In-Memory Grid Computing und Java EE

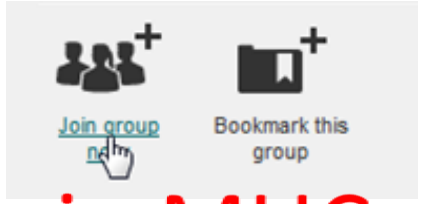
Michael Bräuer
Oracle Deutschland B.V. & Co. KG

Java Application Server Plattform Community

Eine Community von ORACLE für Kunden, Partner und Interessierte

Code Camps
Demos
WebLogic Server
GlassFish Server
Community Treffen
Java EE
JSRs
Vorträge
Serverseitige Entwicklung
Administration
Wissensaustausch

Registrierung: https://www.xing.com/net/oraclejavaappserver
Blog: http://fmwtech.wordpress.com
Ansprechpartner: michael.braeuer@oracle.com
peter.doschkinow@oracle.com



Coherence und WebLogic SIG, 4.Nov in MUC

Safe Harbor Statement

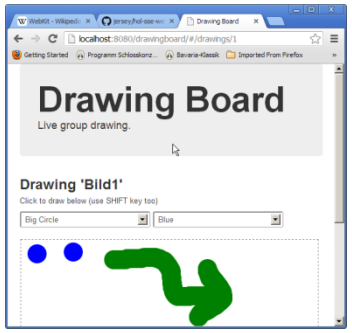
The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

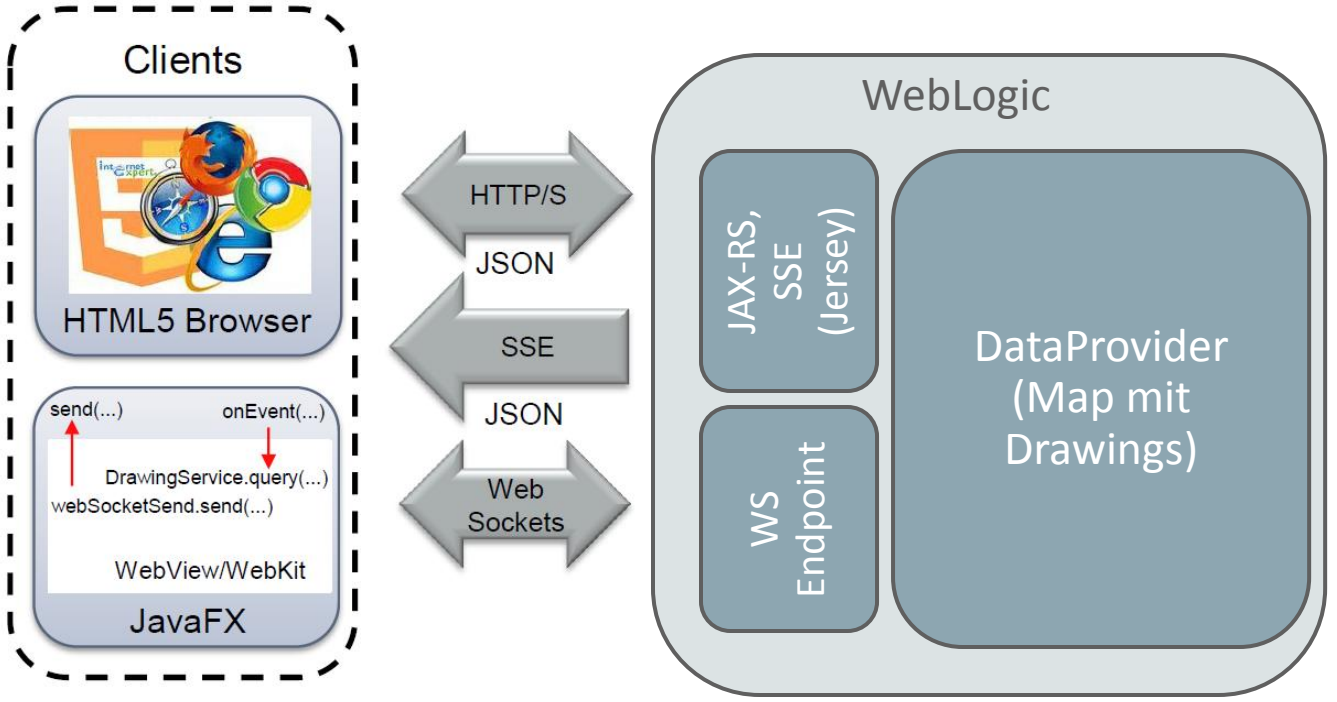
Agenda

- Status Quo: Zustand in verteilten Java EE Umgebungen
- In-Memory Grid Computing und Java EE mit Oracle Coherence

Serverseitiger Zustand in modernen Java EE Anwendungen

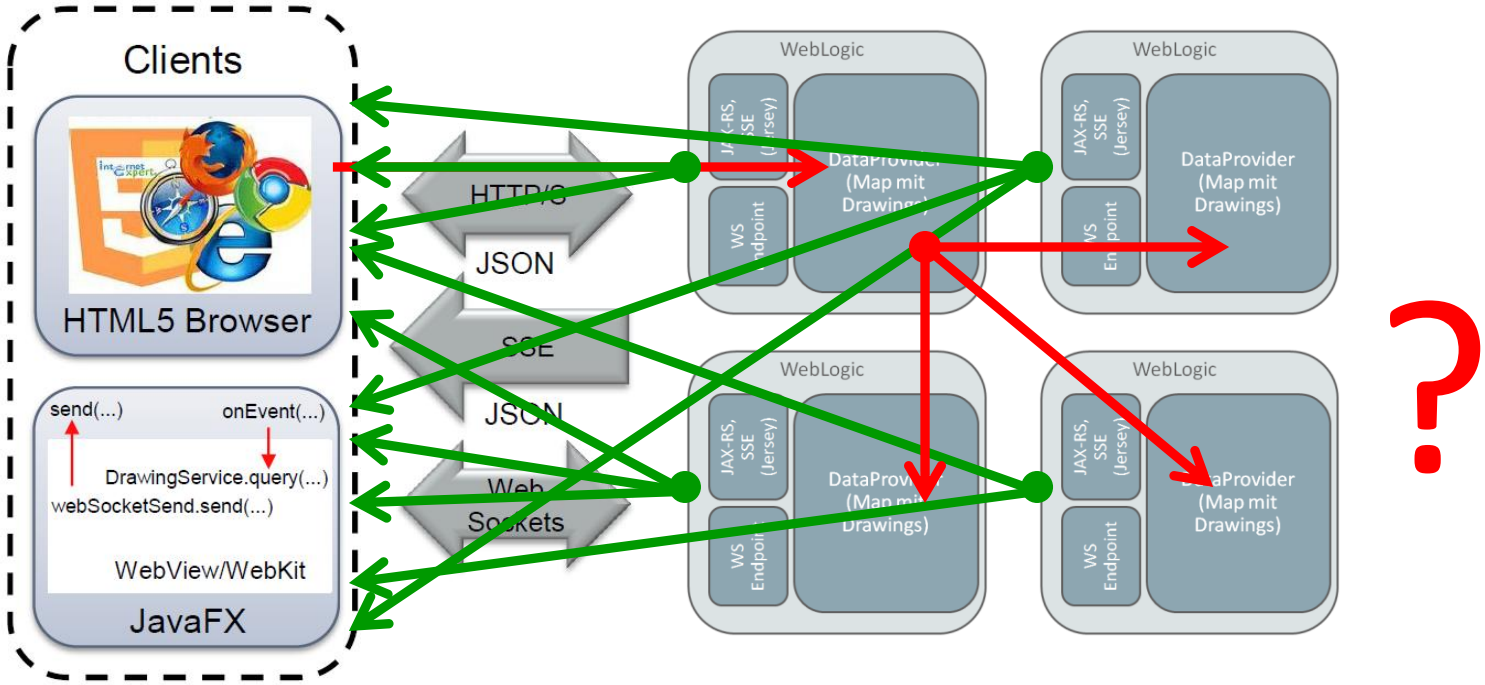
- Beispiel Drawingboard (https://github.com/doschkinow/hol-sse-websocket):
  - Client: HTML5 (angular.js)
  - Server: JAX-RS 2.0 (JSR- 339) und SSE, Websockets 1.0 (JSR-356), u.a.



