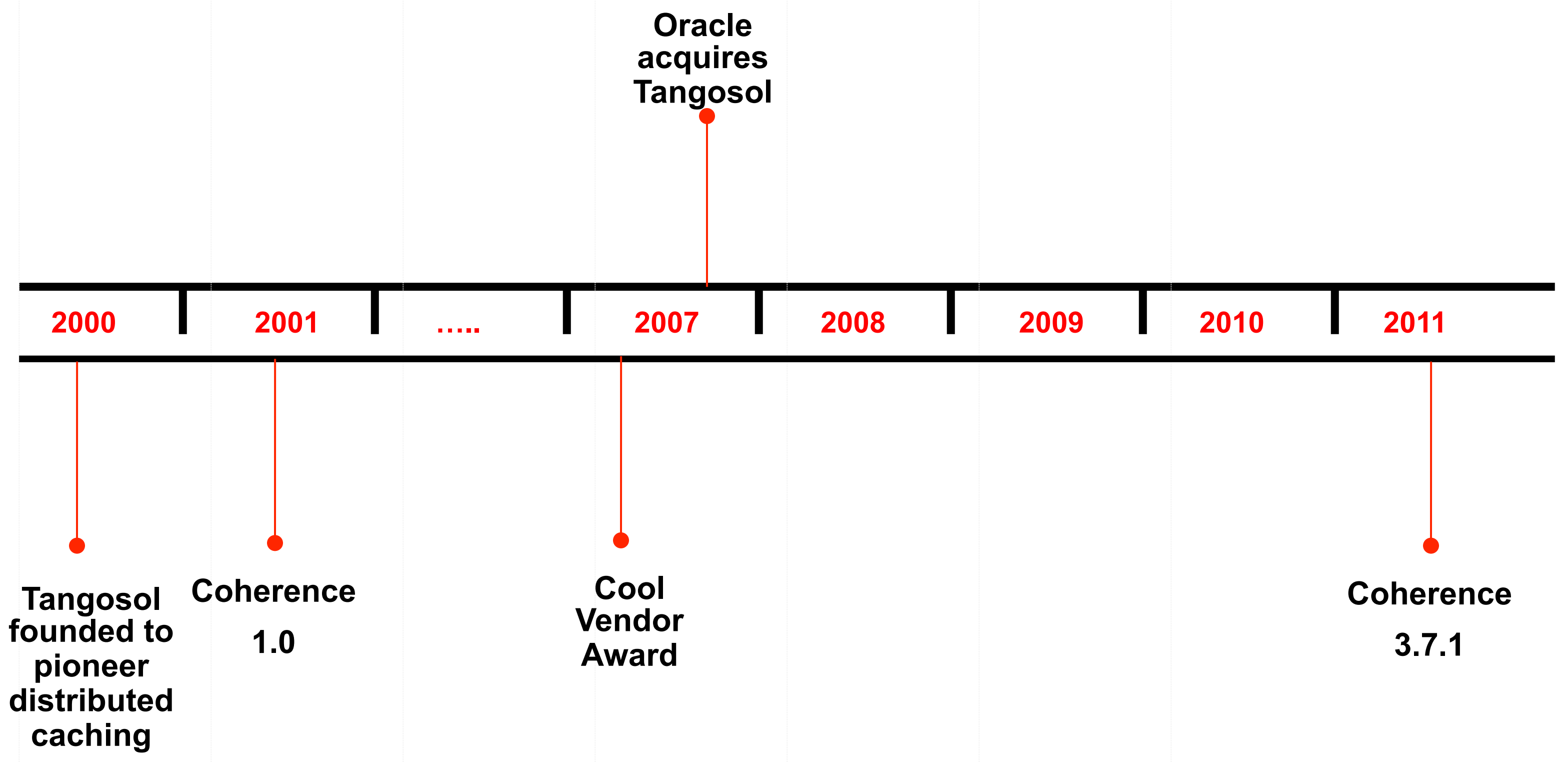
A Brief Introduction to Live Object Pattern

