akquinet

AS 7 / EAP 6 - Clustering

heinz.wilming@akquinet.de
@akquinet
http://blog.akquinet.de

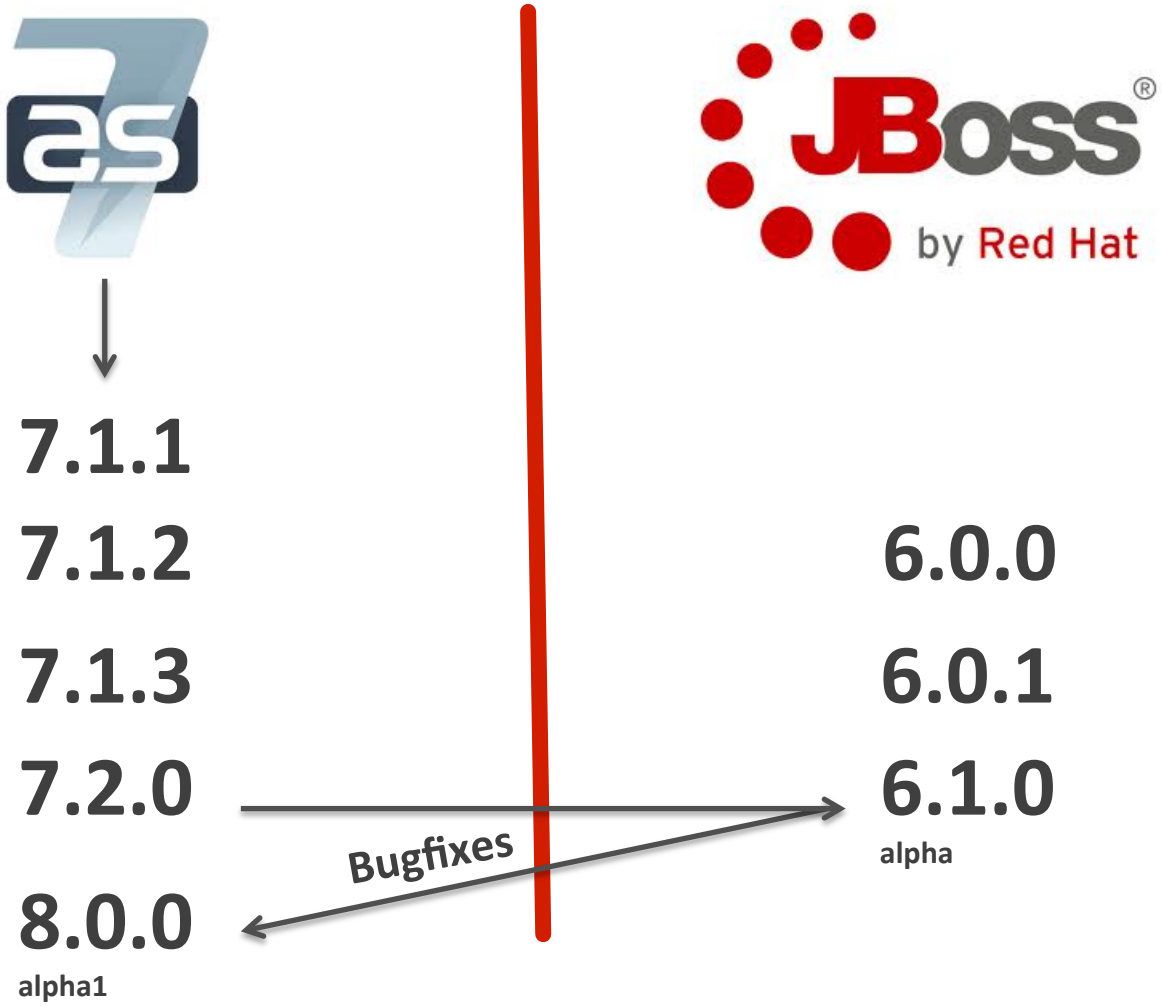
Was ist die EAP 6?

EAP6 != EAP5 +1

JBoss Enterprise Application Platform 6

- Stabile und unterstützte Plattform
- Basiert auf JBoss Application Server 7.1.3
- Implementiert die Java Enterprise Edition 6 Spezifikation
  - Web Profile
  - Full Profile
- EAP6 ist für beide Profile zertifiziert

Enterprise Application Platform 6



EAP6 / JBoss AS 7

akquinet

Ein komplett neuer Kernel

- Einsatz von JBoss **Modules** und JBoss **Modular Service Container (MSC)**
- Sehr schnelle Startzeiten
- Geringer Speicherverbrauch
- Bessere Trennung von Diensten
  (z.B. zwei unterschiedliche Nachrichtendienste in einer JVM möglich)
- Einfache Verwaltung und Konfiguration

