Orientation in Objects

Serverfarmen durch J2EE Clustering

Sabine Winkler (winkler@oio.de)
Christian Dedek (dedek@oio.de)

Orientation in Objects GmbH
Weinheimer Str. 68
68309 Mannheim
www.oio.de

Überblick

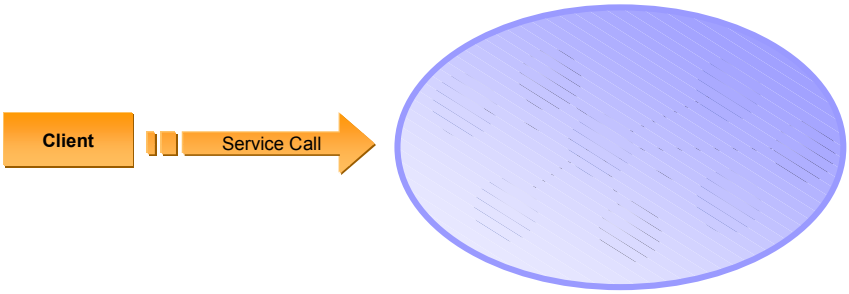
Orientation in Objects

- Grundlagen
- Clustering von J2EE Web Komponenten
  - Implementierungsansätze
  - Apache und Tomcat
- Clustering im Application Server
  - Implementierungsansätze
  - JNDI
  - Enterprise JavaBeans
  - Clustering mit JBoss
- Night School um 21.00 Uhr → Clustering Demo

Clustering in verteilten Systemen

Orientation in Objects

- *A distributed system is a collection of independent computers that appears to the user of the system as a single computer.* (*A. Tanenbaum*)



Ziele des Einsatzes von Clustern

Orientation in Objects

- Hochverfügbarkeit → *High availability*
  - Maximierung der Verfügbarkeit eines Services durch ausweichen auf Alternativkomponenten bei Ausfall einer Komponente
  - Die im Cluster zu nutzenden Komponenten müssen in verschiedenen gleichartigen Umfeldern zur Verfügung stehen.
- Skalierbarkeit → *Scalability*
  - Die Fähigkeit, durch dynamisches Hinzufügen von Komponenten transparent für die Anwender auf Änderungen im Kapazitätsbedarf zu reagieren.
- Grundprinzip: Redundantes Vorhalten sensibler Komponenten

Wege zur Zielerreichung

Orientation in Objects

- A distributed system is one on which I cannot get any work done because some machine I have never heard of has crashed.
  (*Leslie Lamport*).
- Fehlertoleranz → *Failover*
  - Eine Komponente springt bei Ausfall einer anderen automatisch für diese ein. Hierzu wird für die rechtzeitige Aktivierung einer neuen Komponente und die Übergabe des letzen Status der ausgefallenen Sorge getragen.
- Lastverteilung → *Load balancing*
  - Sorgt für eine gleichmässige Verteilung der Kommunikation- und Rechenlast über mehrfach vorhandene Komponenten eines Gesamtsystems.

Fehlertoleranz

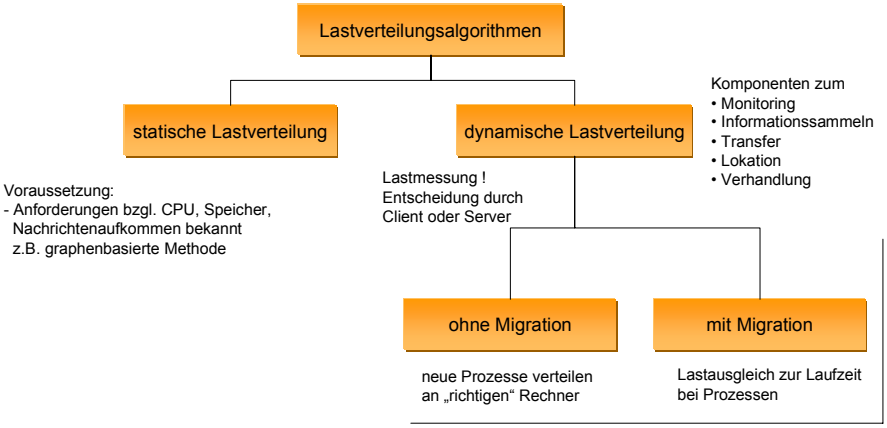
Orientation in Objects

- Ausgleich des Fehlverhaltens einer Komponente durch das gesamte System
  - Unterscheidung zwischen Fehlverhalten, Fehler und Fehlerursache
  - Fehlersemantik:
    - **Serverabsturz**
    - **Dienstverweigerung**
    - **Zeitfehler**
    - **Byzantinischer Fehler**
      - gegenseitiger Informationsaustausch
- Schlüsselverfahren: Redundanz
  - passiv oder aktiv

Lastverteilung

Orientation in Objects

- Wo herrscht Last vor ? Wo sind freie Kapazitäten ?
  - Verteilung von Rechenlast



Überblick

Orientation in Objects

- Grundlagen
- Clustering von J2EE Web Komponenten
  - Implementierungsansätze
  - Apache und Tomcat
- Clustering im Application Server
  - Implementierungsansätze
  - JNDI
  - Enterprise JavaBeans
  - Clustering mit JBoss