Dave Felcey
Coherence Product Management

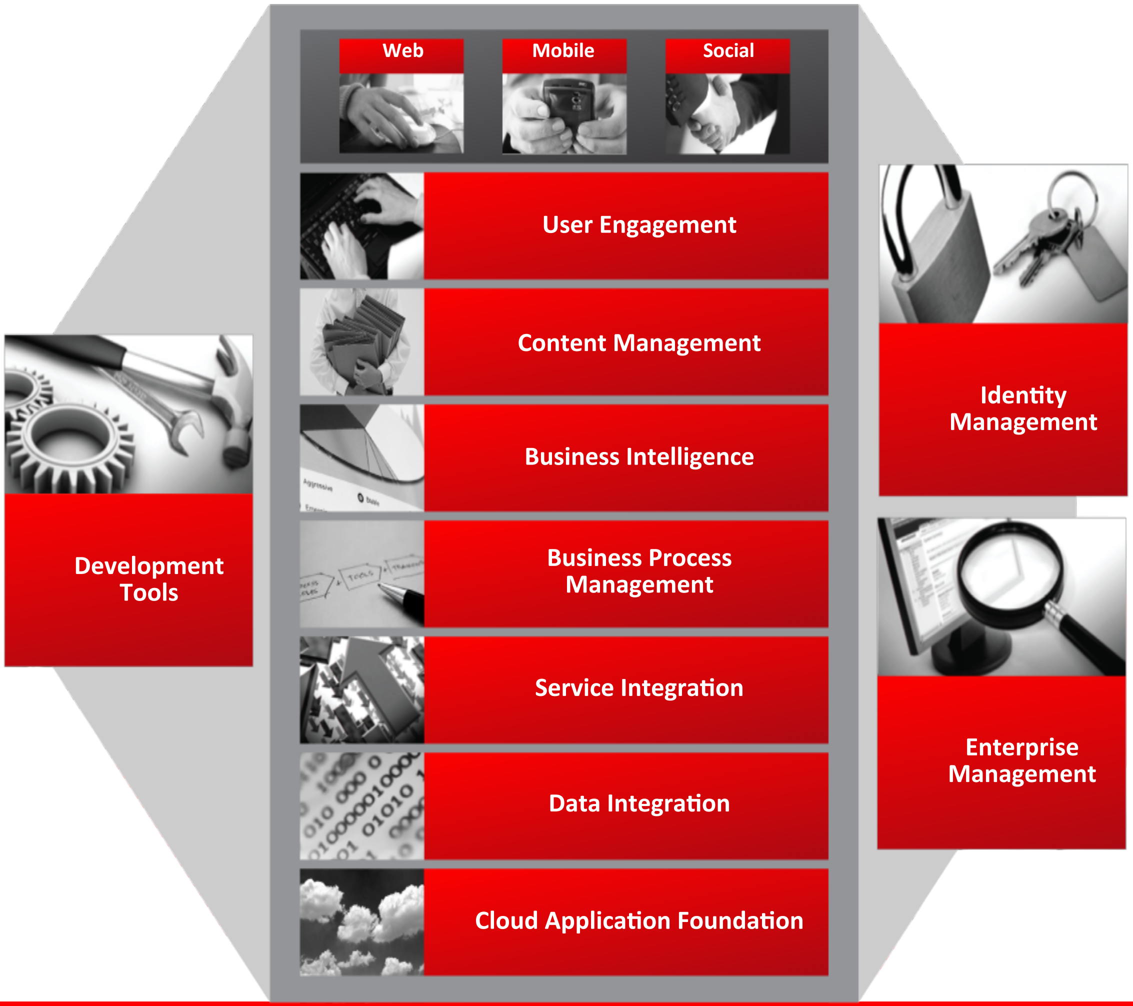
Oracle Coherence In-memory Data Grid

Secret Sauce for Internet Scale



Oracle Fusion Middleware

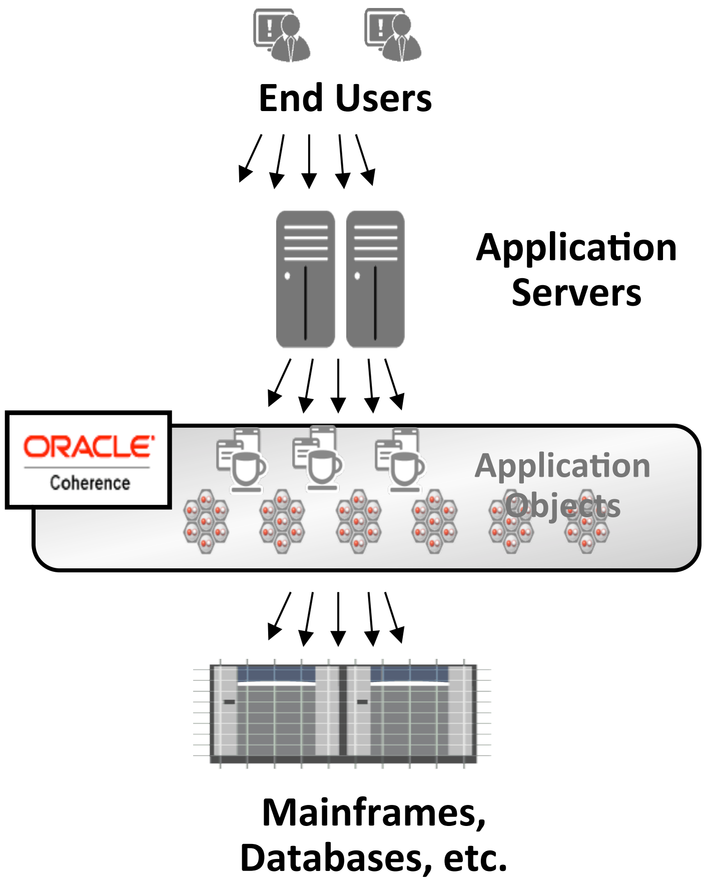
Complete, Open, Integrated, Best-in-Class