akquinet

Domain vs. Standalone
Konfiguration und Verwaltung

Verwaltungskonzept

Domain Mode

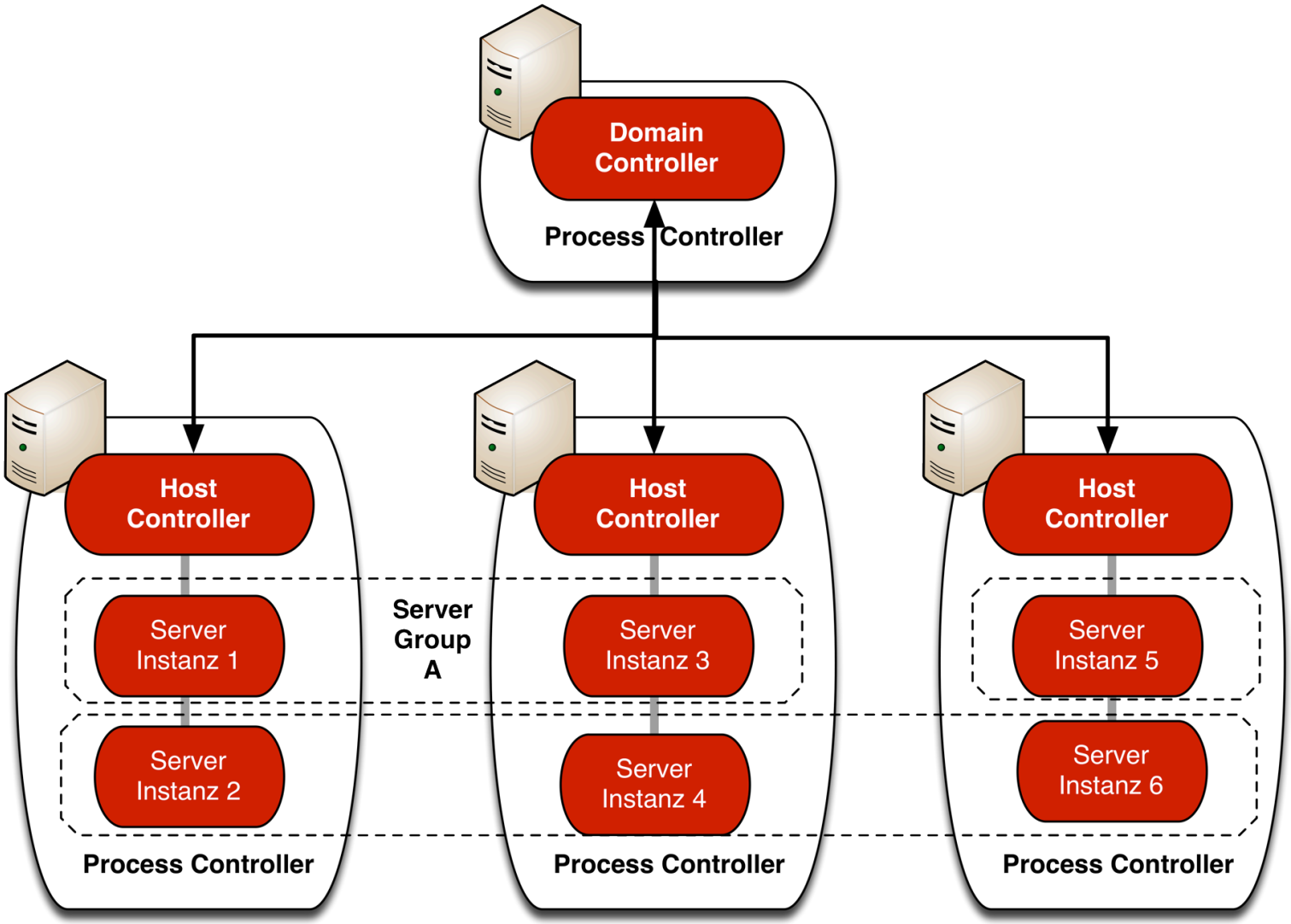
- zentrales Management mehrerer Serverinstanzen und physikalischer Hosts
- Zentraler Domain-Controller
- Verwaltung mehrerer Server (Hosts)
- Zentrale Konfiguration
- Zentrales Deployment via Mangementtools, z.B. CLI

Standalone Mode

- Verwaltung genau einer Serverinstanz
- Verhalten wie JBoss 4/5/6
- Eine Server-Instanz pro JVM
- Kein Farm-Deployment
- Deployment via Filesystem und CLI

⇒ *Unabhängig von den Serverfunktionen und Eigenschaften*
⇒ *Unabhängig von HA, nur ein Verwaltungskonzept*

Host und Domain Controller



EAP 6 Clustering

Cluster = Gruppe von Servern, die gleiche Dienste erbringen
Client sieht nur den Cluster nicht die Server

Ziele:

- Fehlertoleranz durch Ausblenden fehlerhafter Server
  (⇒ Hochverfügbarkeit; High Availability, HA)

- Lastausgleich durch Verteilung an Server Knoten
  (⇒ Skalierbarkeit, Performanz)

EAP 6 - Konfigurationsprofile

- **default**
  Java EE Web Profile + Erweiterungen
  wie z.B.: JAX-RS, EJB Remote Invocation
- **ha**
  default + Clustering Funktionen
- **full**
  Java EE Full Profile
  + Alle Serverfunktionen ohne HA
- **full-ha**
  Full Profile + Clustering

Clustering Services (Channel, Cache) werden nur gestartet wenn eine abhängiger Service Cluster-Funktionalitäten benötigt.

```
<domain>
  <extensions>
    ...
  </extensions>

  <profiles>
    <profile name="ha">
      <subsystem xmlns="urn:">
        ...
      </subsystem>
      ...
    </profile>

    <profile name="full-ha">
      <subsystem xmlns="urn:">
        ...
      </subsystem>
    </profile>
  </profiles>
 </domain>
```

Cluster–fähige Applikationen

Cluster-fähige Applikationen

akquinet

Clustering einer EJB Session-Bean

```
@Stateless
@Remote(ClusteredStateless.class)
@org.jboss.ejb3.annotation.Clustered
public class ClusteredStatelessBean implements
      ClusteredStateless {
   ...
}
```

Zustandslose Session Beans

→ dynamische Lastverteilung auf Clusterknoten

Zustandsbehaftete Session Beans

→ Session Affinity

→ Failover

Alternativ: Deployment-Descriptor (META-INF/jboss-ejb3.xml)

HTTP Session-Replikation

Replikation der HTTP-Session stellt sicher, dass die Sessions der Klienten auf anderen Cluster-Knoten verfügbar sind

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
      http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
      version="3.0">

    <distributable/>

</web-app>
```

→ **WEB-INF/web.xml**

JPA/Hibernate 2nd Level Cache

Infinispan ist der Standard 2nd Level Cache ab Hibernate 3.5

Aktivierung in der Persistence Unit (`persistence.xml`)

```
<property name="hibernate.cache.use_second_level_cache" value="true" />
```

full- und default-Profil

- Local Cache Container für Entitäten / Collections

ha-Profil

- Invalidation Cache Container für Entitäten / Collections

Überschreiben der Default-Konfiguration

Cache-Container

```
<property name="hibernate.cache.infinispan.cachemanager"
      value="java:jboss/infinispan/mycache"/>
```

Cache Region