Clustering von J2EE Web Komponenten

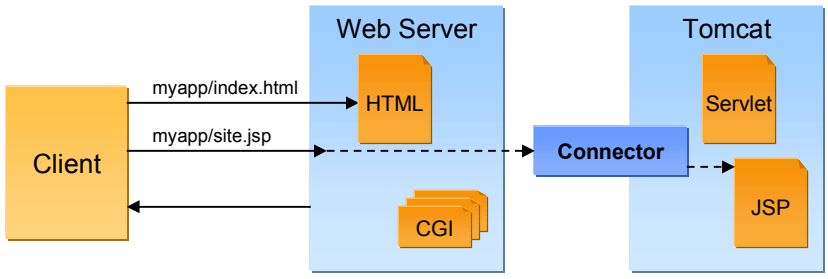
Orientation in Objects

- Anforderungen
  - Server Lastverteilung:
    - **dynamische Verfahren**
      - Round Robin, Auswertung von Connections, Bandbreite, Response Time …
    - **persistente Verfahren**
      - Source-Destination, zeitabhängige IP Zuweisung, Cookies …
    - **Content Balancing**
      - URL/HTTP-Header
- Charakteristika Webschicht:
  - Zustandslosigkeit von Servlets und JavaServer Pages
  - Zustand (zumeist) über Session, Cookies, URL
- Vorstellung:
  - Kombination von Apache und Tomcat
  - Session Replication in Tomcat und JBoss als Bundle

Web Server und Tomcat

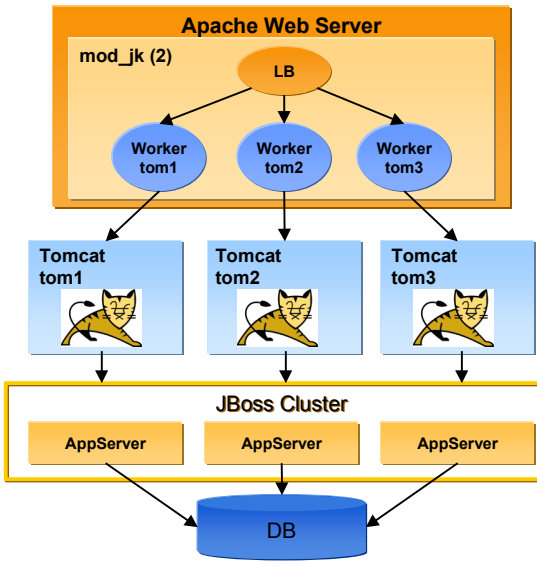
Orientation in Objects

- Tomcat Fähigkeiten für Einsatz als Web Server gegeben
  - Primär: Servlet Container mit JSP Compiler
- Sinnvoll - „Best in breed“
  - Web Server (z.B. Apache, IIS) „erprobter“ in Punkten von Sicherheit, Performance und mit erweiterter Funktionalität



Szenarien: Apache Loadbalancer I

Orientation in Objects



Coyote Connector JK 2 und Worker

Orientation in Objects

- Standard Connector in Tomcat 4.1
- Protokoll: AJP/1.3
  - teilweise binär mit Kompression von häufig verwendeten Strings
  - Wiederverwendung von Sockets
- Apache Modul: mod_jk (1.3 und 2)
  - Plattformunterstützung über JK library
  - Redirektoren für IIS, iPlanet, Domino
  - Unterstützt Tomcat 3.2, 3.3, 4.X, 5.X
- Worker
  - Tomcat, der für den Apache Servlet Requests verarbeitet
  - Werden über workers.properties konfiguriert
    - **Zuordnung von Load Balancern, Prioritäten usw.**

Load Balancing Konfiguration

Orientation in Objects

- Apache mit Tomcat über JK/JK2
  - *httpd.conf* Datei mit Eintrag
    - **LoadModule jk(2)_module modules/mod_jk(2)-2.0.43.dll bzw. .so**
    - **mod_jk(2)_2.0.43 nach *$APACHE_HOME\modules***
  - CoyoteConnector für AJP1.3 im Tomcat freischalten
    - **in *$CATALINA_HOME\conf\server.xml***
  - *workers(2).properties* in *$APACHE_HOME* \conf
    - **Tomcat Instanzen für den Apache**
    - **Load Balancing festlegen**

Load Balancing: Tomcat Standalone

Orientation in Objects

- Tomcat 5.x mit „balancer“ Applikation
  - regelbasiertes Balancing
  - Servlet 2.3 mit Single Filter und URL Mapping
  - Erweiterung des `org.apache.webapp.balancer.Rule` Interface

```
<rules>
    <rule className="org.apache.webapp.balancer.rules.URLStringMatchRule"
     targetString=„Java" redirectUrl="http://www.oio.de"/>
    …
</rules>
```

- ***Beispiel***: URLStringMatchRule

```
public boolean matches(HttpServletRequest request) {
      String requestUrl = request.getRequestURL().toString();
        …
}
```