Oracle Coherence In-Memory Data Grid

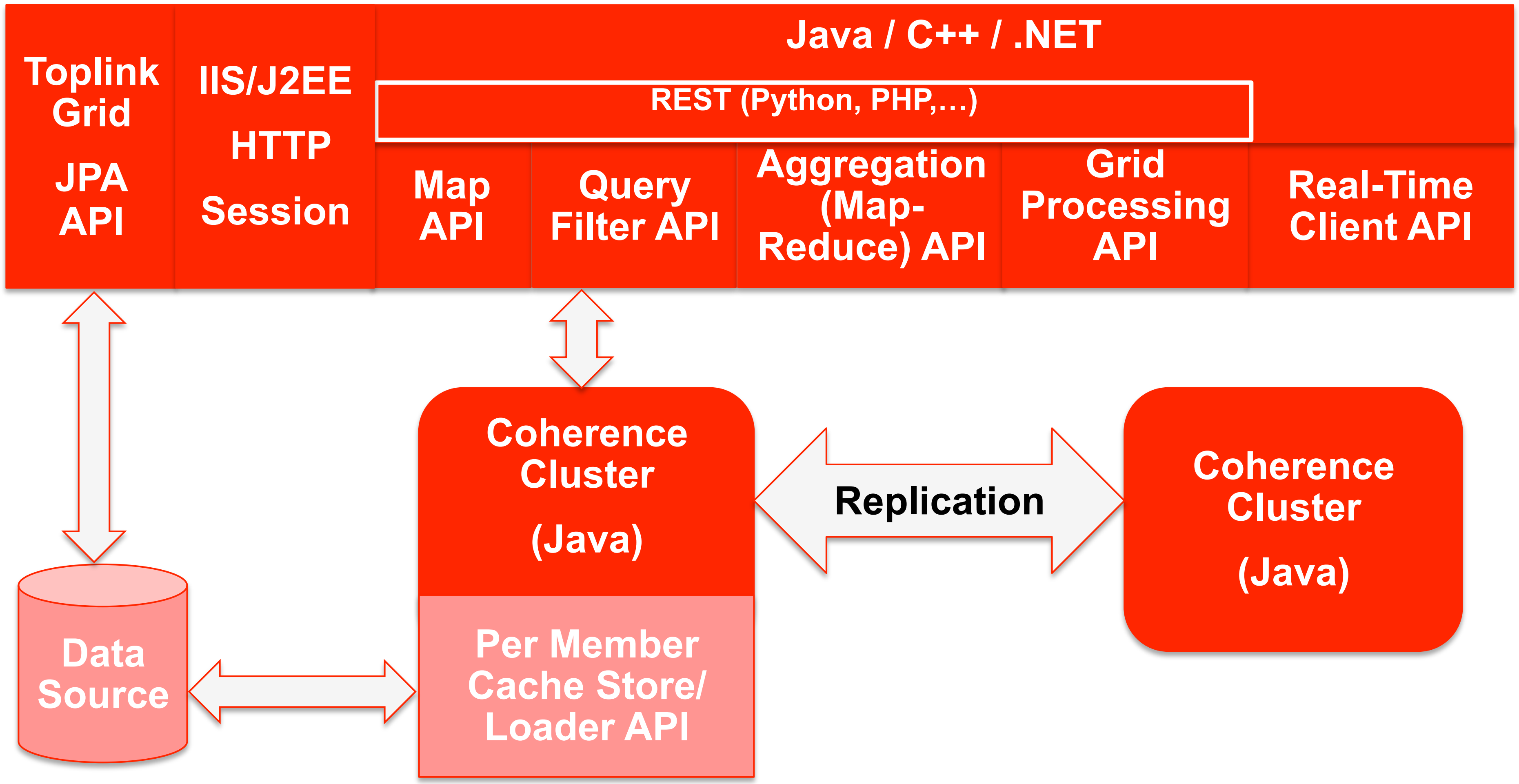
Application Infrastructure for Scalability

- Java application that stores key/value pairs in memory as caches or Maps
- Coherence JVM's communicate to provide fast, consistent and reliable access to data
- It distributes cache data across JVM's – and servers - dynamically
- Enables scalable and very fast processing inside caches – like stored procedures
- Provides real-time eventing, query, and map/reduce aggregations
- Back-end data source offload
- Dynamically scalable platform with seamless failover and recovery for data and processing



Using Oracle Coherence

Scalable Distributed Caching for packaged, database orientated and real-time event driven applications



What is the Live Object Pattern?