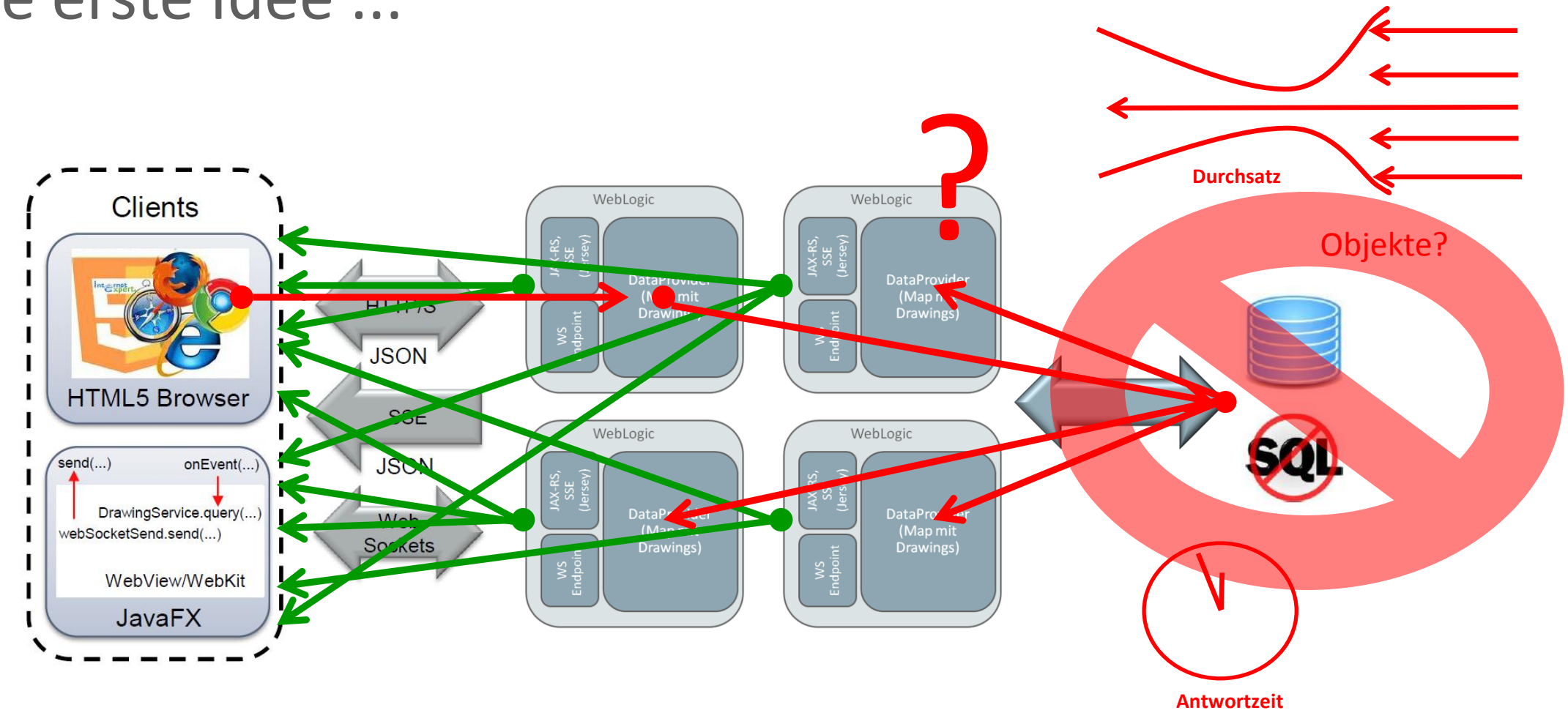


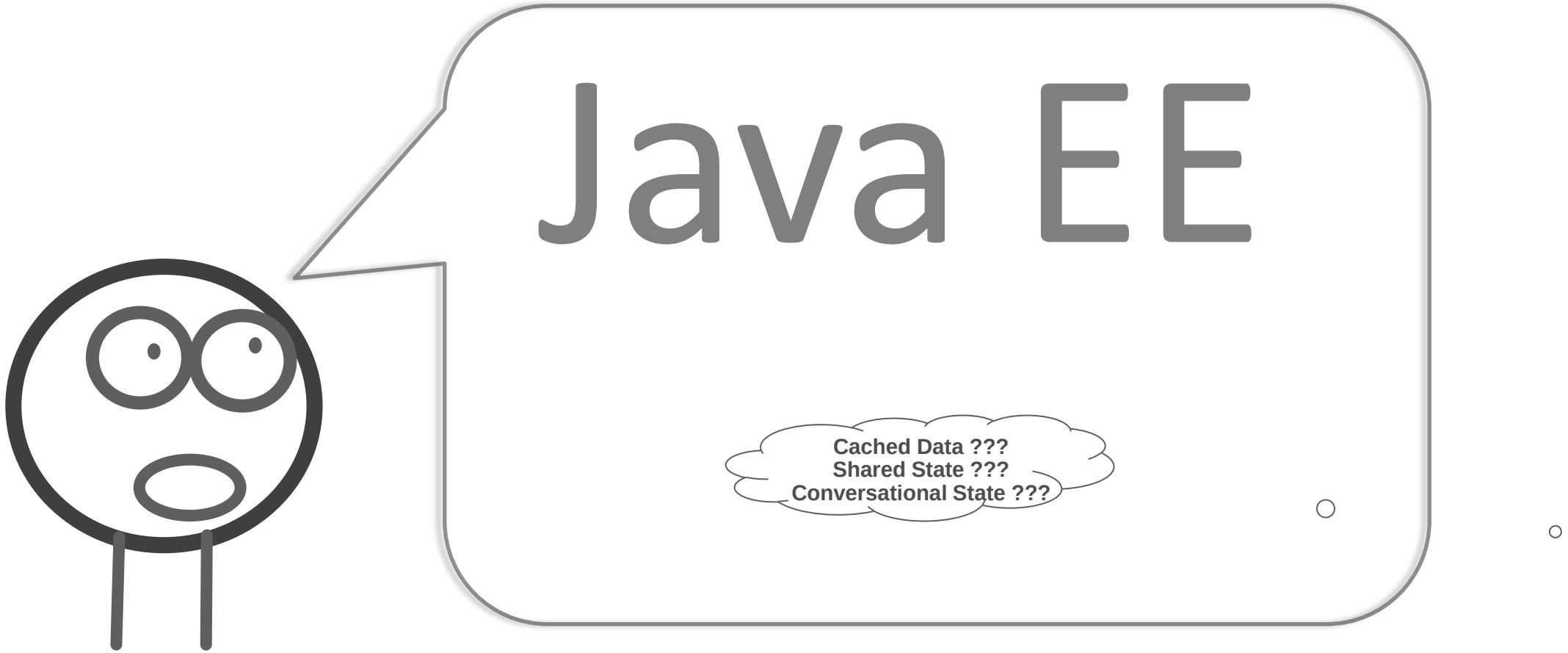
Java EE

Zustand, der in verschiedenen JVMs gebraucht wird



Eine erste Idee …





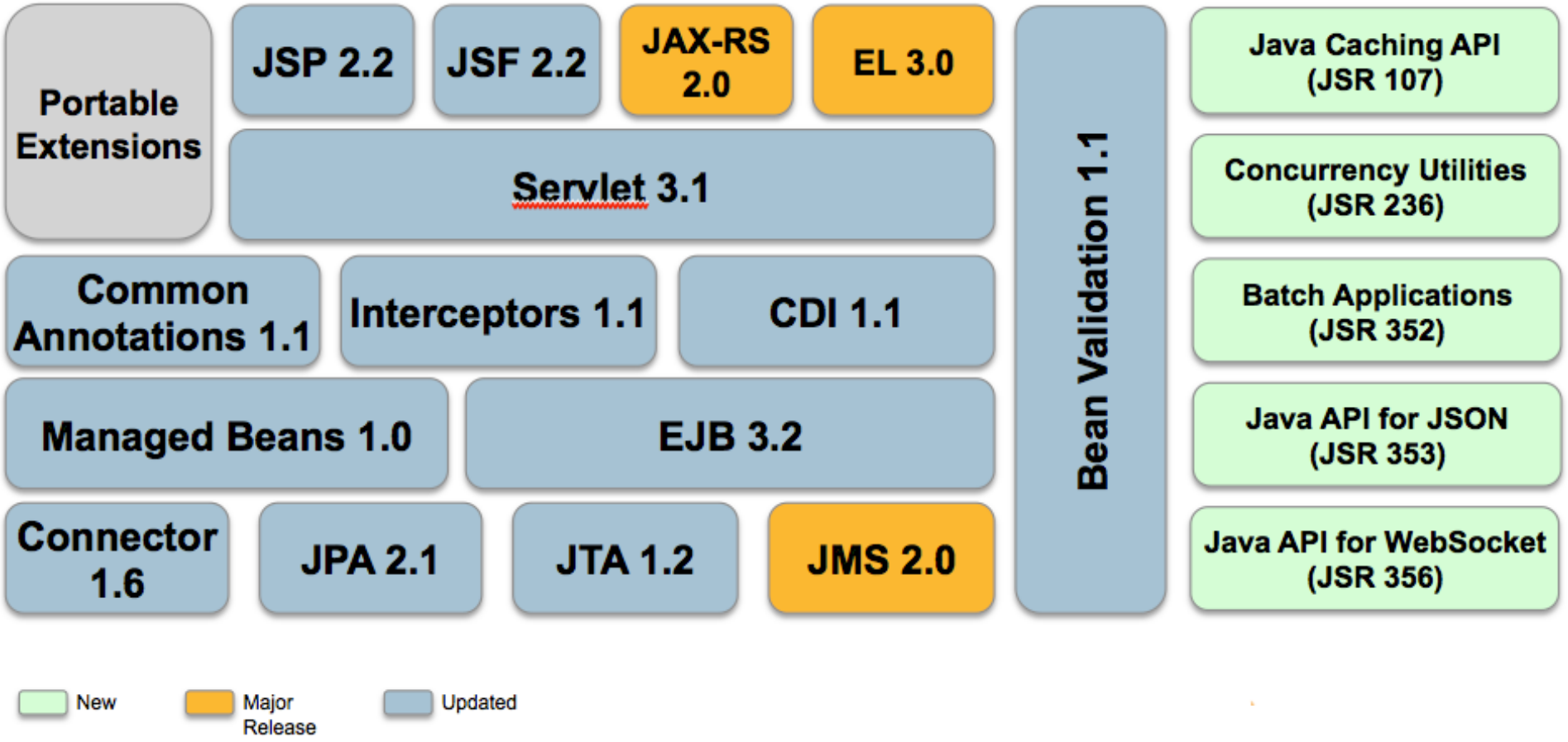