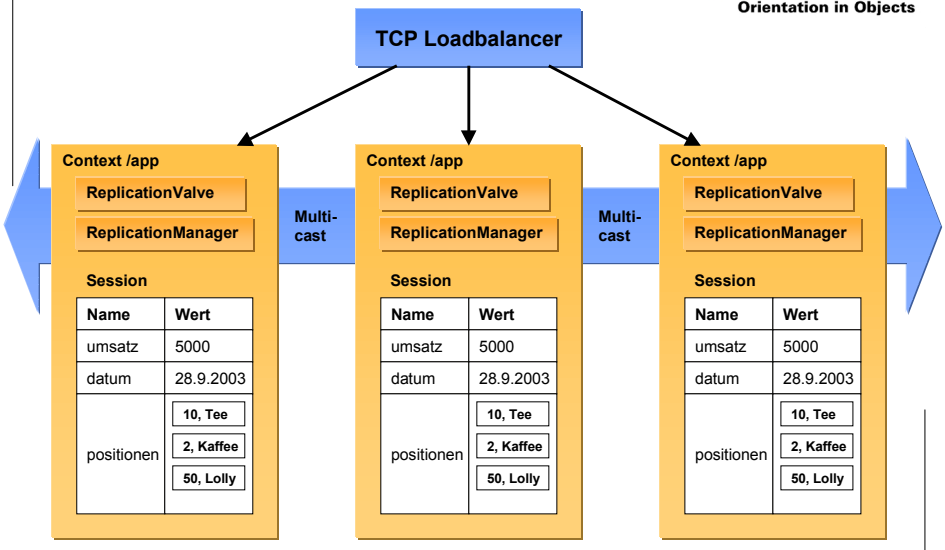
Session Replication in Tomcat 5

Orientation in Objects

- Clustering als Standard Service in Tomcat 5
  - ImMemory Session Replication Mechanismus
  - Session mit serialisierbaren Objekten replizierbar
  - verschiedene Replicationsebenen
    - **Attribut: DirtyFlag**
  - „Aufrüsten“ von Tomcat 4 möglich
    - **bestehende Persistenzmöglichkeiten : Dateisystem, Datenbank**
- Möglichkeiten:
  - Replication Modi: pooled, synchronous, asynchronous

Session Replication mit Tomcat

Orientation in Objects



Vorsicht bei DirtyFlag und komplexen Typen in der Session

JBoss zur In-Memory Session Replication

Orientation in Objects

- Tomcat Session Replication Aktionen:
  - Create, Hinzufügen und Entfernen von Attributen (serialisierbar)
  - Expiration time, Timestamp update, User Principal
- Problematik:

```
Object o = httpSession.getAttribute ("table");
java.util.Hashtable table = (java.util.Hashtable)o;
table.put („something", „changedValue");
```

- Http Session Replication = JBoss Service
  - Snapshot Konfiguration
    - **instant – direkt beim Erzeugen**
    - **intervall – zeitliche Abhängigkeiten**

JBoss HTTP Session Replication

Orientation in Objects

- JBoss Konfiguration
  - Service in „all" Konfiguration
  - jboss-web-xml

```
<jboss-web>
   ...
   <replication-config>
        <replication-trigger>SET_AND_NON_PRIMITIVE_GET
        </replication-trigger>
        <replication-type>SYNC</replication-type>
   </replication-config> ...
</jboss-web>
```

- Tomcat:
  - <Engine> mit ***jvmRoute*** Attribut
- Applikation:
  - web.xml mit ***<distibutable/>*** Tag

Überblick

Orientation in Objects

- Grundlagen
- Clustering von J2EE Web Komponenten
  - Implementierungsansätze
  - Apache und Tomcat
- Clustering im Application Server
  - Implementierungsansätze
  - JNDI
  - Enterprise JavaBeans
  - Clustering mit JBoss

Ansatzpunkte

Orientation in Objects

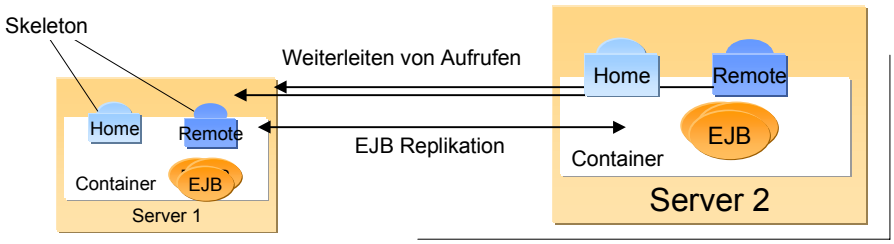
- EJB Container
- Home stub
- Remote stub
- JNDI (Java Naming Directory Interface) Naming Server

- Charakteristika:
  - Diese Stellen besitzen Logik und Möglichkeiten, die zum Clustern von EJB benutzt bzw. erweitert werden kann.

EJB Container Strategien

Orientation in Objects

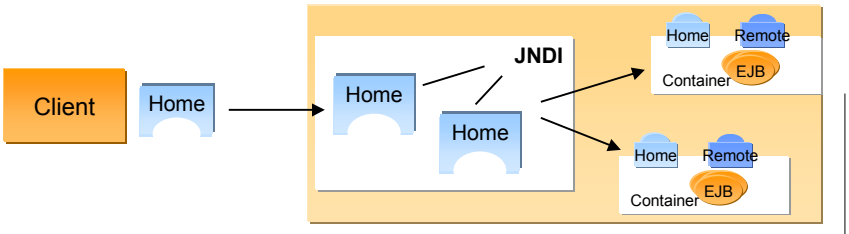
- EJB Container
  - Call bereits direkt an dem Punkt, wo auf einen Zustand ausweichend reagiert muss.
- Load Balancing
  - Weiterleiten von Calls bei vollem Cache
- Fail over
  - State Replication



Clustering auf JNDI Basis

Orientation in Objects

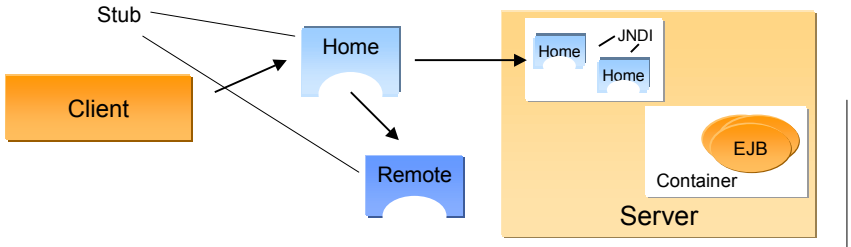
- JNDI Naming Server
  - Ansatzpunkt als „Zwischenschicht“
  - Abhängigkeit von Applikation Server und Implementierung
- einheitliche Schnittstelle zu Namens- und Verzeichnisdiensten
- Basisdienst von J2EE basierten Applikationsservern



