ORDIX AG

Loadbalancing und Clustering mit Tomcat 6

Java Forum Stuttgart
3. Juli 2008

Michael Heß
ORDIX AG, Paderborn
mhe@ordix.de
www.ordix.de

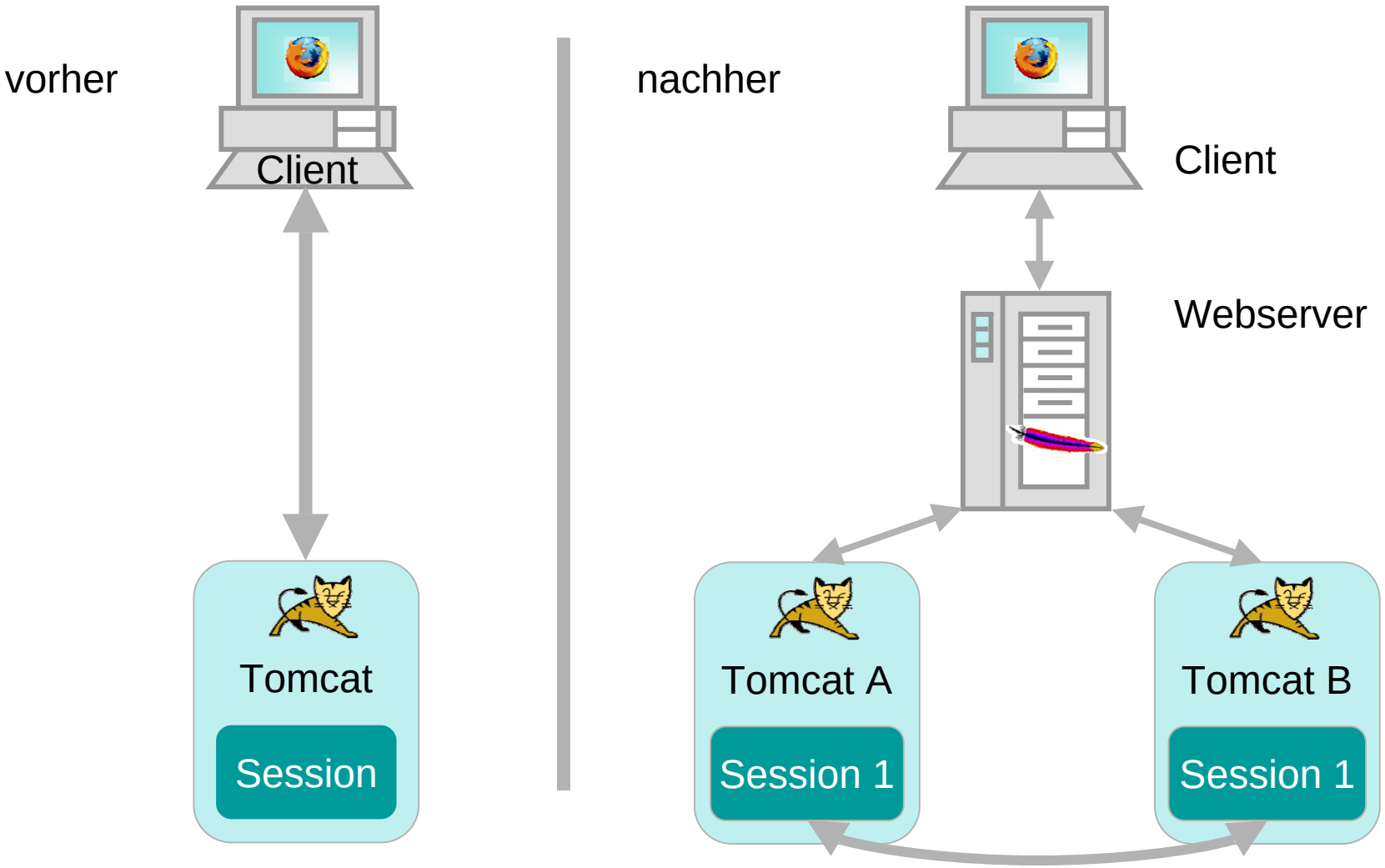
Agenda

ORDIX AG

- Zielsetzung des Vortrags
- Webserver Integration
- Loadbalancing
- Clustering
- Resümee

Zielsetzung des Vortrags

ORDIX AG



Agenda

ORDIX AG

- Zielsetzung des Vortrags
- Webserver Integration
- Loadbalancing
- Clustering
- Resümee

Webserver Integration

ORDIX AG

- Lastverteilung
- Transparenter Failover

- Als Webservermodul realisiert
  - Tomcat JK Connector (Apache, IIS, SunOne)
  - mod_proxy (Apache 2.2 spezifisch)