Java EE 7 Fokus

- More annotated POJOs
- Less boilerplate code
- Cohesive integrated platform

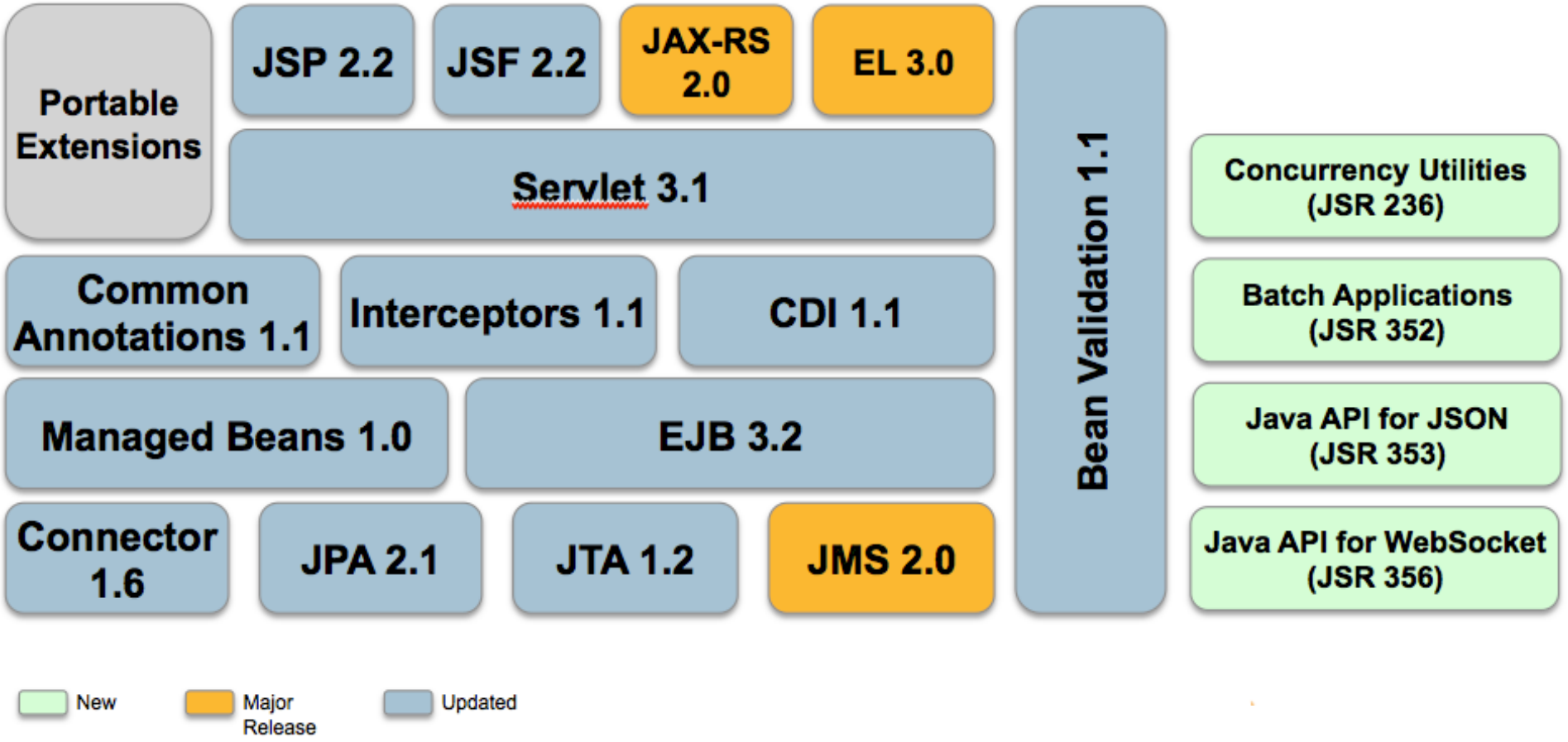
- WebSockets
- JSON
- Servlet 3.1 NIO
- REST