Home und Remote Stub

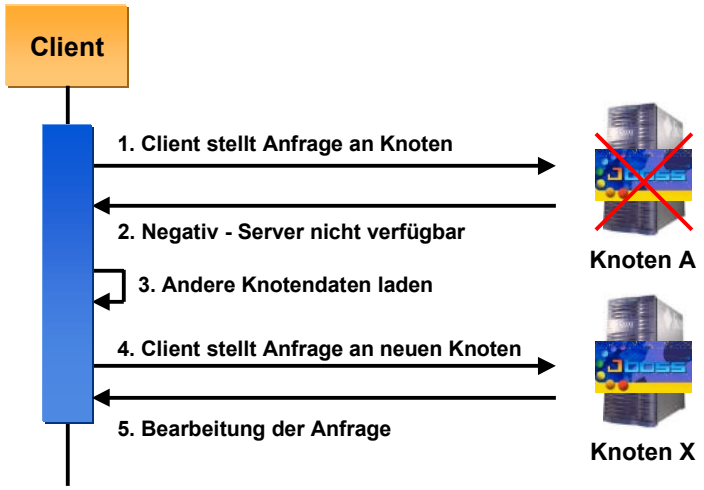
Orientation in Objects

- Home und Remote Aufrufe zu unterschiedlichen Zeitpunkten - verschiedene Strategien möglich
  - Logik muss bereits im Code implementiert sein
  - Unterschiedliche Betrachtung bei verschiedenen EJB Arten



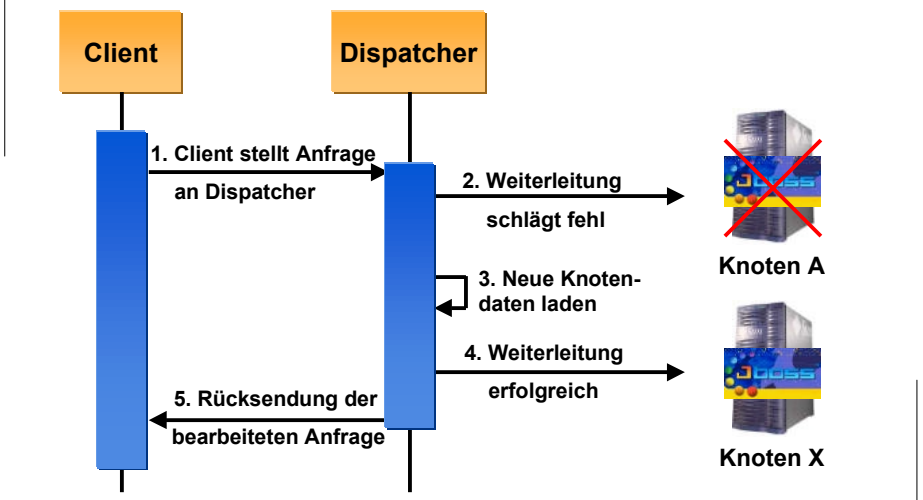
Failover - Client managed

Orientation in Objects



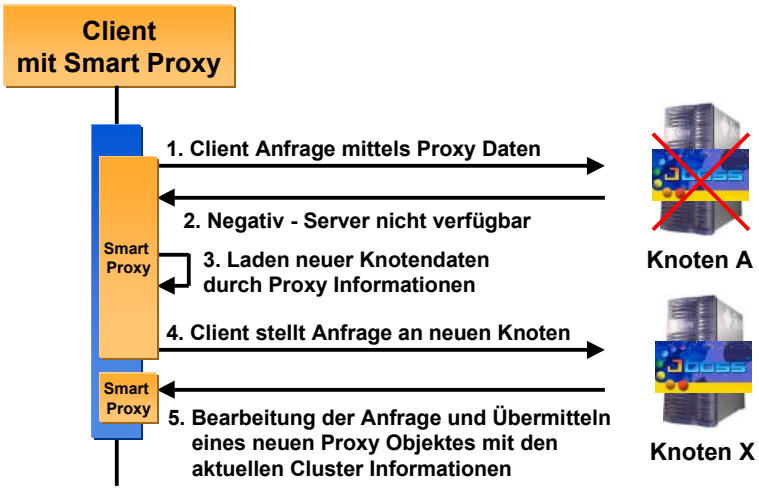
Failover - Dispatcher managed

Orientation in Objects



Failover mit dynamischem Proxy

Orientation in Objects



Idempotenz Problem

Orientation in Objects

- Konzept:
  - Jeder Methodenaufruf kann wiederholbar ausgeführt werden, ohne das sich am Verhalten oder Zustand der Applikation etwas ändert.
- transaktionale Methode - nicht automatisch idempotent !
- nie Datenmanipulation
- Verantwortung beim Entwickler und Container Vendor
  - Exceptions
  - Deklaration

JBoss Clustering Solutions

Orientation in Objects

- Verwendung von Smart Proxies
  - zur Laufzeit geladene Stubs Objekte für Remotereferenzen
  - informiert Client über verfügbare Nodes
  - enthält Code für Failover und Load-Balancing
  - für Client transparent
- Load-Balancing Algorithmen sind pluggable