Object Taxonomy

- Data Transfer Object
  - AKA Value Object
- Domain Object
- Live Object

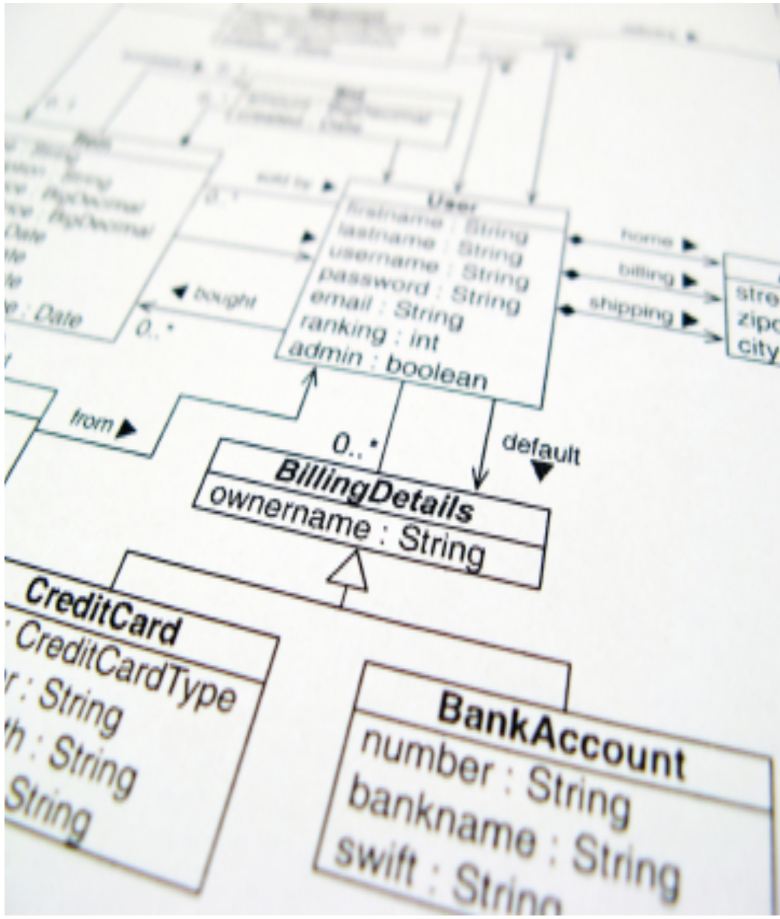
ORACLE

Data Transfer Object

- Discrete
- Serializable
- Only holds data
- **Most common type of Object in a distributed cache**

Domain Object

- Encapsulation
  - Data
  - Behavior
- Associated with other Domain Objects
- Model of the "real world"



Live Object

- Encapsulation
  - Data
  - Behavior
  - **Events**
- Can react to
  - Data change
  - Behavior from other objects

Traditional Distributed Caches

- “What happens in the cluster, stays in the cluster!”
- Data is accessed and modified via a service layer
  - `get()`
  - *Modify data*
  - `put()`

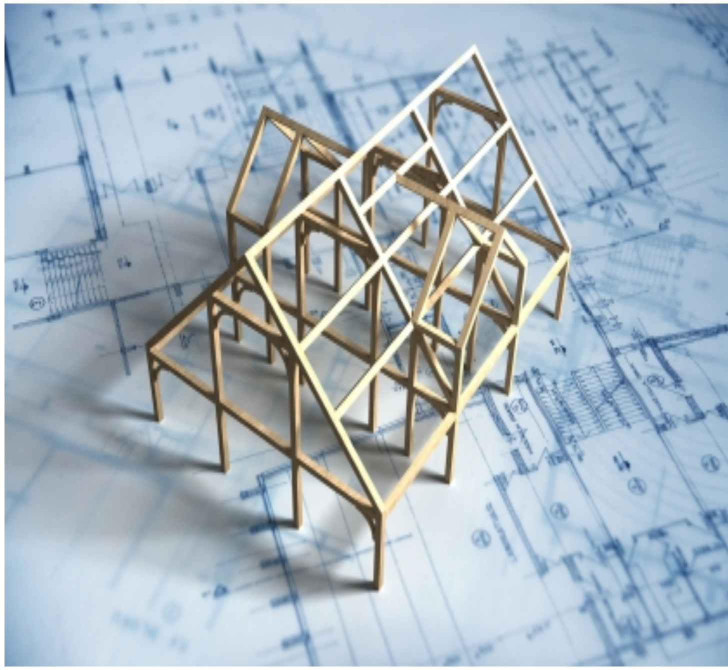
ORACLE

Live Object Data Grid

- Data and services live together
- Mutation and processing of data changes happens in grid
- More of a “control center” than a “repository”