- Batch
- Concurrency
- Simplified JMS

Java EE 7 JSRs – Prefinal

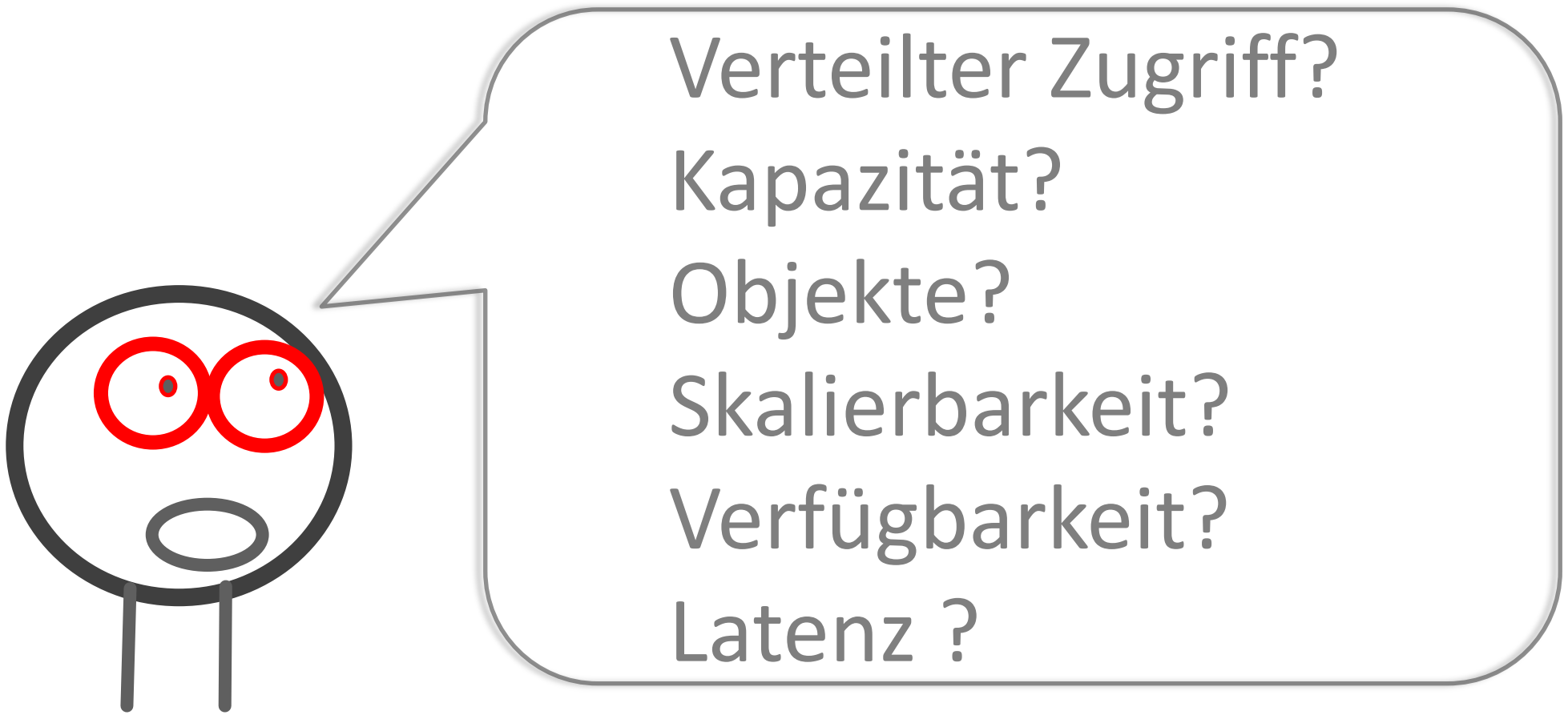


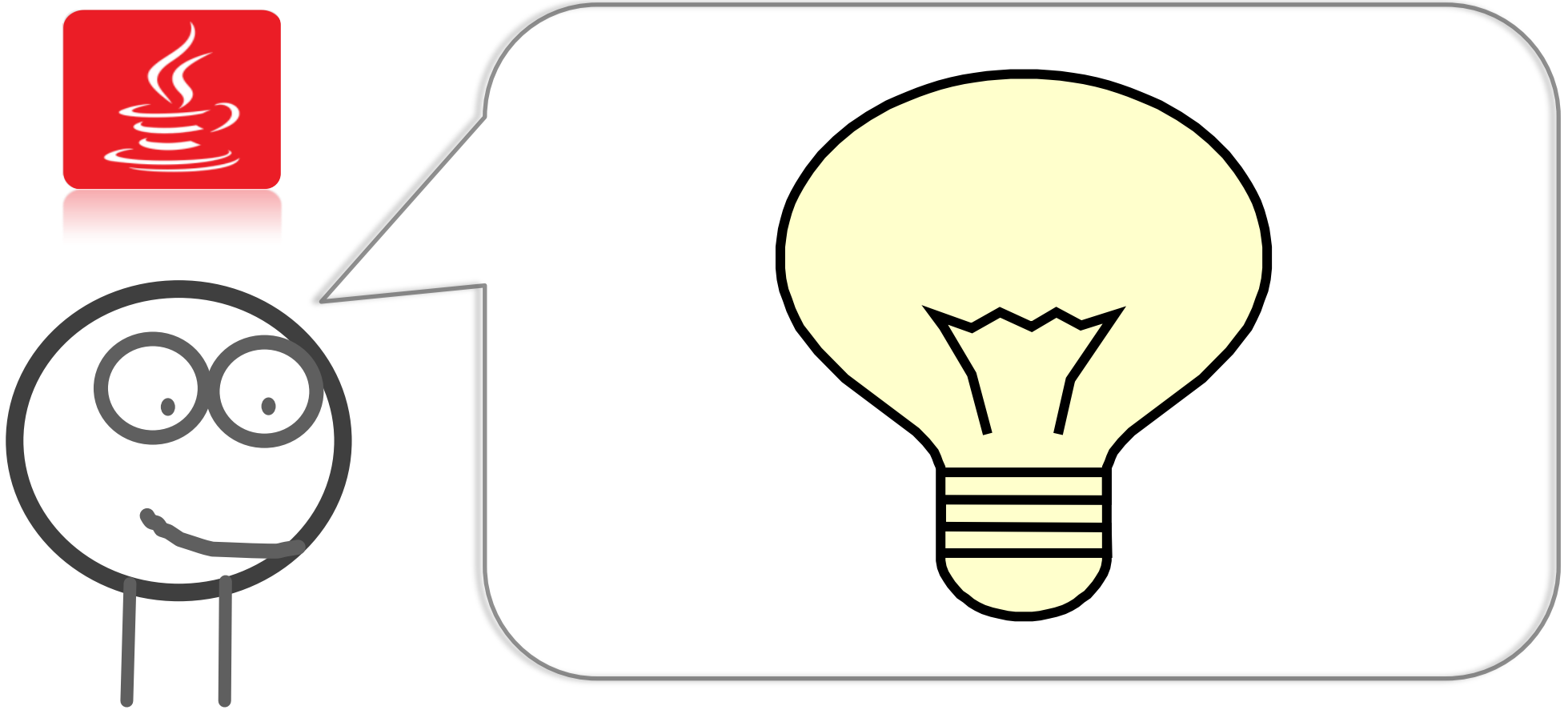
Java EE 7 JSRs – Final



Status Quo: Zustand in der Java EE Welt

- JSR für Caching innerhalb von Java EE?
  - Frühjahr 2014
    - Finalisierung JSR-107: Caching in Java SE
    - Keine Aussagen über Zusammenarbeit mit Java EE → Java EE 8 (?)
- JSR für Java Grid Computing (was passiert in JSR-347: „Data Grids for the Java Plattform“)?





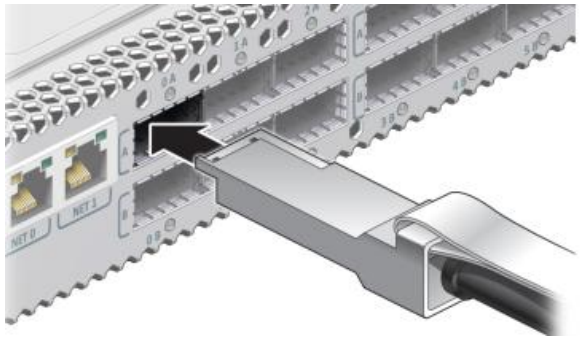
In-Memory Data Grid*

* angelehnt an http://www.jroller.com/cpurdy/entry/defining_a_data_grid

- In-Memory basiertes Management System für verteilt genutzte Objekte mit
  - geringen Antwortzeiten
  - hohem Duchsatz
  - vorhersehbarer Skalierbarkeit
  - Fehlertoleranz: Verfügbarkeit und Zuverlässigkeit.

Anforderungen/Herausforderungen

- Hohe Kapazitäten: TBs
  - In-Memory: On-Heap
  - alternative Medien: SSDs, Flash
  - Schnelles Netzwerk (!!)

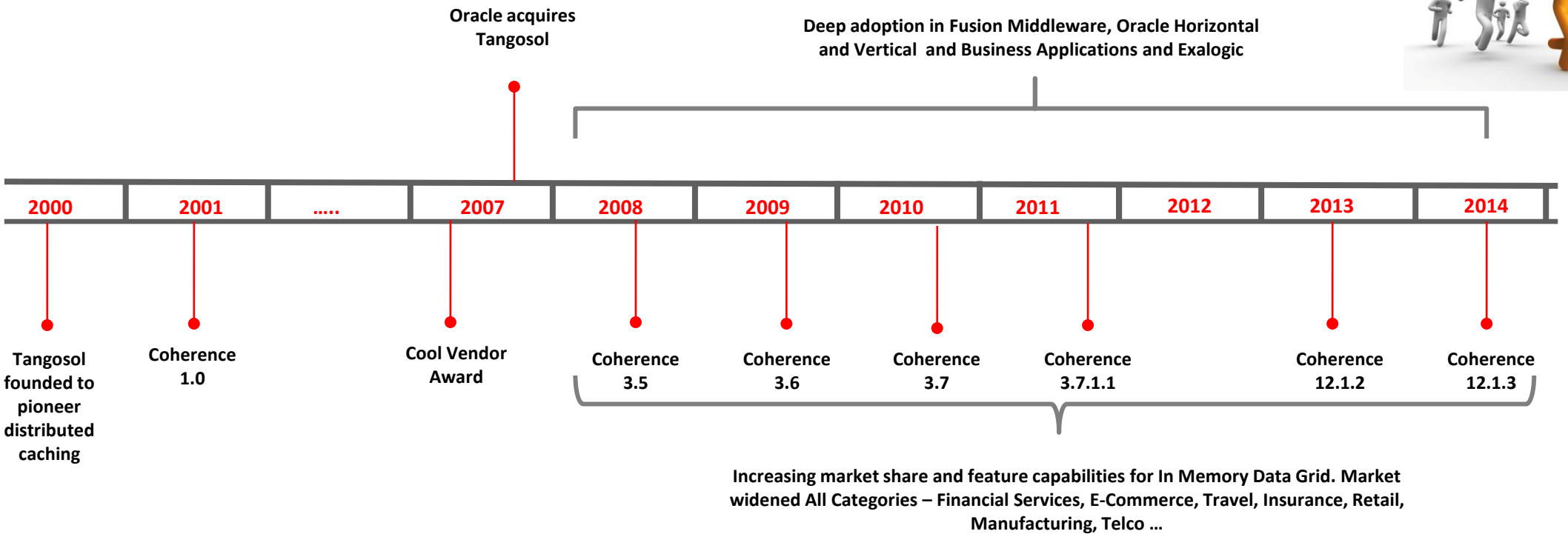


- Verteiltes Rechnen: Datenmanagement und parallele, auch asynchrone Verarbeitung, z.B. mittels EntryProzessor, Filter, Aggregationen
- Integration in Java SE und Java EE Umgebungen
- Vielseitiger Clientzugriff: Java, C++, .Net, REST, memcached
- Wie serialisieren?

Anforderungen/Herausforderungen

- Integration mit persistenten Systemen (Vermeidung Stale Cache Problem)
- Einfache Konfiguration, Administration und Monitoring vieler JVMs

Pionier im Bereich Java In-Memory Data Grid Computing



Coherence in a Nutshell

- Coherence-Knoten:
  - Java Knoten:(Non-)Storage Knoten und optional Proxy-Knoten: XML-Konfigurationen und Bibliotheken im Klassenpfad
  - C++, .NET, REST, memcached Zugriff mittels Proxy-Knoten.
- Storageknoten können Daten aufnehmen/verwalten, spezielle Serialisierungsmechanismen sind möglich (POF), Evolvable
- Sehr effiziente Protokolle für Clusterkommunikation (TCMP)
- APIs:
  - Coherence spezifische APIs: NamedCache API (= Map<K,V> ähnliche Datenstruktur)
  - JCache API (JSR-107)

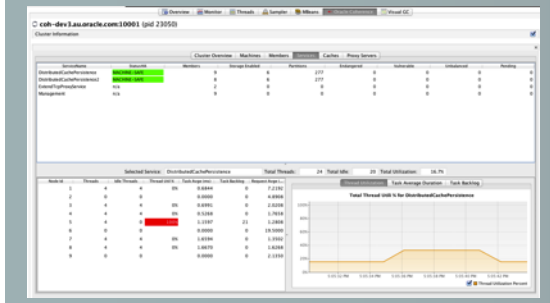
Coherence in a Nutshell

- Alle IDEs
- Builds mit Maven
- Konfiguration over Default
  - XML
  - Java System Properties
- Management mit JMX
  - JVisualVM Plug-In
  - Oracle Enterprise Manager Cloud Control und 3rd Party