JBoss Clustering - Wichtige Features

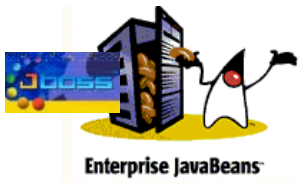
Orientation in Objects

- Automatisches Erkennen
  - Knoten in einem Cluster finden sich selbst ohne zusätzliche Konfiguration.
- Fail-over und Load balancing für:
  - JNDI, RMI, Entity Beans, Stateful / Stateless Session Beans
- Dynamisches JNDI Erkennen
  - JNDI Clients können automatisch den JNDI InitialContext erkennen.
- HTTP Session Reproduktion
  - für Tomcat und Jetty über JBoss

Cluster einrichten

Orientation in Objects

- JBoss Server im Cluster
  - starten in der „all“ - Konfiguration bei allen JBoss Servern des Clusters
    - **[JBOSS_HOME]\bin\run -c all**
- Enterprise Java Beans
  - die Beans „cluster-fähig“ machen
  - auf allen Knoten des Clusters deployen in das Verzeichnis [JBOSS_HOME]\server\all\deploy
- Client - JNDI Properties Konfiguration
  - java.naming.provider.url



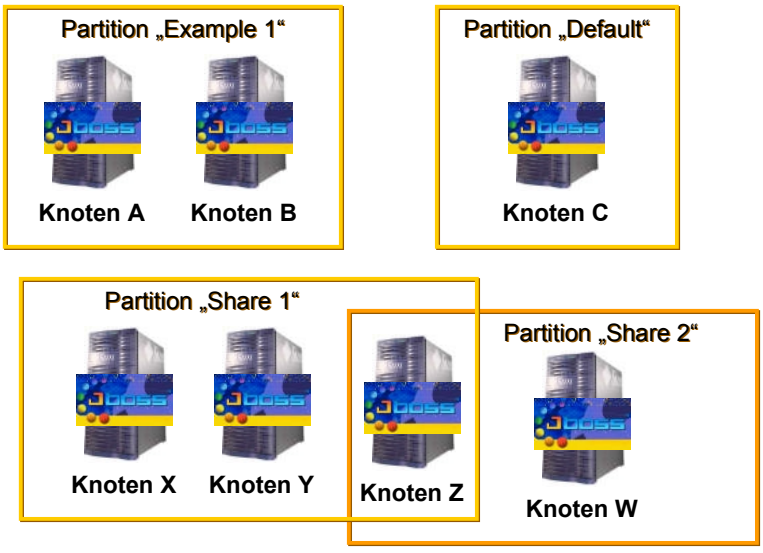
JBoss Sprechweise

Orientation in Objects

- Partition (äquivalent zu einem Cluster)
  - Gruppierung von JBoss Instanzen zum Erstellen des Clusters
  - eine JBoss Instanz kann mehreren Partitionen angehören
- Sub-Partitionen
  - werden momentan von JBoss noch nicht unterstützt
  - interessant für strategische Lastverteilung
    - **Beispiel: Stateful Session**
  - Fehlertoleranz bspw. durch Quorum

JBoss Partitionen

Orientation in Objects



JBoss Cluster Konfiguration

Orientation in Objects

- Clustering nur in der „***all***“ Konfiguration möglich
- im Verzeichnis *.../all/server/deploy* befindet sich die *cluster-service.xml*

```
<mbean code="org.jboss.ha.framework.server.ClusterPartition"
   name="jboss:service=DefaultPartition"/>
```

- Einstellungen für
  - Partitionierung
  - Naming Service
  - SessionState
  - Cache

„Clusterfähigkeit“ der EJB

Orientation in Objects

- <clustered> Tag in der *jboss.xml*

```
<jboss>
    <enterprise-beans>
          <session>
            <ejb-name>SimpleWorker</ejb-name>
            <jndi-name>SimpleWorker</jndi-name>
            <clustered>True</clustered>
          </session>
     </enterprise-beans>
    ...
</jboss>
```

JNDI Implementierungen

Orientation in Objects

- JNDI Clustering Möglichkeit - Abhängig von der JNDI-Implementierung des Applikationsservers
- Varianten:
  - Unabhängiger JNDI- Baum
  - Zentraler JNDI- Baum
  - Shared global JNDI- Baum

Unabhängiger JNDI- Baum

Orientation in Objects

- lokaler JNDI Baum ohne Kenntnis der anderen Server im Cluster
- Vorteil:
  - einfach zu skalieren (Hinzufügen weiterer Server)
  - Zeitaufwand sehr kurz, alle Clusterkomponenten beim Hochfahren der Server des Clusters zu erkennen = *Konvergenzzeit*
- Nachteil:
  - keine interne Umsetzungsstrategien für Ausfallsicherheit
    - **extern über Dispatcher Entwurf, Proxy Dienst möglich**

Zentraler JNDI-Baum

Orientation in Objects

- Cluster mit einem zentralem JNDI-Baum, an den alle Objekte gebunden werden
- Name Server kennt diesen JDNI-Baum, d.h. ein Client erhält „globale“ Referenz, mit der er beim einem Server im Cluster Home- und Remote erhält

- Nachteil:
  - Ausfallsicherheit vs. Name Server down !
    - **Einbinden neuer Name Server bedingt neues Ermitteln und Binden aller Daten = hoher zeitlicher Aufwand**
  - Verdopplung der Anzahl der JNDI-Aufrufe
  - Vergrößern des Clusters erhöht die Konvergenzzeit und erfordert evtl. das Einbeziehen mehrerer Name Server