```
<property name="hibernate.cache.infinispan.entity.cfg" value="entity"/>
<property name="hibernate.cache.infinispan.collection.cfg" value="entity"/>
<property name="hibernate.cache.infinispan.query.cfg" value="local-query"/>
<property name="hibernate.cache.infinispan.timestamp.cfg" value="timestamp"/>
```

EAP6 – Cluster-relevante Subsysteme

JGroups
Gruppenkommunikation

Infinispan
Verteilter Cache

mod_cluster
HTTP Load-Balancer

EJB Client Library
Remote EJB Invocation

JGroups

JGroups ist ein Framework für Gruppenkommunikation.

Eigenständiges Projekt unter dem Dach von JBoss
http://www.jgroups.org/

JGroups wird seit JBoss AS 3.x für die Gruppenkommunikation verwendet

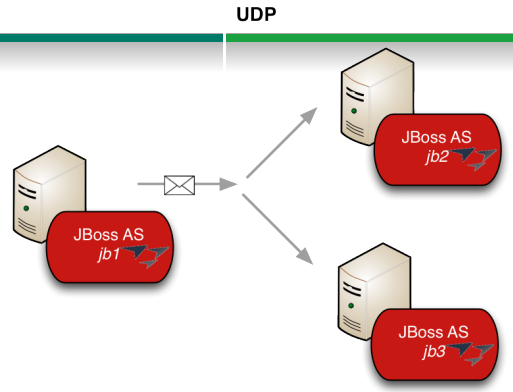
Eigenschaften/Fähigkeiten:

- fehlertolerantes Verschicken von Nachrichten in einer Gruppe von Sendern/ Empfängern;
  sowohl Punkt-zu-Punkt als auch Punkt-zu-Gruppe
- Die Gruppe kann dynamisch vergrößert und verkleinert werden
- Automatische Erkennung von funktionsunfähigen Teilnehmern
- Verschlüsselung und Authentifizierung
- Unterschiedliche Protokollpipelines (JGroup Stack)

JGroups - Transport

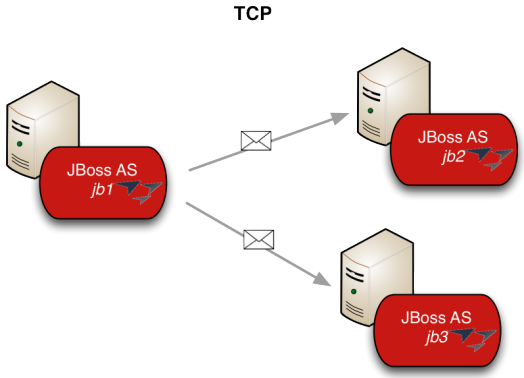
UDP

- UDP-Unicast für Unicast-Kommunikation
- IP-Multicast für Multicast-Kommunikation



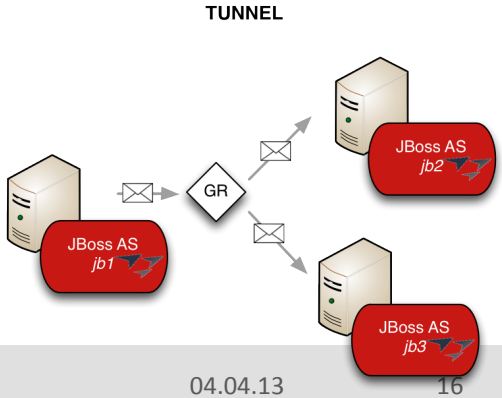
TCP

- TCP-Unicast für Unicast-Kommunikation
- mehrere TCP-Unicasts
  für Multicast-Kommunikation



TUNNEL

- TCP-Unicast (über GossipRouter) für Unicast-Kommunikation
- mehrere TCP-Unicasts (Host nur einer, GossipRouter mehrere)
  für Multicast-Kommunikation



JGroups – JBoss AS 7 – Konfiguration

JGroups Subsystem-Module enthält eine Standard-Konfiguration des JGroups Stacks

```
JBOSS_HOME/modules/org/jboss/as/clustering/jgroups/main/
   jboss-as-clustering-jgroups-7.x.x.Final.jar
→ jgroups-defaults.xml
```

Überschreiben der Standard-Konfiguration im JGroups Subsystem:

```
<subsystem xmlns="urn:jboss:domain:jgroups:1.1" default-
   stack="udp">
    <stack name="udp">
        ...
    </stack>
    <stack name="tcp">
       ...
    </stack>
</subsystem>
```

JGroups - Wahl des Kommunikationsprotokolls

UDP über IP-Multicast

- Parameter: Multicastadresse
- Gruppenmitglieder finden sich ohne weitere Konfiguration selbstständig

Alternativ kann UDP (Unicast) oder TCP benutzt werden, dann müssen die einzelnen IP-Adressen gepflegt werden