JVisualVM Plugin

NEW

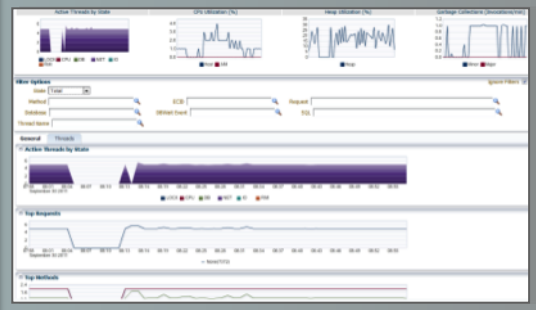
- Available now for 3.x on Coherence Community Website
- Lightweight plugin to JVM



Fusion Middleware Control

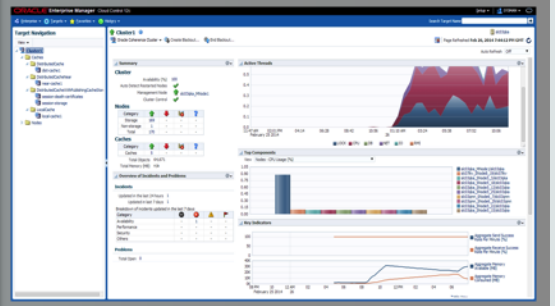
- OOTB administration and monitoring for all FMW
- Dev/QA point-in-time insight into cluster

NEW

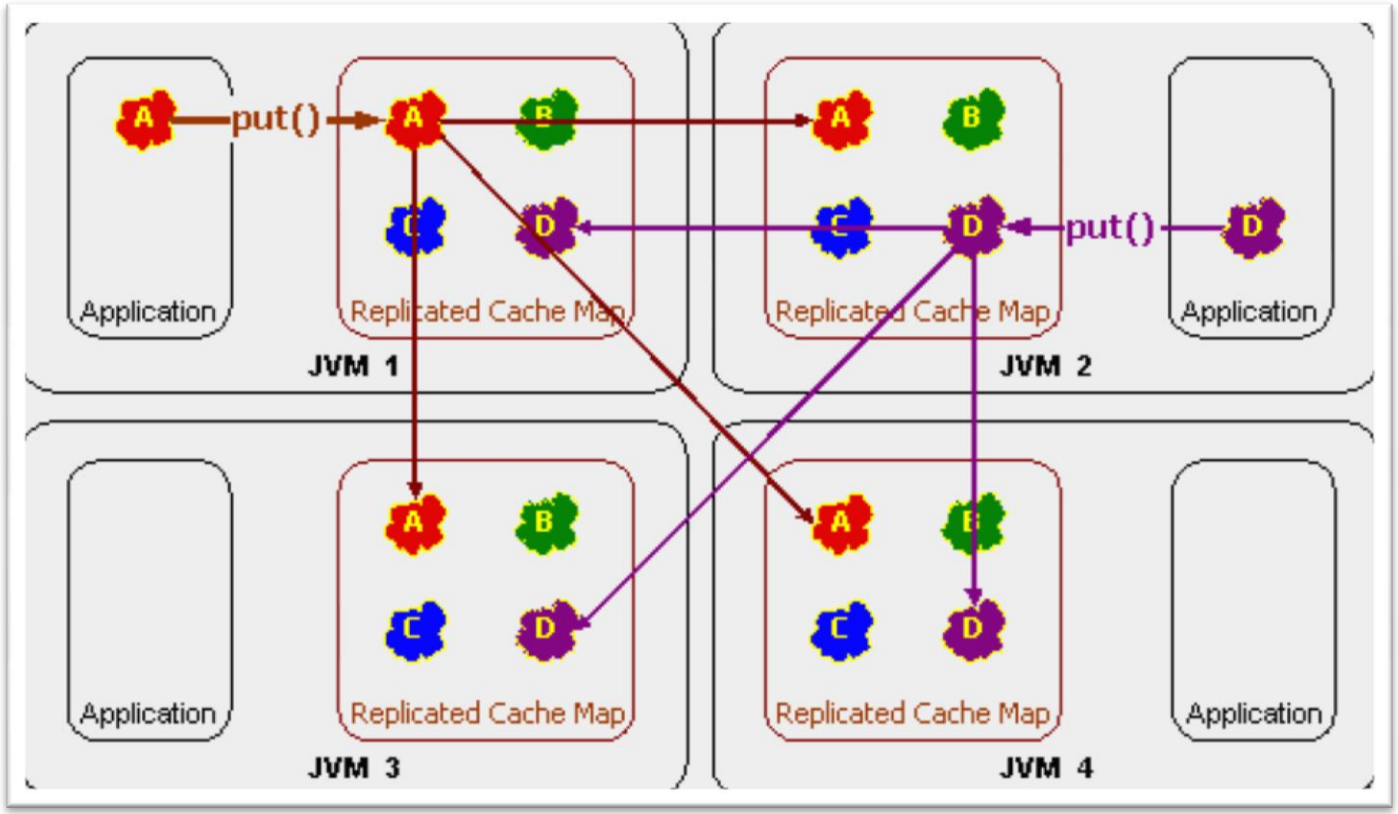


Coherence Management Pack for OEM

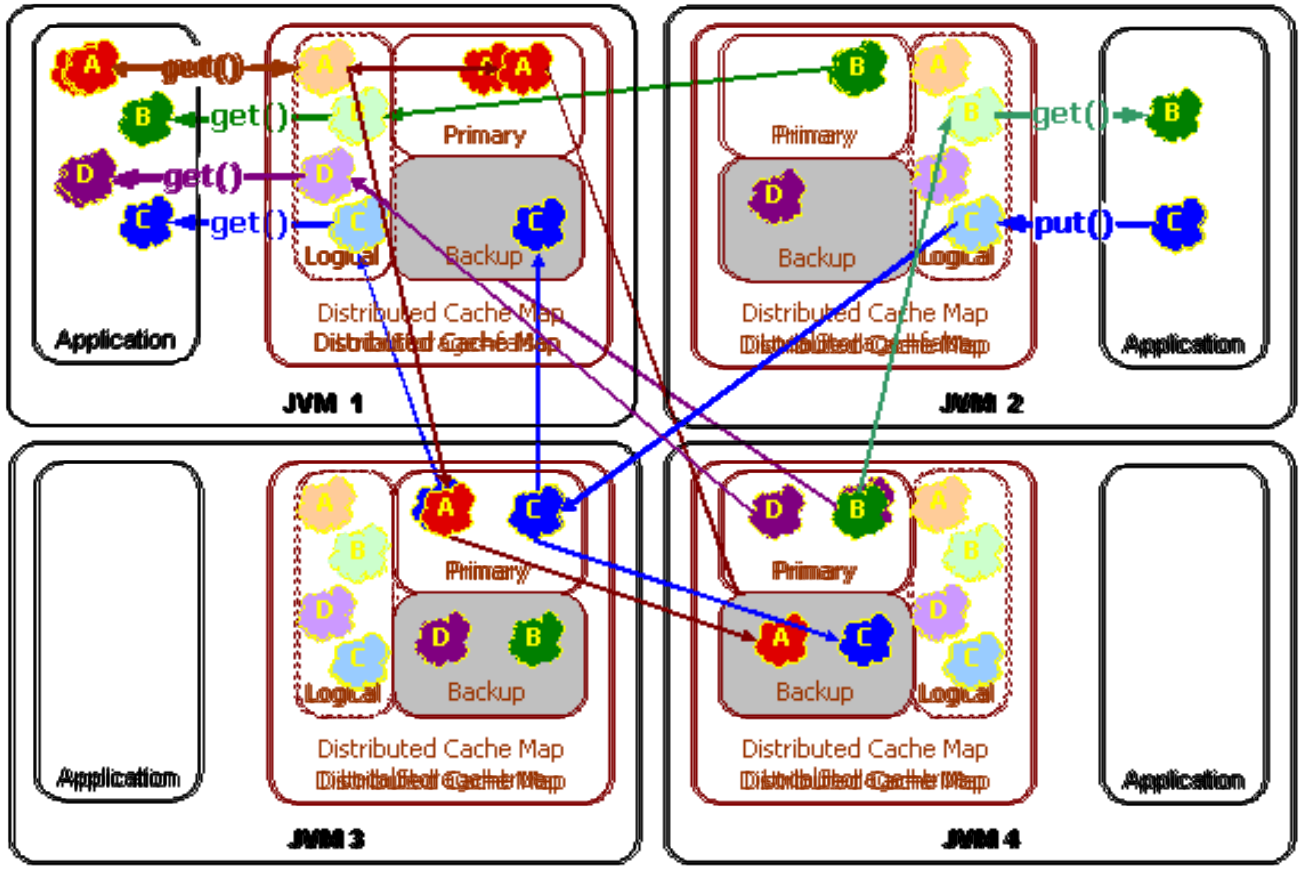
- Complete management and monitoring solution
- Store historical results
- Java diagnostics tooling