Shared Global JNDI- Baum I

Orientation in Objects

- Verwenden von IP Multicasting
  - Bekanntgabe neuer Server im Cluster - Übertragen des JNDI Baumes

- Objekte werden im shared global JNDI- Baum des Clusters und lokalen JNDI- Baum des Servers gebunden.
  - Failover und schneller JNDI Lookup

- spezielle Home Objekte im shared global JNDI-Baum, wenn ein Objekt auf mehr als einem Server im Cluster verfügbar ist
  - Kenntnis über alle verfügbaren Objekte

Shared Global JNDI- Baum II

Orientation in Objects

- Vorteile
  - einfachere Skalierbarkeit sowie höhere Verfügbarkeit als im zentralen Cluster
  - Netzwerkbelastung im Betrieb geringer als im zentralen Cluster - pro Lookup nur einen Netzwerkaufruf
- Nachteile
  - hohe Netzwerkbelastung beim Hochfahren der Server -höhere Konvergenzzeit , abhängig von der Anzahl der Server des Clusters
- Shared Global JNDI teilweise in JBoss

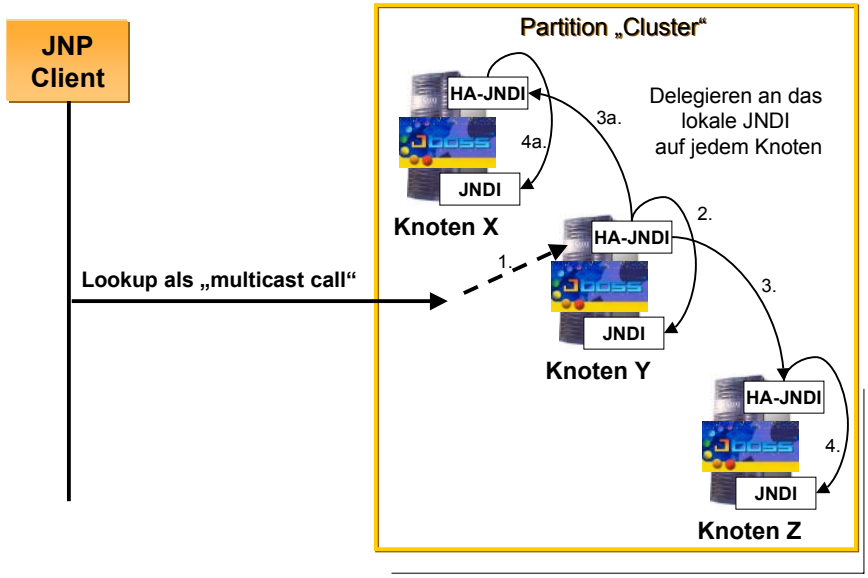
HA-JNDI

Orientation in Objects

- HA-JNDI
  - ein cluster-weiter, globaler JNDI Context zum Binden und Abfragen von Objekten
  - durch Replikationsmechanismus sind Objekte, die an diesen Context gebunden sind, auch beim Ausfall eines Knotens erreichbar
  - serverseitiges Binden von Objekten wird über „normales“ JNDI nur lokal ausgeführt, d.h. ein entfernter Client Aufruf wird vom HA-JNDI an das lokale JNDI delegiert
  - Konfiguration möglich in der *cluster-service.xml*

HA-JNDI Regeln zum Lookup und Binden

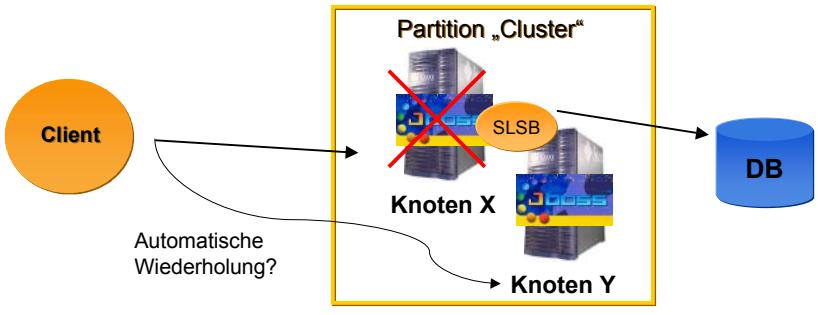
Orientation in Objects



Stateless Session Beans - Fail over

Orientation in Objects

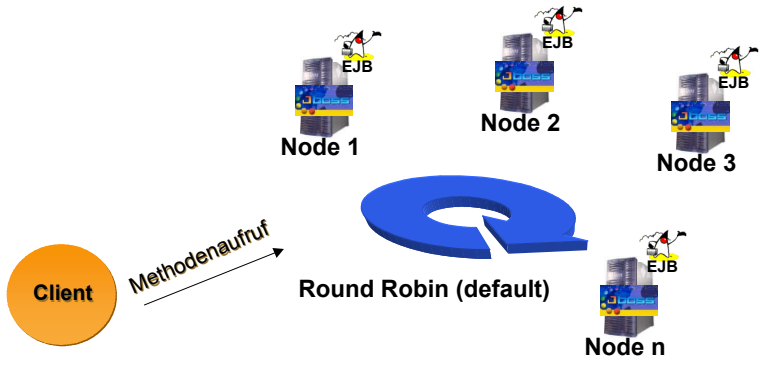
- Home Stub: trivial
- Remote Stub:
  - Idempotenz Eigenschaft → Entwicklerverantwortung !
  - no state - no problem - just try again ! (JBoss)



Stateless Session Beans - Load Balancing

Orientation in Objects

- Home und Remote per Default - Round Robin
  - andere Strategien möglich
- Konfiguration in der *jboss.xml*