Kommunikation via AJP

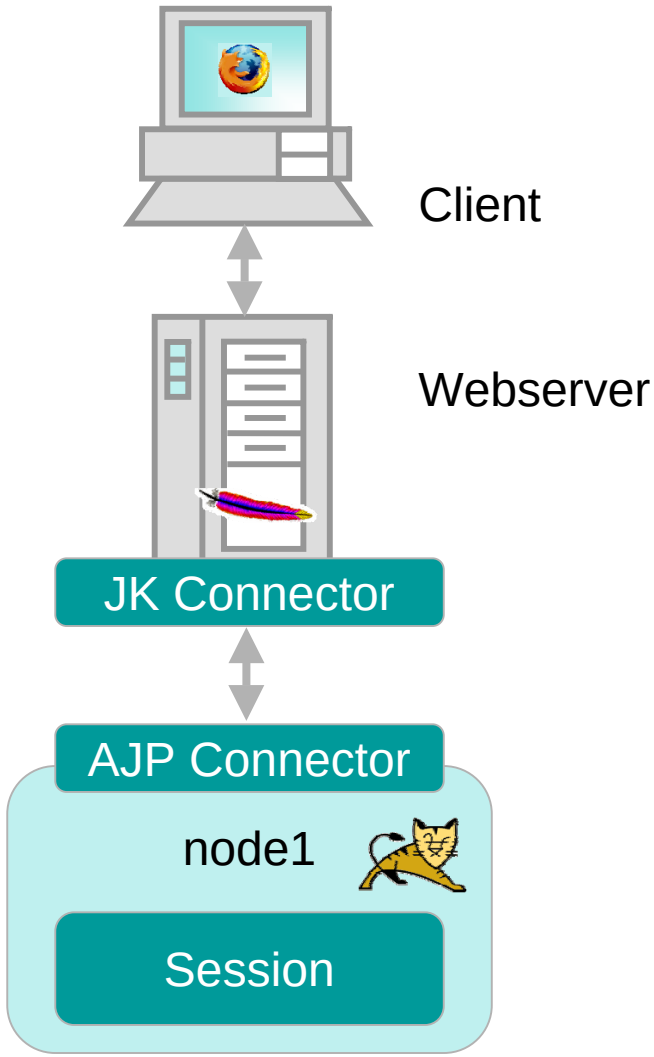
ORDIX AG

- Apache JServ Protocol v1.3 (ajp13)
- AJP ist ein binäres Protokoll

- Tomcat -> AJP Connector
- Webserver -> JK Modul

Beteiligte Komponenten

ORDIX AG



AJP Connector

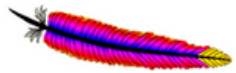
ORDIX AG

- Ermöglicht die Kommunikation via AJP
- Konfiguration analog zum HTTPConnector
  - port
  - minSpareThreads
  - maxSpareThreads
  - maxThreads

JK Connector

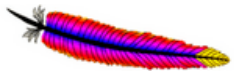
ORDIX AG

- Installation spezifisch für den jeweiligen Server
- Implementiert u. a. die Logik für Lastverteilung
- Steuert die Zuordnung URI -> Tomcat-Instanz
- Tomcat-Instanzen werden als Worker bezeichnet
- Webserver definiert „seine“ Worker



Workerkonfiguration

ORDIX AG



- In workers.properties
- Position im FS abhängig vom Webservertyp
- Auflistung aller definierten Worker

```
worker.list=node1,node2, ...
```

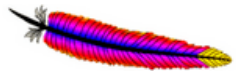
- Beschreibung einzelner Worker

```
worker.<name>.<direktive>=<wert>
```

- Gängige Direktiven:
  - Type (ajp13, lb, status)
  - Host
  - Port (→ AJP Connector Port, default 8009)

Beispiel Konfiguration

ORDIX AG

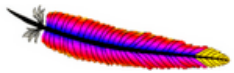


```
worker.list=node1,node2

worker.node1.type=ajp13
worker.node1.host=192.168.1.1

worker.node2.type=ajp13
worker.node2.host=192.168.1.2
```

Worker Mapping

ORDIX AG

- Welcher Request wird wohin geschickt?