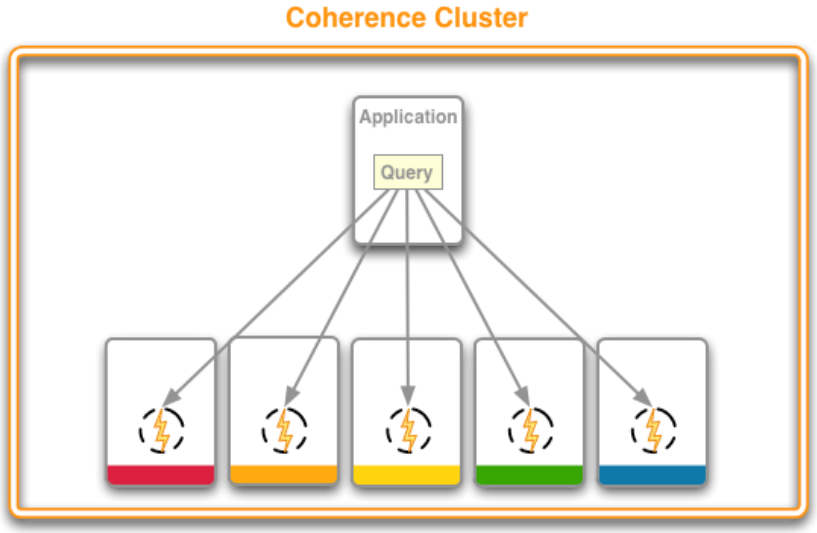
Coherence in a Nutshell

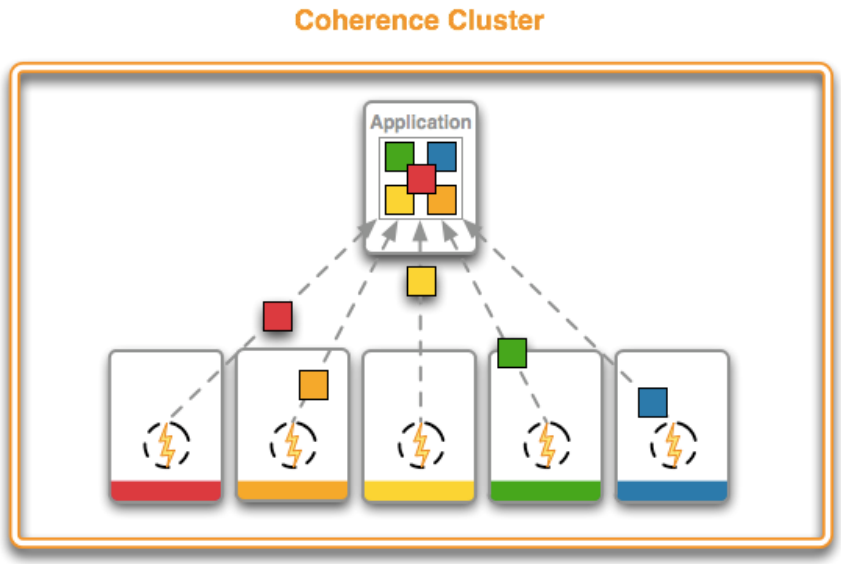


Coherence in a Nutshell



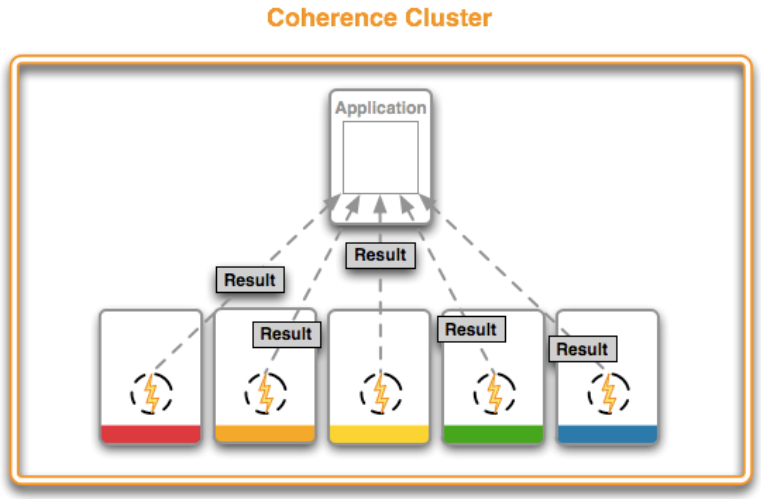
Coherence in a Nutshell





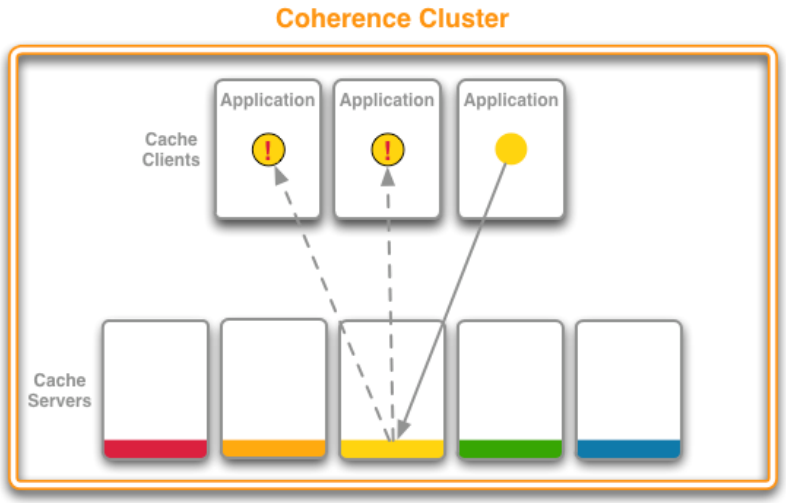
Parallele Verarbeitung: Filter mit Indizes, Aggregationen

Coherence in a Nutshell



Parallele Verabeitung: synchrone und asynchrone Entryprozessoren
Bringe Code zu Verabeitungsknoten und nicht Daten in den Client

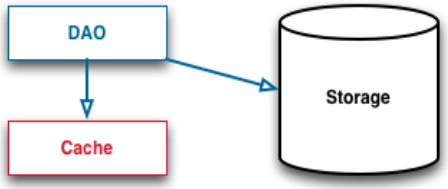
Coherence in a Nutshell



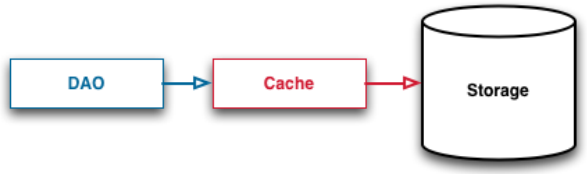
Map Events: entryInserted, entryUpdated, entryDeleted
Live Events: Partitioned Cache Events, Partitioned Service Events, Lifecycle Events, …