Balancing Strategien

Orientation in Objects

- Konfiguration für Home und Remote Stub
  - Round Robin
    - **jeder Call wird einem neuen Knoten im Cluster zugewiesen**
  - FirstAvailable
    - **verfügbarer Stub sendet Calls immer auf einen Knoten**
  - FirstAvailableIdenticalAllProxies
    - **wie FirstAvailable, der Zielknoten wird von den Proxies der gleichen „family“ geteilt (ab 3.2 und höher)**
- Strategien erweiterbar
  - Laufzeitinformationen der VM
    - **verfügbarer Speicher**
    - **Anzahl der Threads, die gerade warten/laufen**

Balancing Konfiguration *optional*

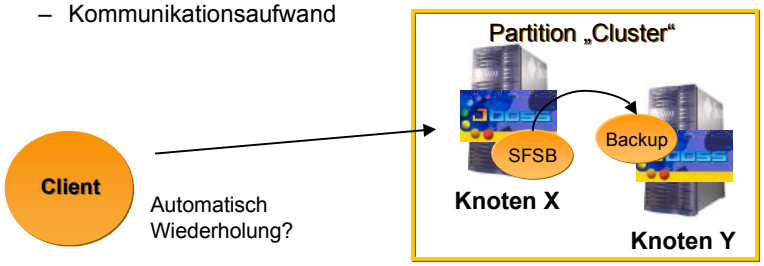
Orientation in Objects

```
jboss>
    <enterprise-beans>
      <session>
          <ejb-name>SimpleWorker</ejb-name>
          <jndi-name>SimpleWorker</jndi-name>
          <clustered>True</clustered>
          <cluster-config>
                <partition-name>DefaultPartition</partition-name>
                <home-load-balance-policy>
                      org.jboss.ha.framework.interfaces.RoundRobin
                </home-load-balance-policy>
                <bean-load-balance-policy>
                      org.jboss.ha.framework.interfaces.RoundRobin
                </bean-load-balance-policy>
          </cluster-config>
      </session>
    </enterprise-beans>
    <enforce-ejb-restrictions>false</enforce-ejb-restrictions>
</jboss>
```


Stateful Session Beans - Fail over I

Orientation in Objects

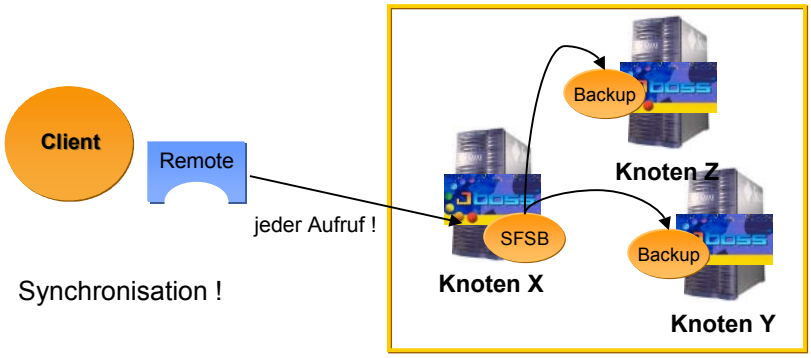
- Home Stub und Remote Stub: Idempotenz !
  - Home: create - nicht idempotent !
  - Remote
    - **nur getter, keine Zustandsveränderung herbeiführend**
- Fail over über Replikation / Back up
  - Kommunikationsaufwand



Stateful Session Beans - Load Balancing

Orientation in Objects

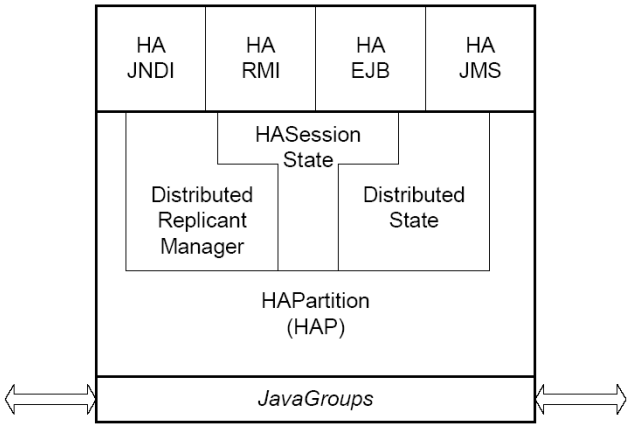
- Konfiguration wie bei Stateless Session Bean
  - Home stub: Round Robin
  - Remote stub: FirstAvailable
    - **„pinned“**



- Synchronisation !

JBoss Clustering Architecture

Orientation in Objects



vgl. JBoss Docs

JBossClustering - JavaGroups

Orientation in Objects

- JavaGroups – Open-source group communication Toolkit
  - Verbindungslose Übertragung von Nachrichten über Multicast
    - **Paketgröße, Paketverlust, Paketreihenfolge**
  - Atomicity: all or nothing
  - Gruppenmitgliedschaft
  - Benachrichtigung über Gruppenveränderung
  - Ablaufkontrolle
  - Fehlererkennung
  - Zustandsreplikation verschiedener Transportprotokolle

Entity Beans - Fail over

Orientation in Objects

- Remember: Idempotenz !
  - idempotent: getter, finder, evtl. home business Methoden
- keine Notwendigkeit von Replikation bei Einsatz einer Datenbank
  - Caching !
  - Fail over nur am Ende einer Transaktion
- Local Interfaces
- Session Facade