```
<protocol type="TCPPING">
   <property name="initial_hosts">
      192.168.0.1[7600], 192.168.0.2[7600]
   </property>
   <property name="num_initial_members">2</property>
   <property name="port_range">0</property>
   <property name="timeout">2000</property>
</protocol>
```

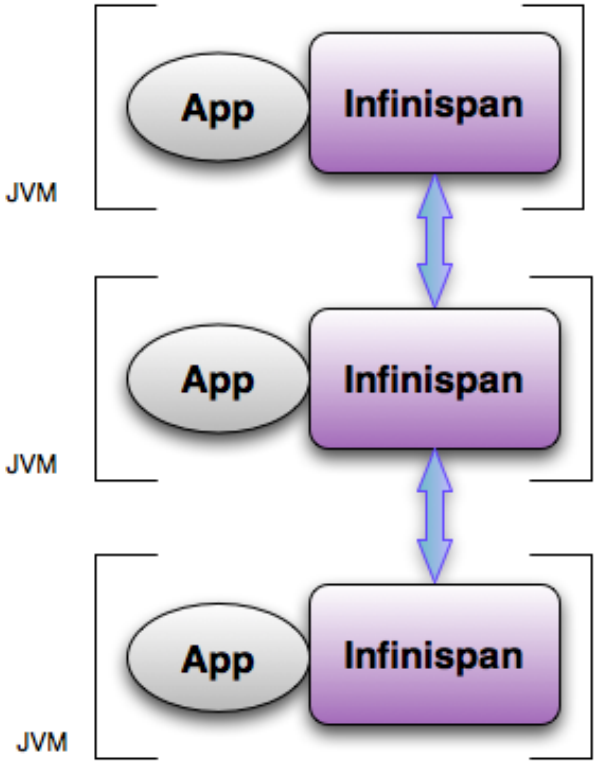
Infinispan

Um Hochverfügbarkeit zu unterstützen, müssen die Daten innerhalb des Clusters repliziert werden

Infinispan wird intern für die Replikation verwendet

Eigenschaften

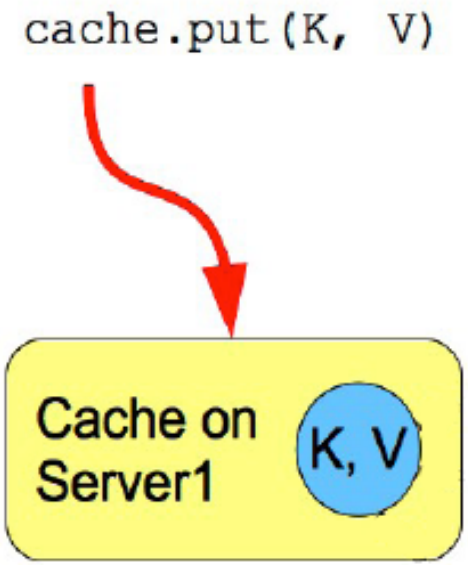
- JSR-107 kompatible Cache Schnittstelle
- Map
- Schlüssel-Wert Paaren



Local - Cache mode

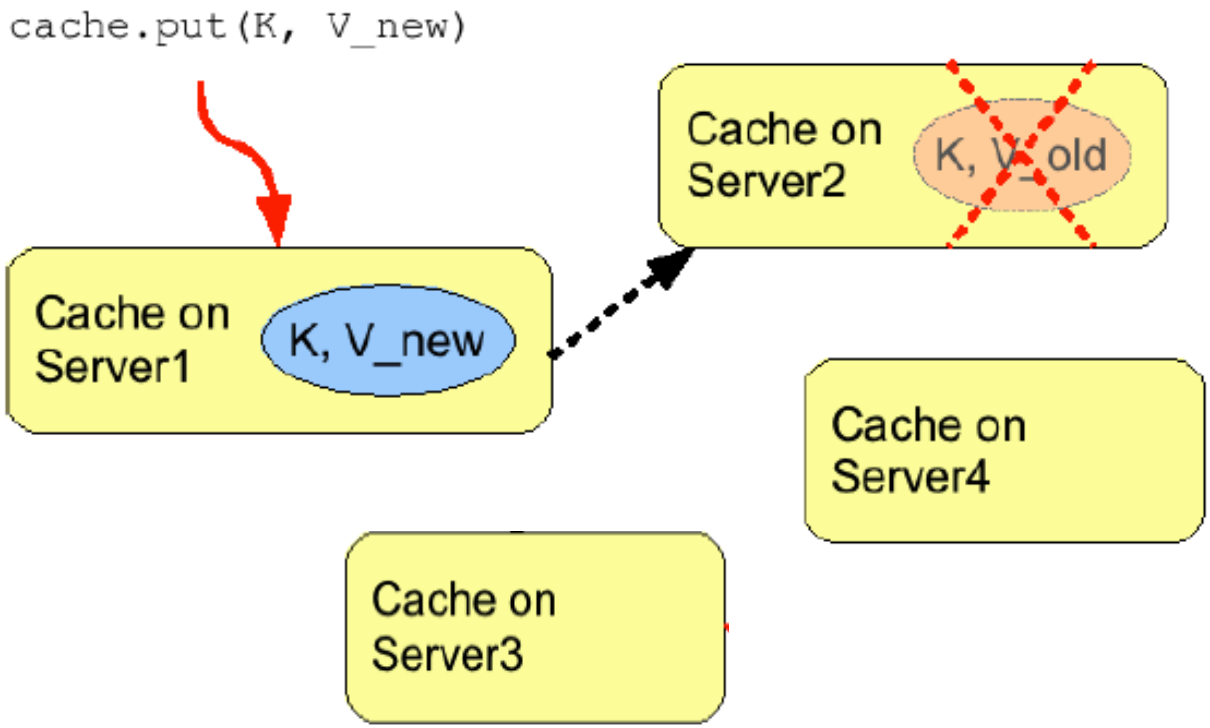