Live Objects As a Foundation

- Building blocks for an **event-driven finite-state machine**
- …that is distributed
- …and scalable
- …and resistant to machine failure
- Architectural possibilities:
  - Staged Event Driven Processing
  - Ripple Effect



More Possibilities

- Asynchronous Initialization Tasks
- Scheduling of Background Work
- Subscriptions to External Systems
- Distributed “Workers”

Live Objects on Oracle Coherence

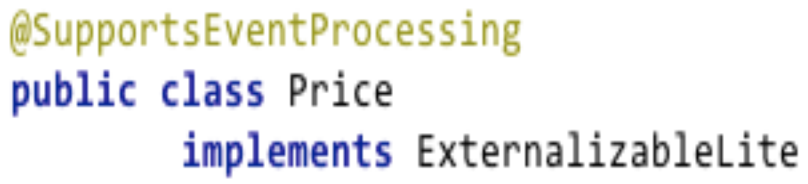
- A Live Object is an Object in a Coherence cache
- After each mutation Coherence will back up state
- Since the state is stored, it is recoverable!
- Each object is autonomous

Coherence Incubator

- The Coherence Incubator hosts a repository of example implementations
  - Design Patterns
  - System Integrations
  - Distributed Computing
  - …and Live Objects!

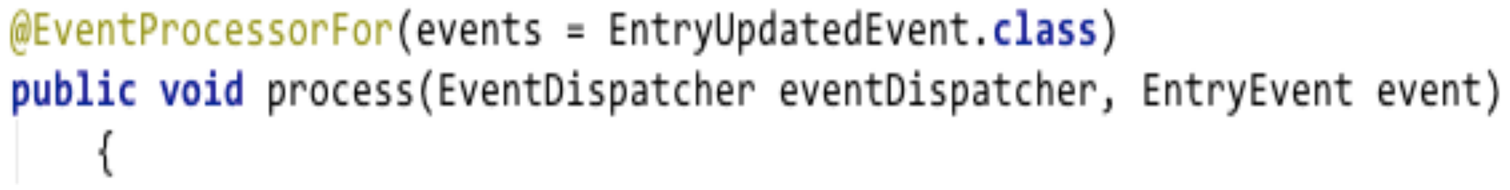
The Making of an Incubator Live Object

- Annotation to declare a Live Object



```java
@SupportsEventProcessing
public class Price
        implements ExternalizableLite
```

- Annotation for event handling



```java
@EventProcessorFor(events = EntryUpdatedEvent.class)
public void process(EventDispatcher eventDispatcher, EntryEvent event)
    {
```