Coherence in a Nutshell

Cache Aside

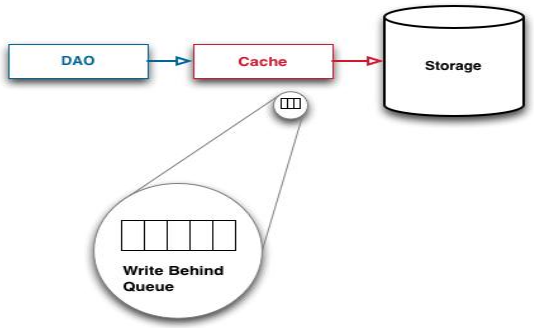


Write-Through

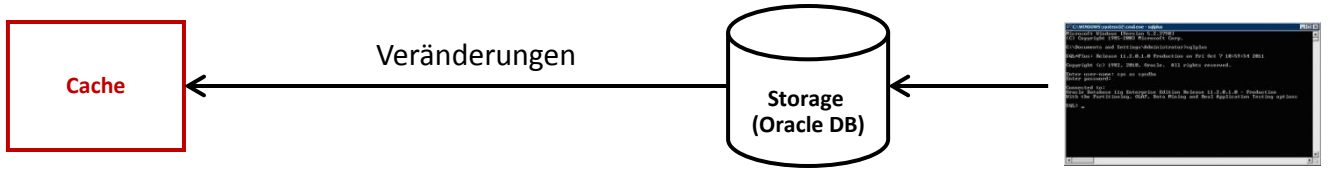


CacheStore

Write-Behind

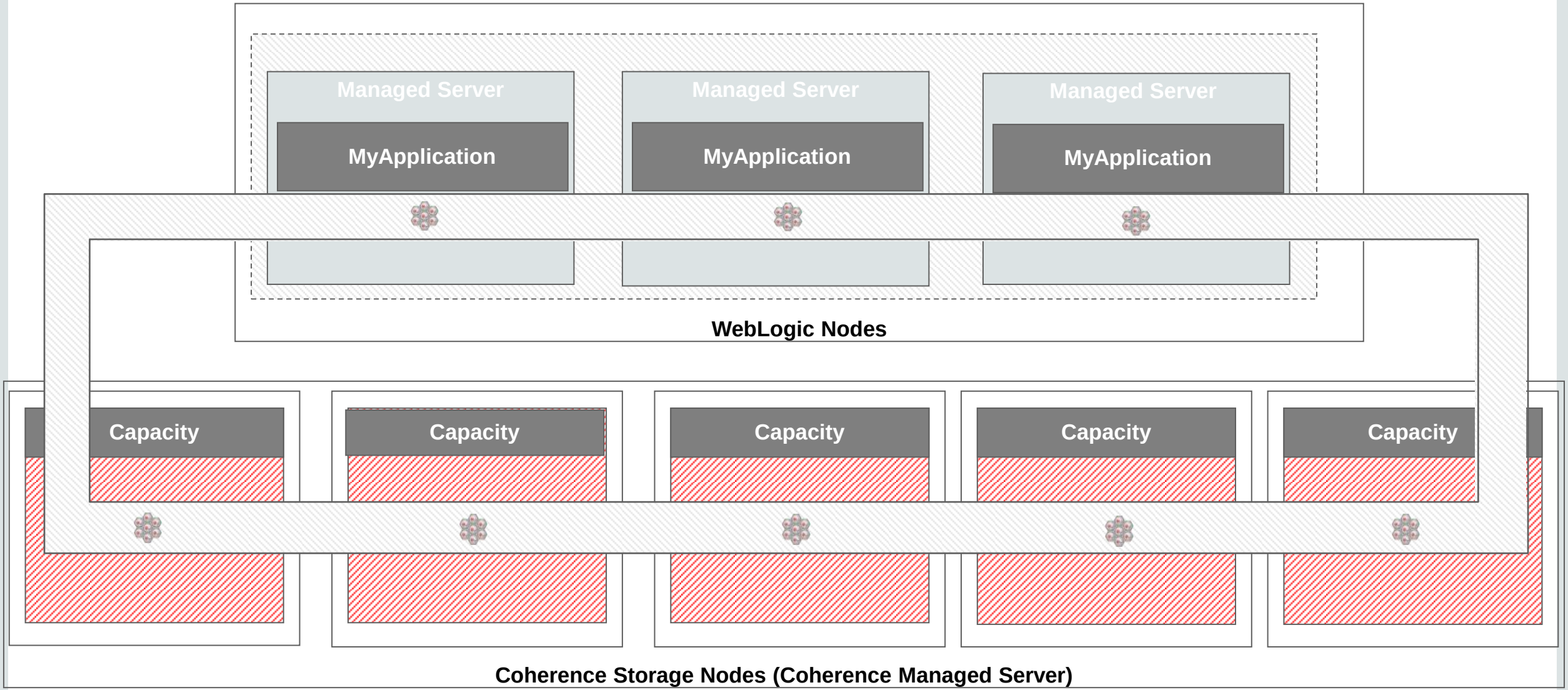


Coherence GoldenGate HotCache

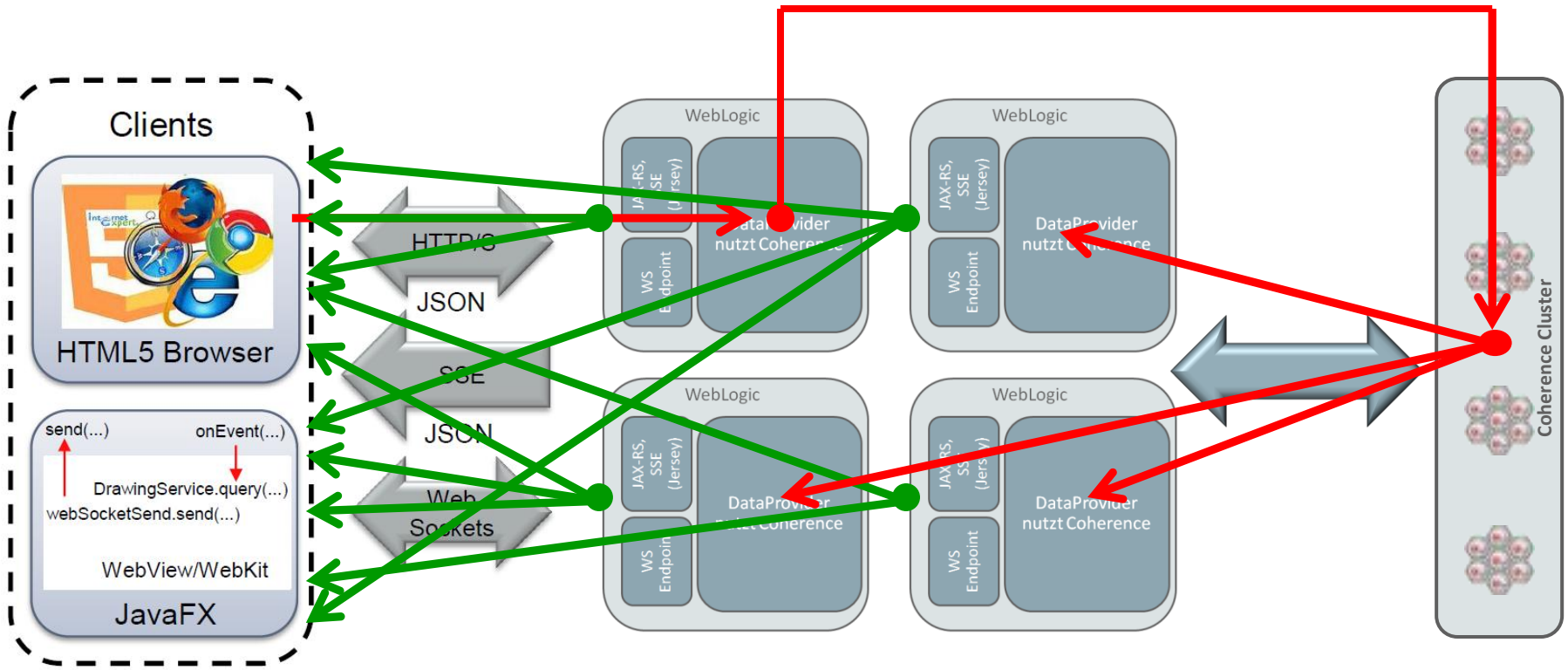


Veränderungen (z.B. Batch)

Java EE UND In-Memory Grid



Skalierbare Architektur: Eventbasiert



https://github.com/doschkinow/hol-sse-websocket, solutions/exercise9

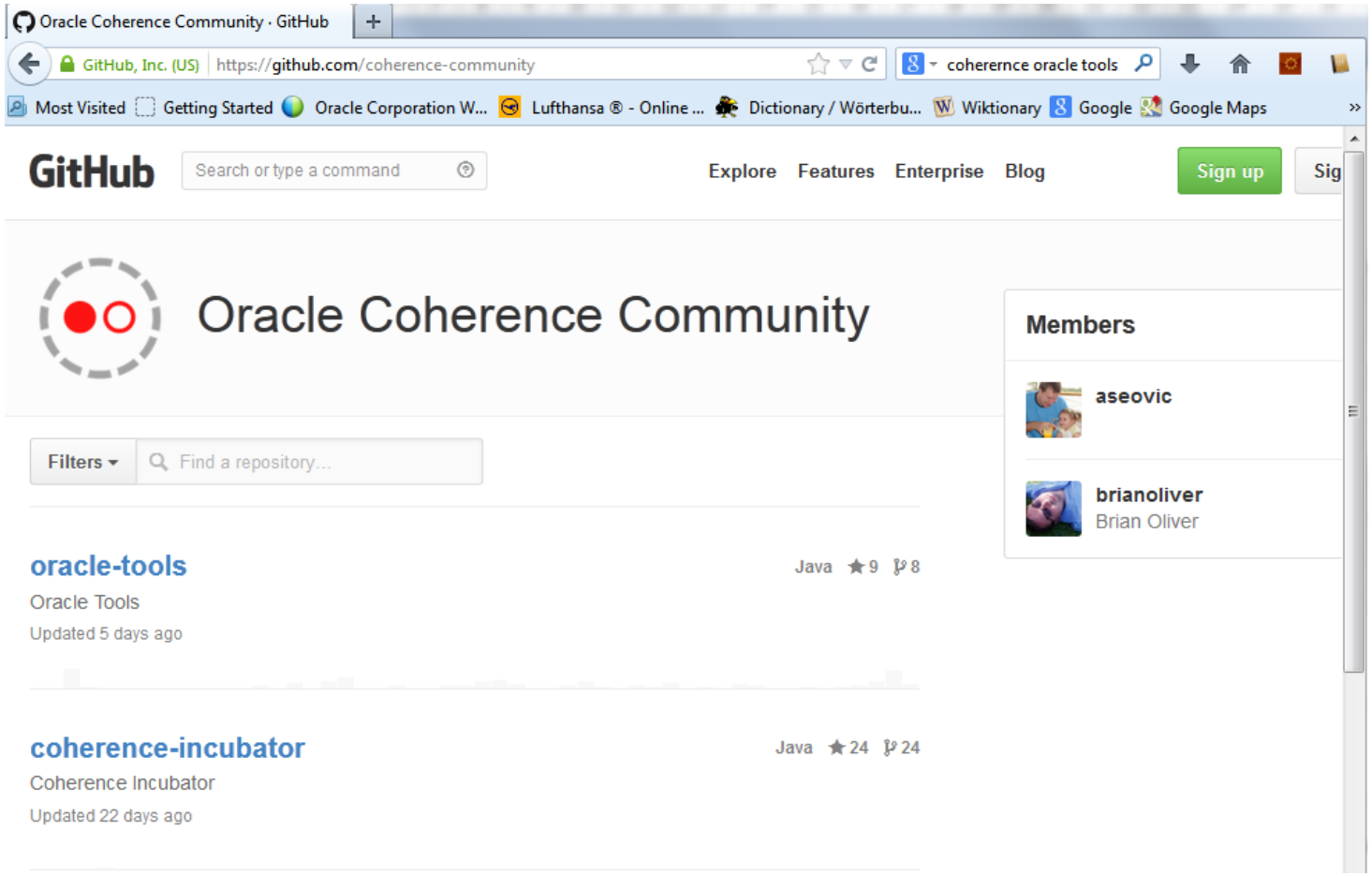
Einige kritische Erfolgsfaktoren - Betriebssicht