Lokaler In-memory Cache
ähnlich wie JBoss Cache und EHCache

Hibernate 2nd Level Cache

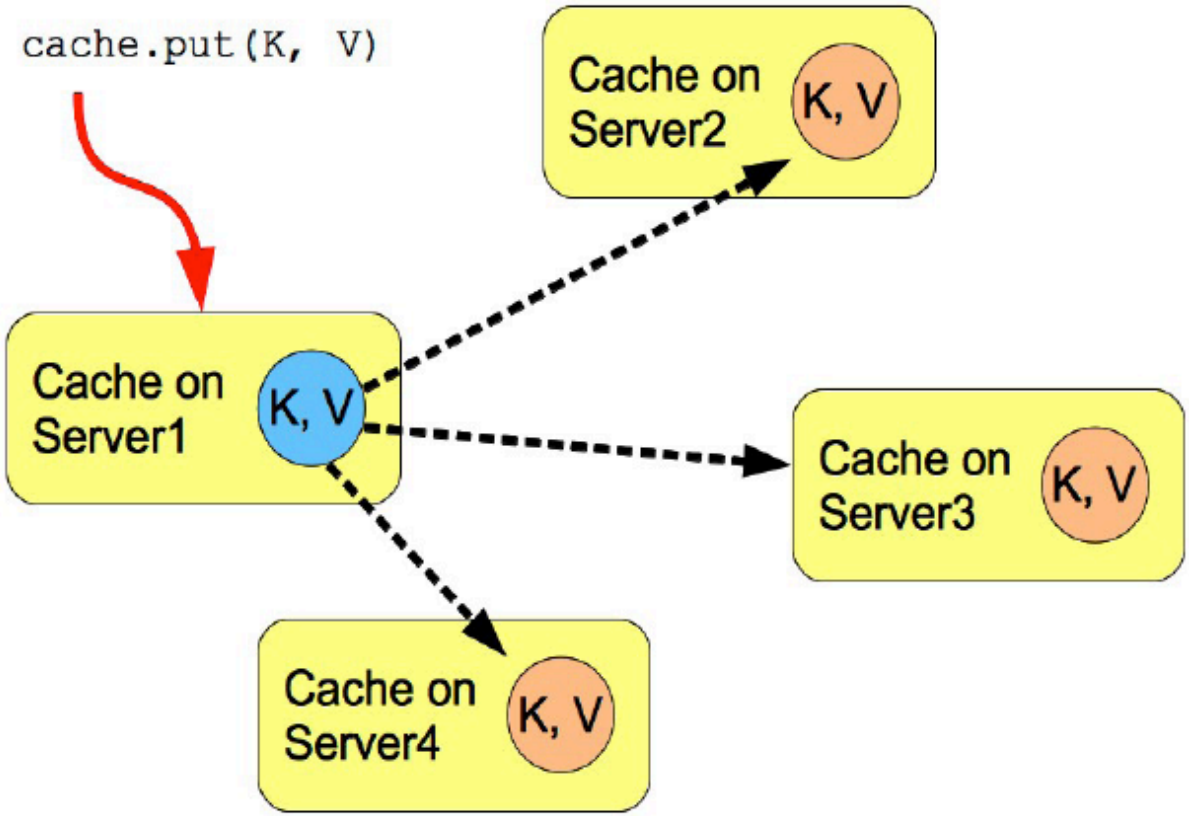


Invalidation - Cache mode

z.B.: Hibernate 2nd Level Cache in einer Cluster Umgebung

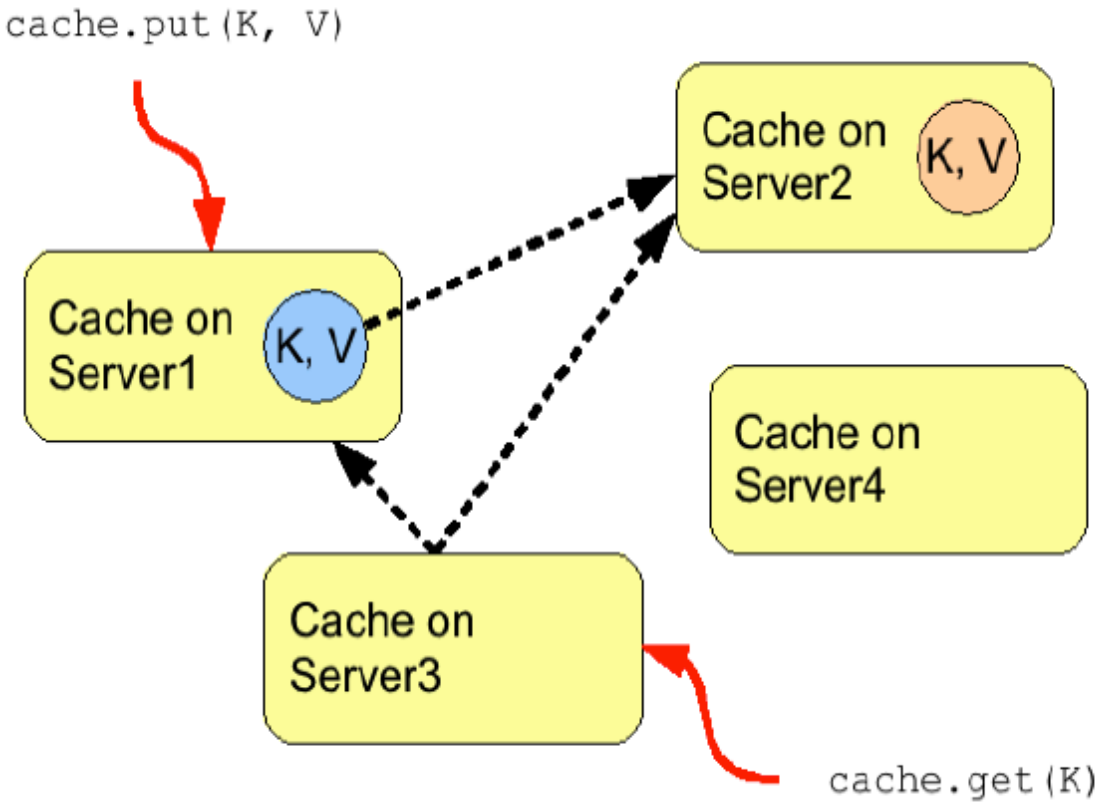


Replication - Cache mode



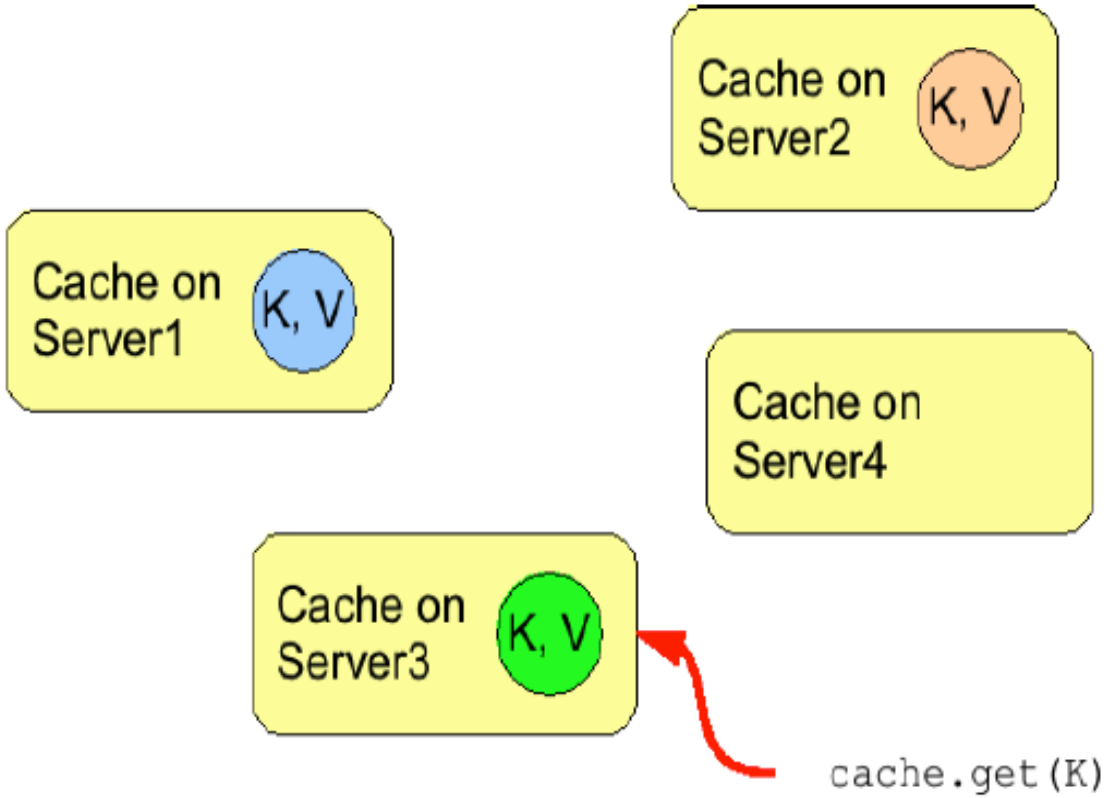
Distribution - Cache mode

Konfigurierbare Anzahl von Kopien (Gute Skalierbarkeit)
Basiert auf Consistent Hash Algorithmus

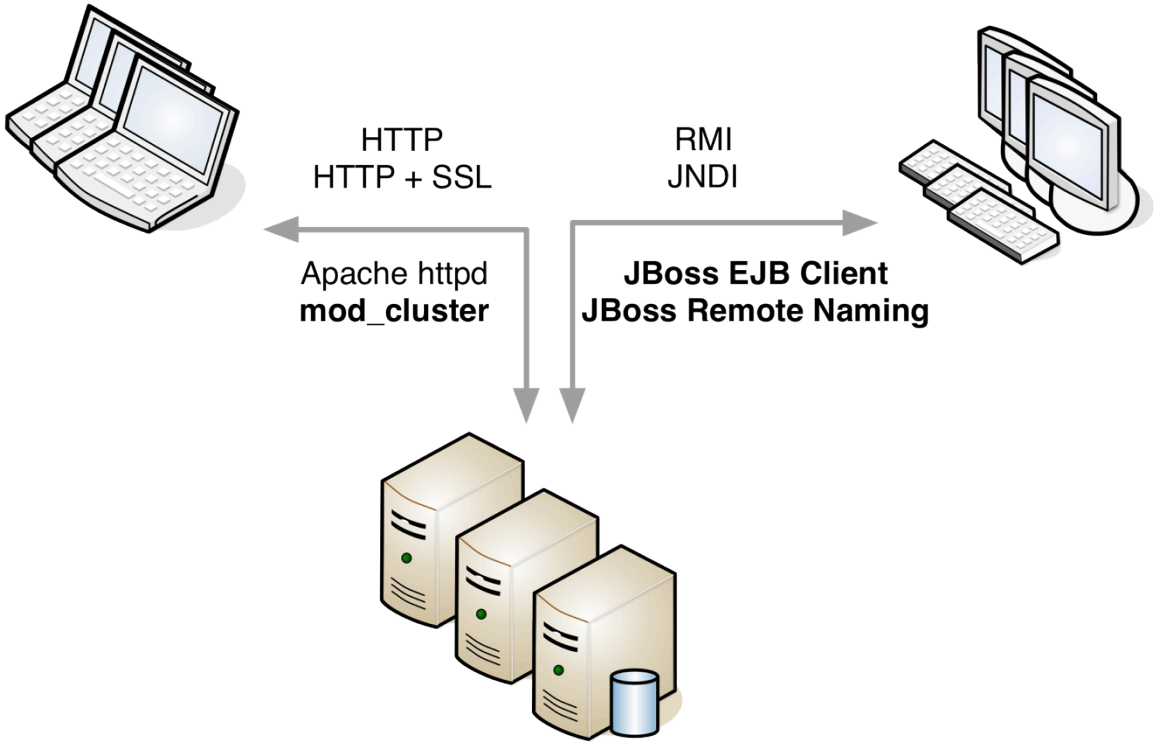


L1 - Cache mode

First Level Cache um Netzwerkaufrufe zu verringern



Lastverteilung



HTTP Lastverteilung - mod_cluster vs. mod_jk

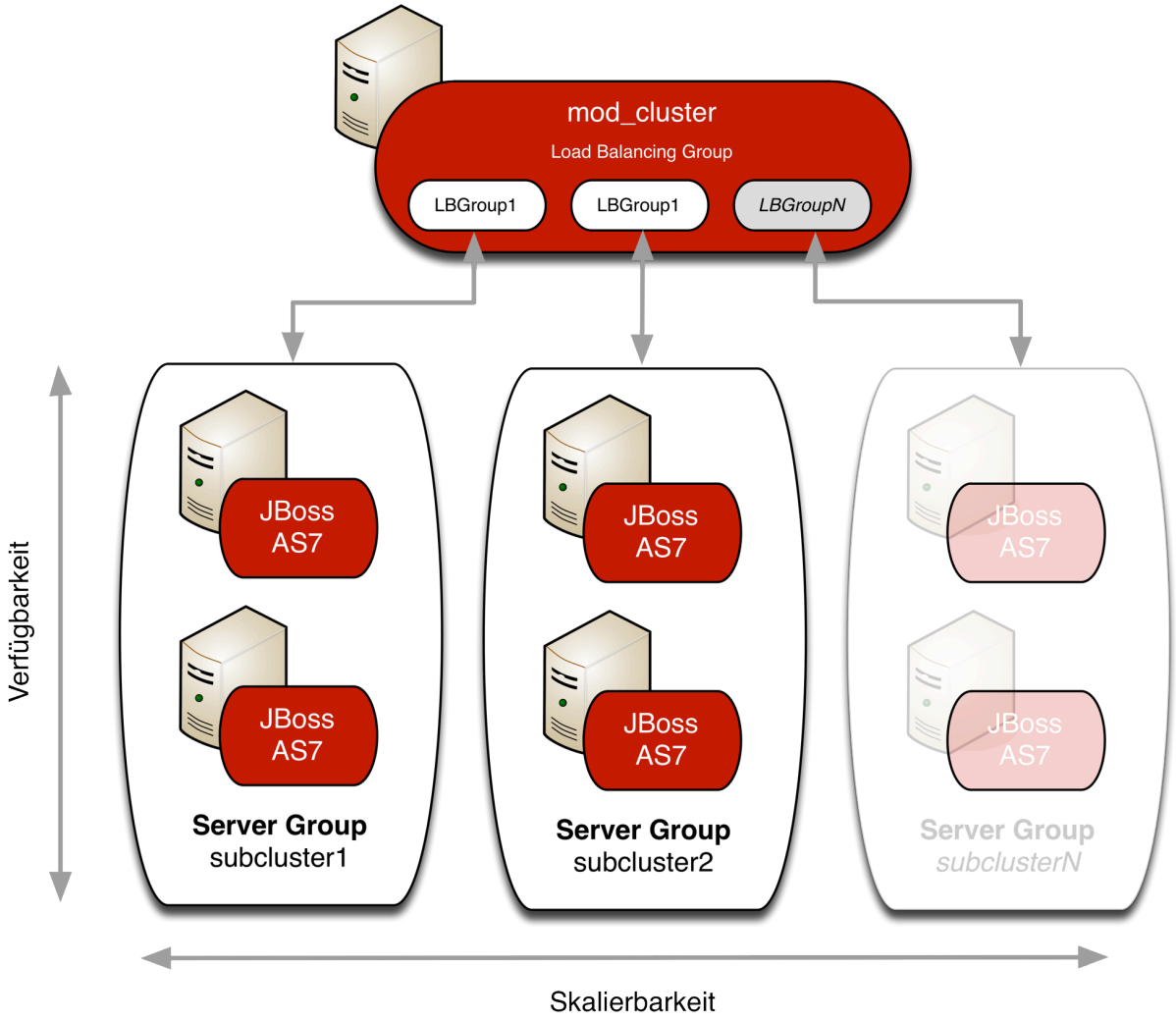