ORACLE®

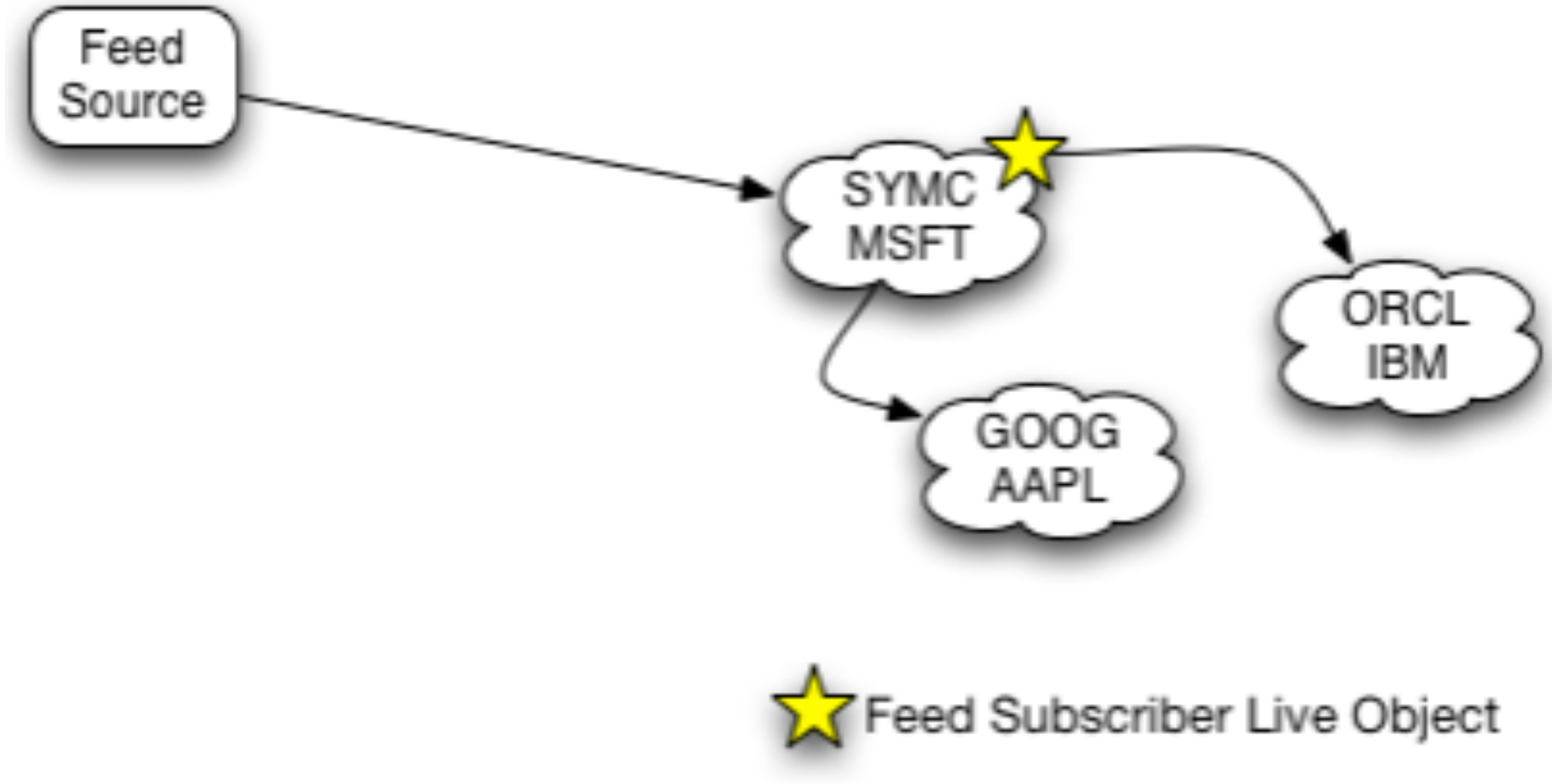
Live Objects Demo

- Stock Feed
  - Singleton Live Object
- Tickers and Prices
  - Live Objects

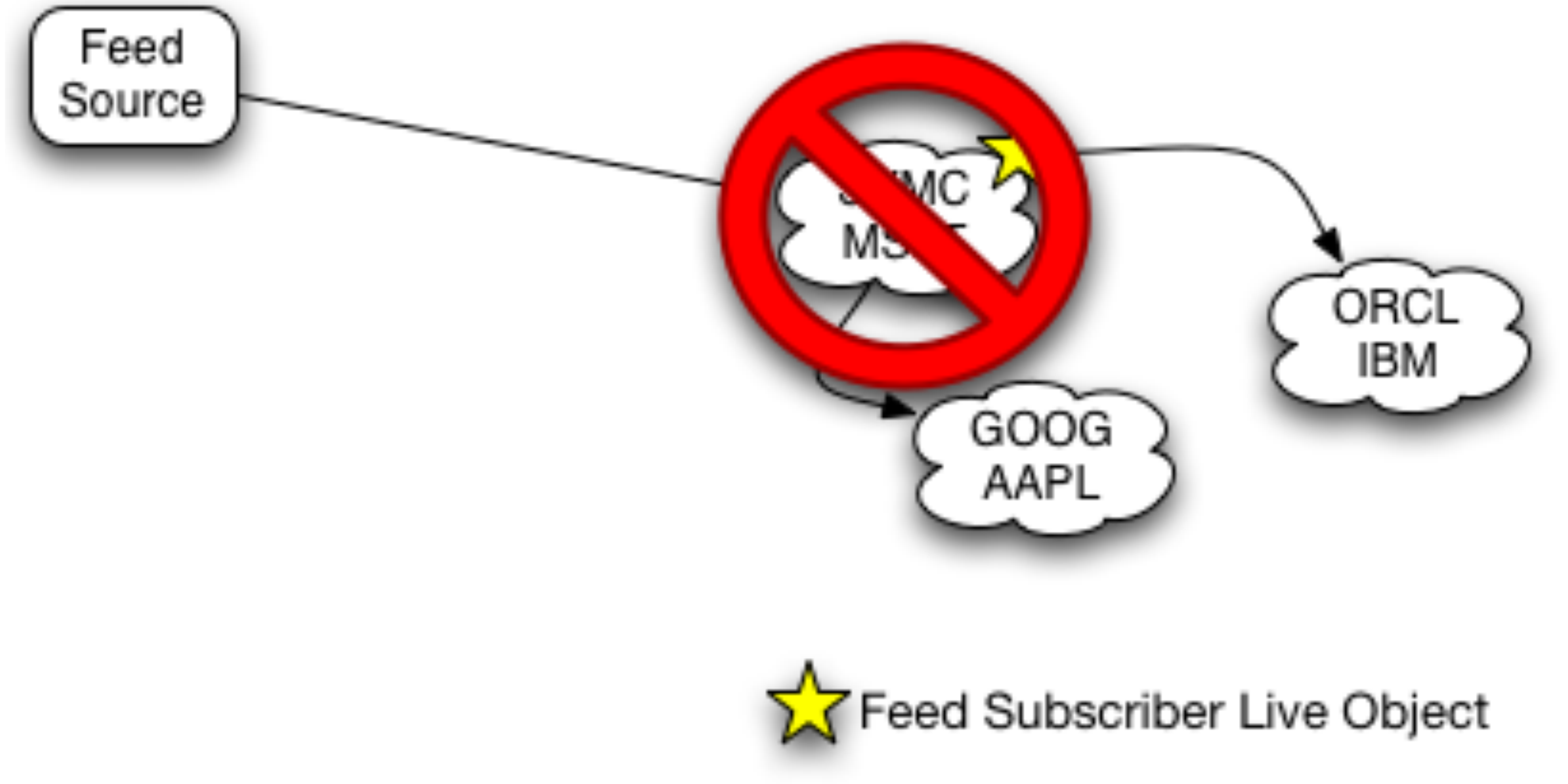
Stock Feed Demo

- Feed object subscribes to external data feed
  - Each connection costs $
  - If connection is lost, must reconnect automatically
- When price updates arrive, existing prices are updated
- Price objects are Live
  - Each Price can take action based on updates
  - Example: alert when price changes by > 5%