- Netzwerkkonfiguration und -planung
  - Bandbreite und Latenz: Coherence Netzwerk Tools kommen mit!
  - Konfiguration Netzwerkkarten (Linkaggregation unter Solaris (aktiv-aktiv vs. aktiv-passiv), z.B. IP Multipathing, DataLink Multipathing, etc. bei Solaris)
  - Konfiguration Switches
  - Rebalancing im Ausfall!
- Sorgfältige Kapazitätsplanung:
  - Anzahl der Knoten/JVMs: Kapazität, Planung für Fehlertoleranz
  - HW Planung: JVM Knoten per Rechenknoten (RAM, Prozessorkerne)
  - http://www.oracle.com/technetwork/middleware/coherence/planning-coherence-deployment-1985929.pdf

Einige kritische Erfolgsfaktoren - Entwicklersicht

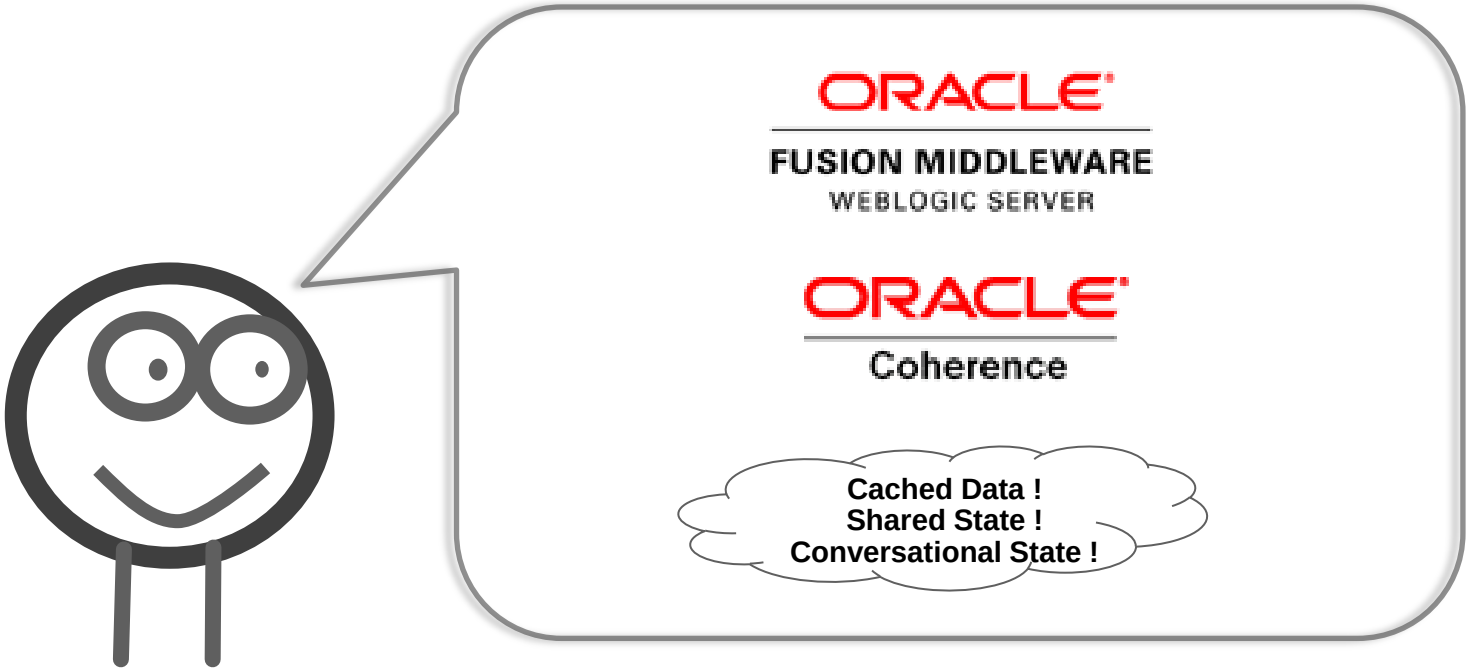
- Entscheide, was lokal oder verteilt benötigt wird
- Vermeide Standardserialisierungmechanismus: implementiere POF (Portable Object Format) mit 98:2 Regel
- Bringe Code zu den Daten, nicht Daten zum Code
- Denke über Parallelisierung nach
- Überdenke Dein Objektmodell
- Suche Einfachheit und Eleganz: kein „Overengineering“ von Anforderungen:
  - Bitte keine RDBMS nachimplementieren!

Hilfsmittel, Testen & Co



Demos

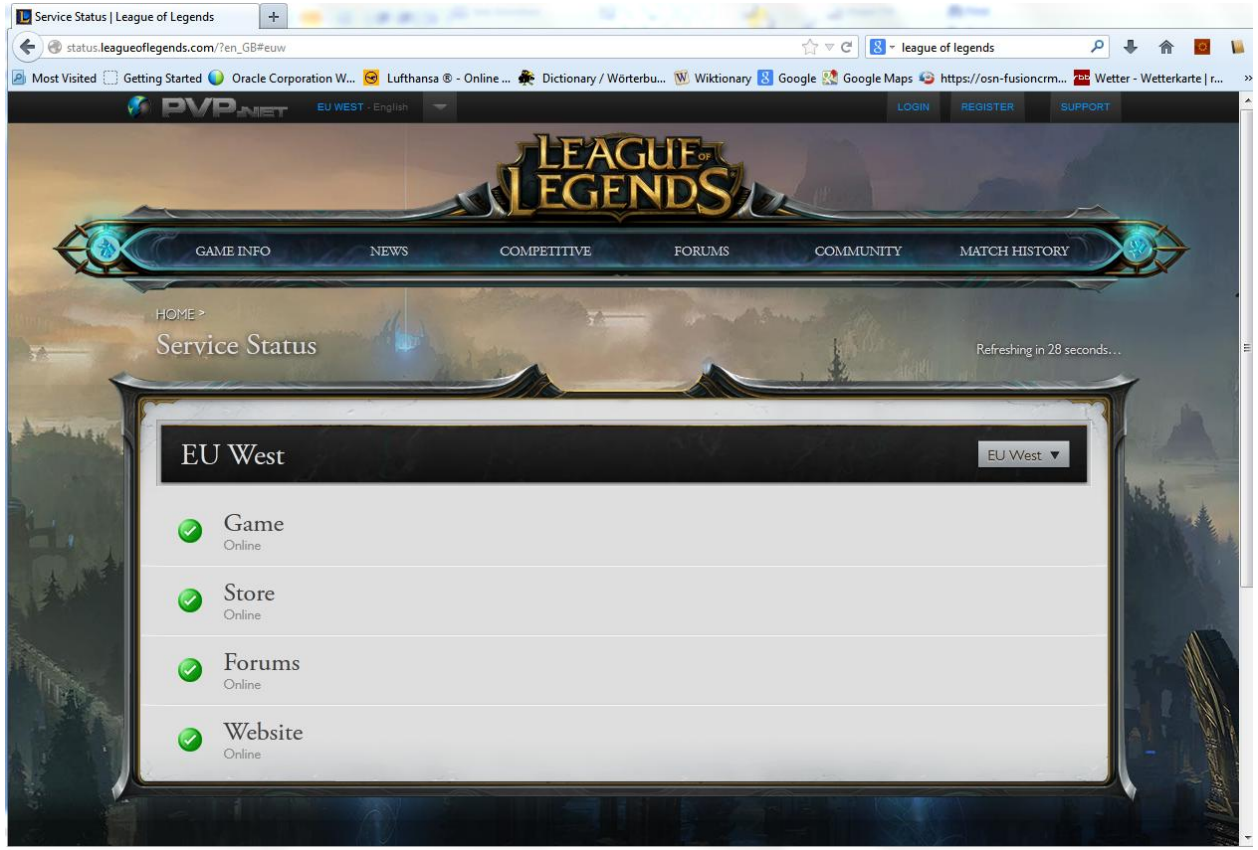
Zusammenfassung



Aber auch mit  , 

, ... ☺ . Try it.

Zusammenfassung



http://www.infoq.com/presentations/League-of-Legends
http://qconsf.com/sf2010/dl/qcon-sanfran-2010/slides/RandyStafford_and_ScottDelap_LeagueOfLegendsScalingToMillionsOfNinjasYordlesAndWizards.pdf
http://www.oracle.com/technetwork/middleware/coherence/coherence-case-studies-091909.html

Join the Coherence Community

@OracleCoherence

/OracleCoherence

/OracleCoherence

Oracle Coherence Users

blogs.oracle.com/OracleCoherence

coherence.oracle.com


ORACLE®