mod_jk benötigt statische Konfiguration

- Cluster Knoten hinzufügen oder entfernen (worker.properties)
- Anwendung hinzufügen oder entfernen (uriworkermap.properties)

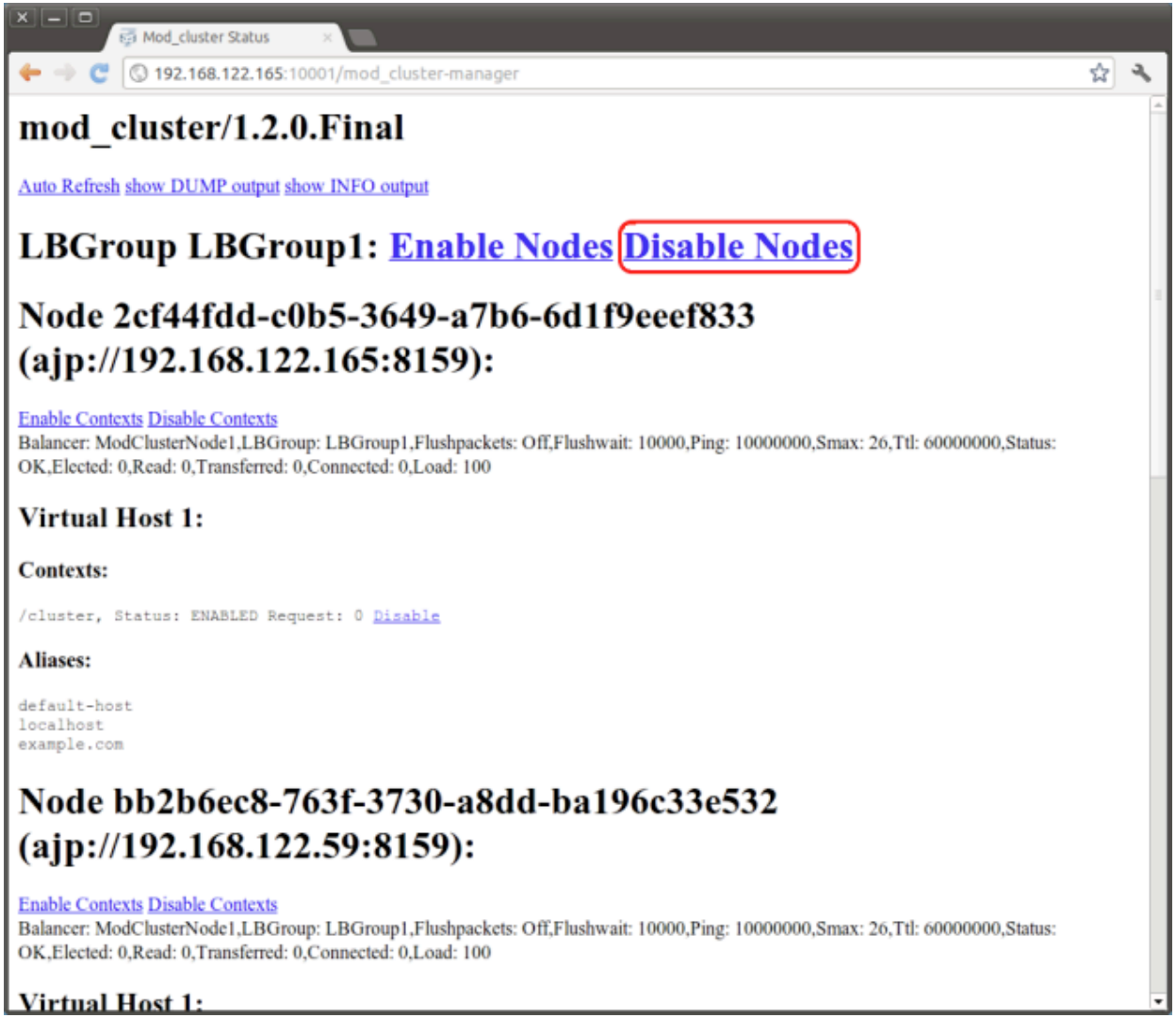
mod_cluster

- Dynamische Konfiguration der HTTPD worker
  Advertise mittels Multicast
- Intelligente Lastverteilung
  Einbezug von Lastverteilungsmetriken
  (cpu, mem, heap, sessions, receive-traffic, send-traffic, requests, busyness)
- Load-Balancing Groups (früher Domäne)

Load-balancing Groups und Partitionen



mod_cluster Manager



Entfernte EJB Aufrufe

Remote JNDI → JBoss Remote Naming

- entfernter Zugriff auf JNDI

Client JNDI → JBoss EJB Client