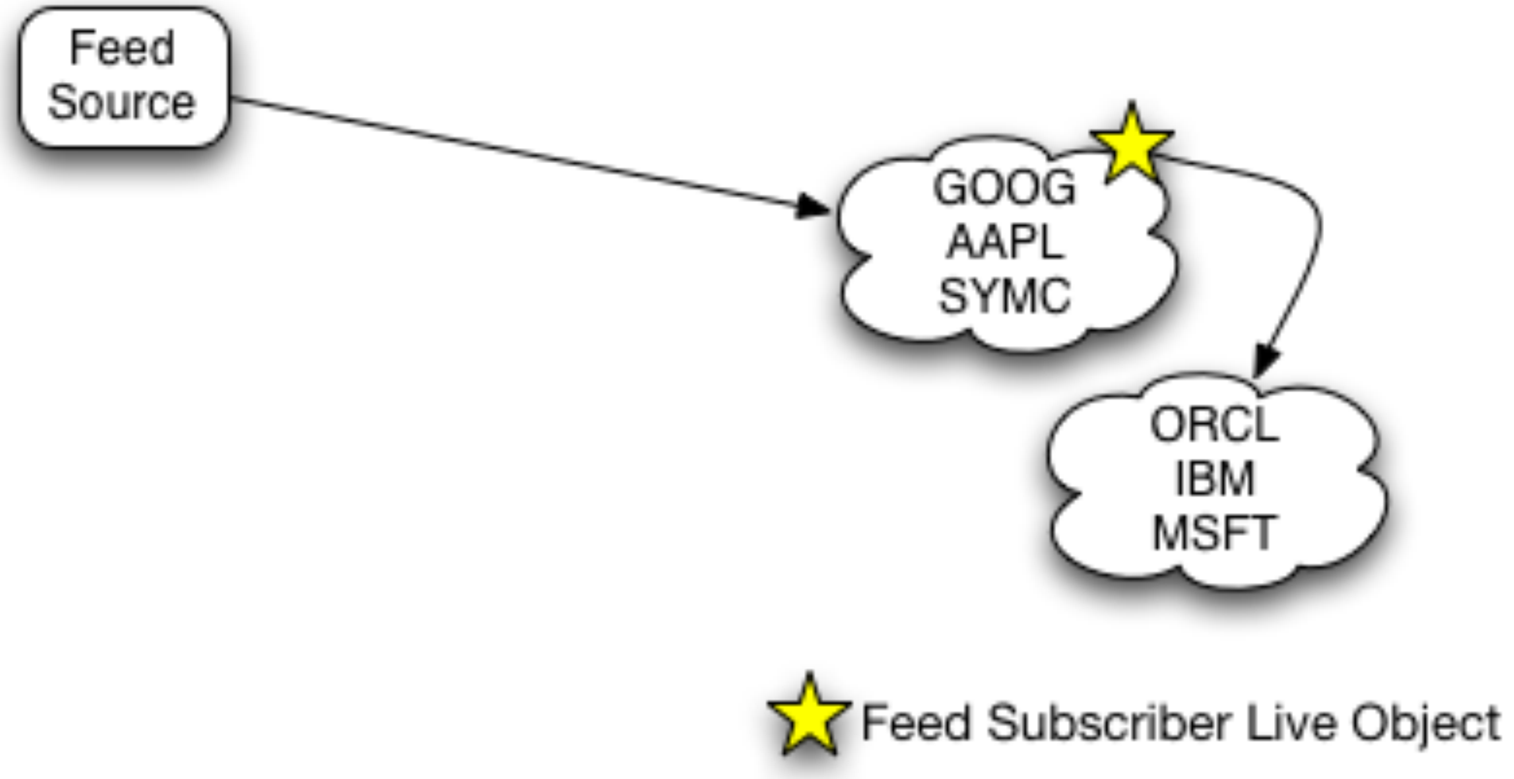
Stock Feed Demo



Stock Feed Demo

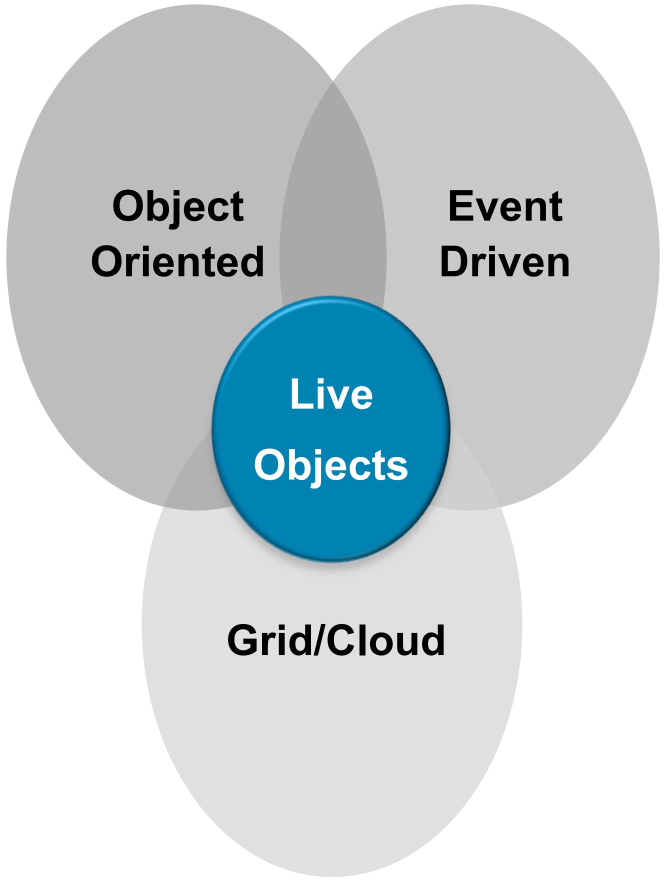


Stock Feed Demo



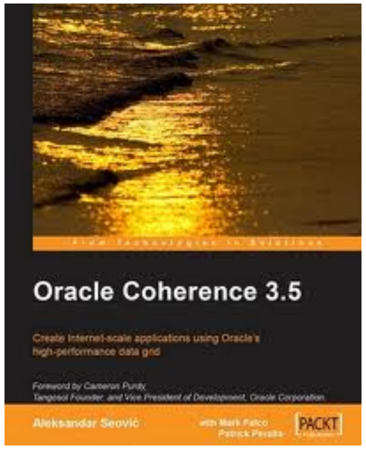
Conclusion

- Traditional distributed caches are simple data repositories
- Live Objects in a Data Grid
  - Event Driven programming
  - Distributed finite-state machine
  - Scalable and reliable



For More Information

- General Information: http://coherence.oracle.com
- Coherence YouTube Channel: http://www.youtube.com/user/OracleCoherence
- Coherence Training: http://education.oracle.com
- Coherence Discussion Forum: http://forums.oracle.com
- Coherence User Group on Linkedin
- "Oracle Coherence 3.5" by Aleks Seovic
- My email: david.felcey@oracle.com


Oracle Coherence 3.5
Create Internet-scale applications using Oracle's high-performance data grid
Foreword by Cameron Purdy
Tangosol Founder, and Vice President of Development, Oracle Corporation
Aleksandar Seović
with Mark Falco
Patrick Peralta
PACKT