Caching und Synchronization

Orientation in Objects

- Cache
  - vor der Datenbank - Synchronisation
  - *distributed shared object cache*
    - **Cache und DB synchron und umgekehrt**
- Kommunikations- und Verwaltungsaufwand

Commit Time Options (Container Option)

Orientation in Objects

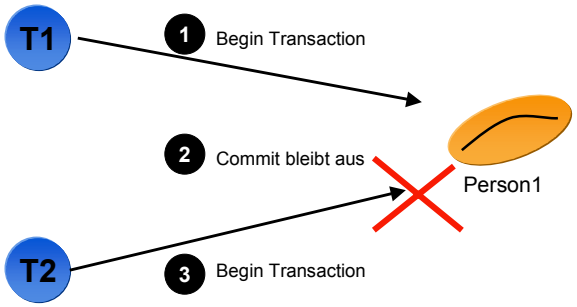
	Instanz Status in DB schreiben	Instanz bleibt im „ready“ Zustand	Instanz-Zustand wird ungültig (vor Transaktion Synchronisation)
Option A	Ja	Ja	Nein
Option B	Ja	Ja	Ja
Option C	Ja	Nein	Ja
Option D	Ja	Ja	Nein

Periodisches Update!

Entity Locking Problem

Orientation in Objects

- Optimistic vs. pessimistic locking
- Verantwortung bei Datenbank im JBoss
  - Konfiguration in jbossjdbc-cmp.xml - Tag ***<row-locking>***

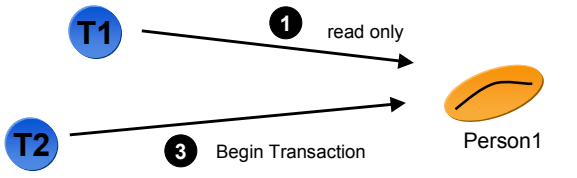


Strategie: Read-only

Orientation in Objects

- Konfiguration in jboss.xml

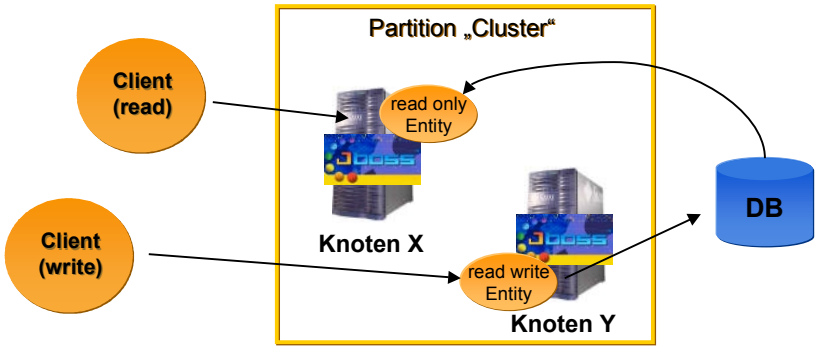
```
...
<entity>
    <ejb-name>Person</ejb-name>
    ...
    <read-only>true</read-only>
</entity>
...
```



Read only cache

Orientation in Objects

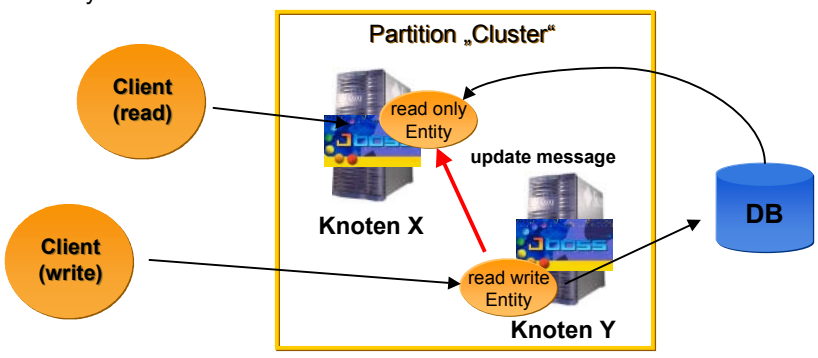
- Update nach Commit Option
- Locking umgangen
  - Wirksam: Cluster und Standalone



Read mostly cache

Orientation in Objects

- mögliche Lösung für „stale data“ Problem
- Zwei Kommunikationswege
  - synchron / multicast
  - asynchron / JMS

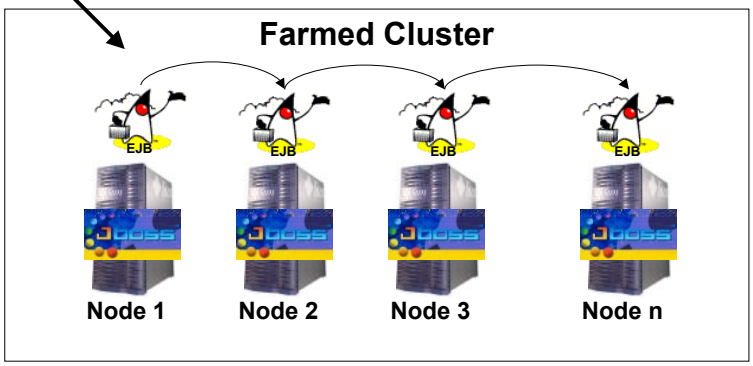


What is farming ?

Orientation in Objects

- „hot deploy“ im gesamten Cluster möglich

deploy *.ear



Orientation in Objects

Vielen Dank für Ihre Aufmerksamkeit

Orientation in Objects GmbH
Weinheimer Str. 68
68309 Mannheim
www.oio.de