- Nur für EJB-Aufrufe
- Optimiert

HornetQ
Clustering und HA

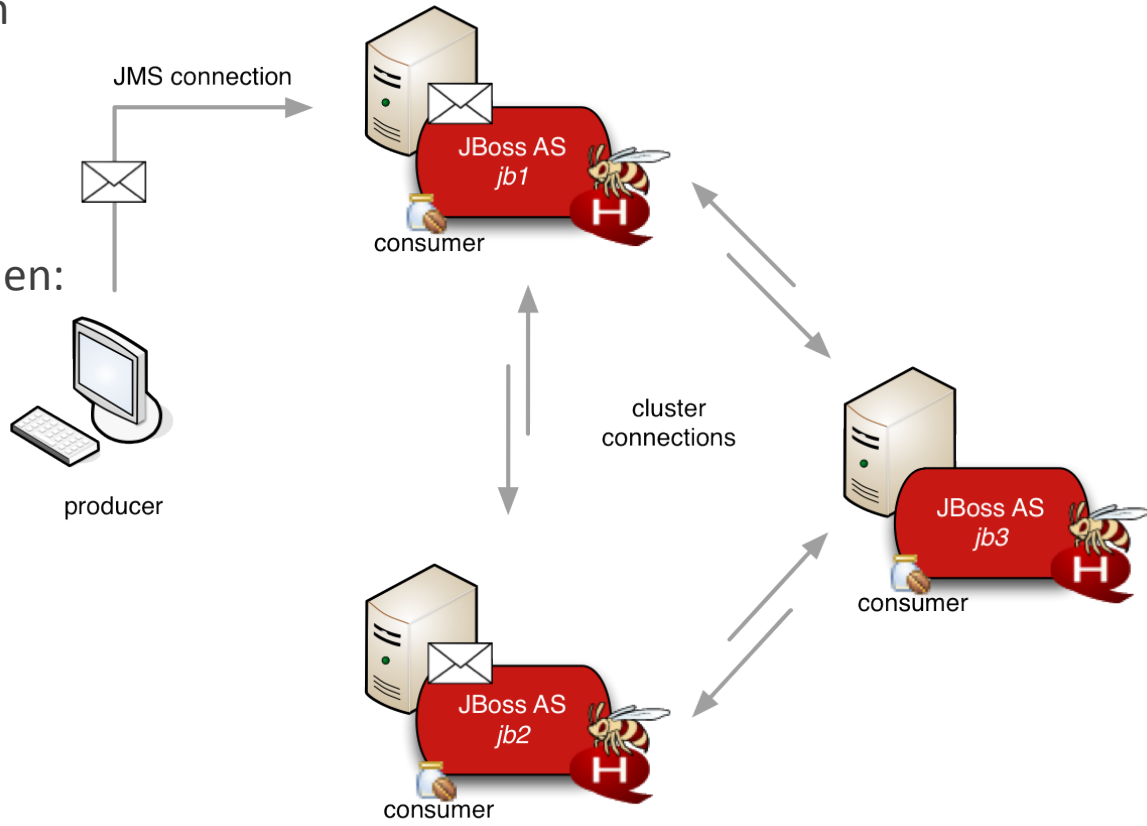
Motivation

1. Zuverlässigkeit
   bereits empfangene Nachrichten werden garantiert zugestellt (gehen nicht verloren)
2. höherer Nachrichtendurchsatz
3. Lastverteilung
4. automatisches Failover bestehender Verbindungen

1. Bietet bereits JMS
2. Auch möglich ohne HA und Clustering
3. Clustering
4. HA

Cluster-Connection zu jedem Nachbarn

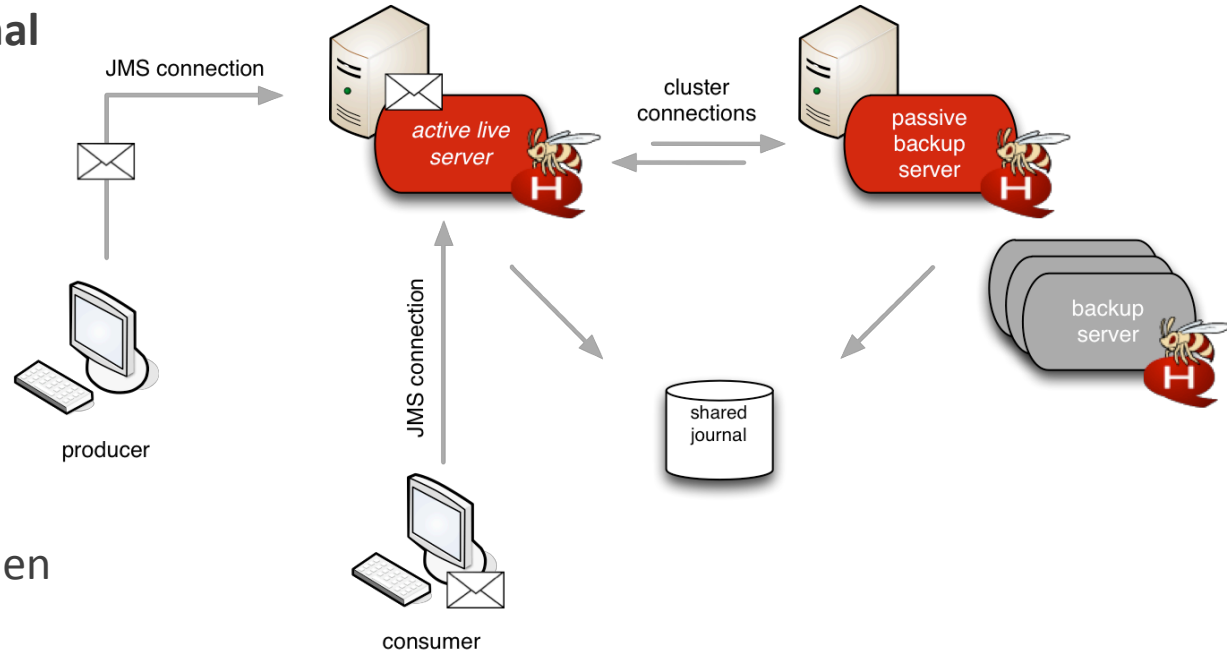
- empfangene Nachrichten werden weitergeleitet (RoundRobin)
- wenn kein passender Consumer: Message-Redistribution
- RoundRobin kann optimiert werden: nur an Knoten mit passendem Consumer



Hochverfügbarkeit (HA) mit HornetQ

mehrere Server teilen ein Journal

- gemeinsames Dateisystem
- Replikation über Netzwerk
  (ab Version 2.3 bzw. EAP 6.1)



Live-Backup Prinzip

- 1 Live
- 0-1 Backup
- 0-* bereit Backup zu werden

Backup Server ist eine JBoss AS 7 Instanz mit HornetQ Subsystem

HornetQ – Server Discovery

Server Discovery mehrere Server

- **UDP Multicast (oder JGroups Stack , ab 2.3)**
  - Broadcast-Group: Verbindungsdaten senden
  - Discovery-Group: Verbindungsdaten empfangen
- oder **statisch konfigurierte Nachbarn (Connectors)**

Diskussion, Fragen und Antworten

akquinet AG

heinz.wilming@akquinet.de
@akquinet
http://blog.akquinet.de