- Spezifisch in Konfiguration des Webserver
- Generisch in urimapping.properties

- <URI>=<Worker>

```
/myApp   = node1
/myApp/* = node1
*.jsp    = node1
```

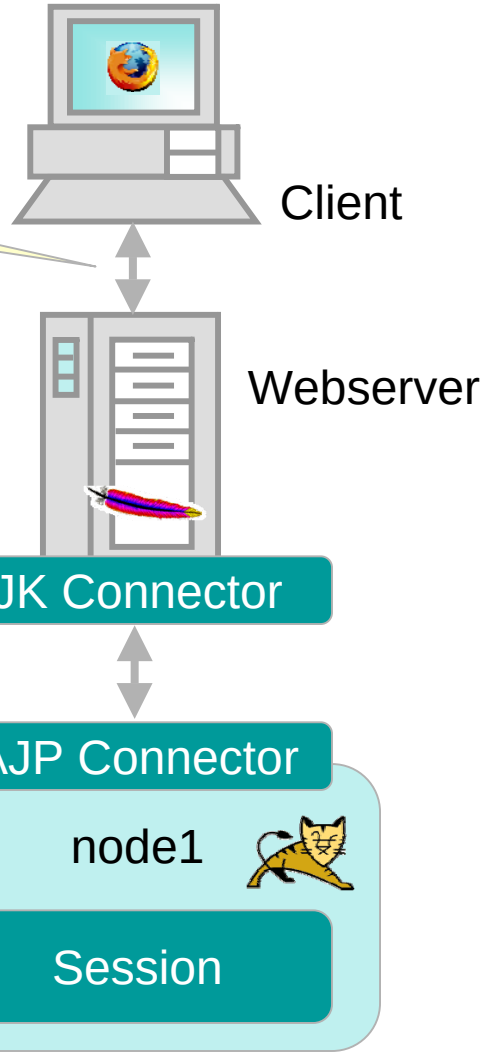
Request Zuordnung

ORDIX AG

Request:
http://site1/myApp

urimapping.properties:
/myApp = node1

workers.properties:
worker.node1.type = ajp13
worker.node1.host = 192.168.99.120



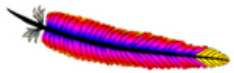
Agenda

ORDIX AG

- Zielsetzung des Vortrags
- Webserver Integration
- Loadbalancing
- Clustering
- Resümee

Loadbalancer

ORDIX AG



- Spezieller Worker Type „lb“
- Direktive balance_workers listet Instanzen
- Verschiedene Verteilungsmethoden
  - Requests, Sessions, Traffic, Busyness

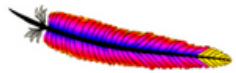
- Lastverteilung gesteuert über

```
worker.<name>.lbfactor
```

Beispiel Konfiguration Loadbalancer

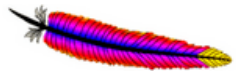
ORDIX AG

```
worker.list=lb
worker.lb.type=lb
worker.lb.balance_workers=node1,node2
[...]
worker.node1.lbfactor=2
[...]
worker.node2.lbfactor=3
```



Sticky Sessions

ORDIX AG



- Loadbalancer muss bestehende Sessions dem zugehörigen Knoten zuordnen
- SessionID enthält jvmRoute des Engine Elements
- worker Name == jvmRoute Attribut der Engine

Frage: Was passiert bei Ausfall des zugehörigen Knotens?

- „sticky_session_force“ Direktive des Loadbalancer
  - true -> HTTP Code 500 an Client senden
  - false -> Failover auf anderen Knoten (Verlust der Session)

Request Zuordnung mit Loadbalancer

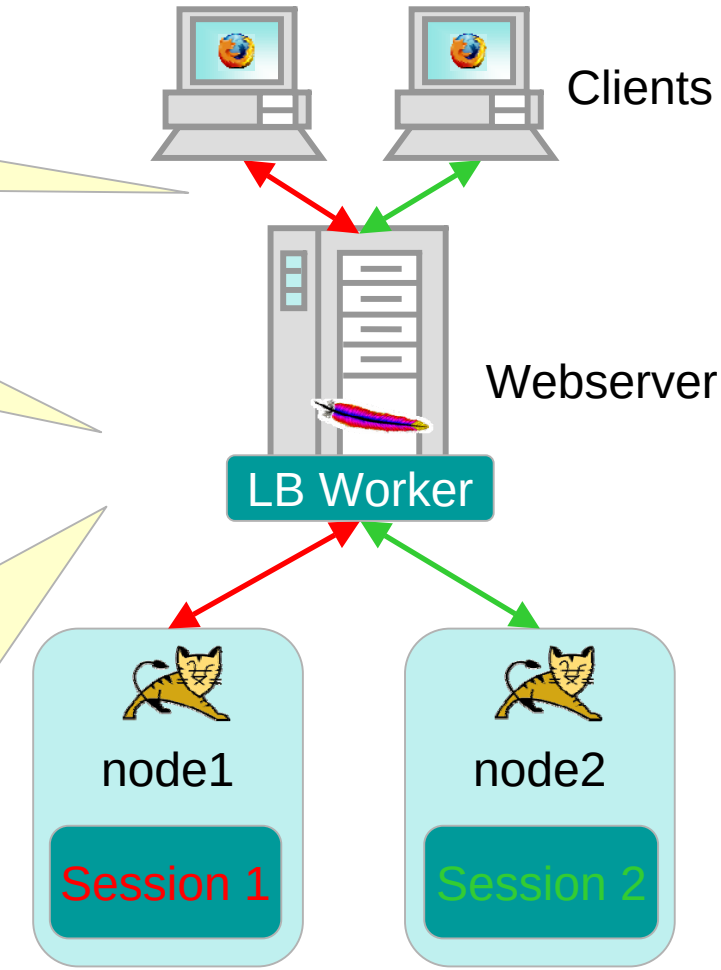
ORDIX AG

mehrere Requests für:
http://site1/myApp

urimapping.properties:
/myApp = lb

workers.properties:

worker.lb.type=lb
worker.lb.balance_workers=node1, node2
worker.lb.sticky_session=true
worker.lb.sticky_session_force=true
[...]



Agenda

ORDIX AG

- Zielsetzung des Vortrags
- Webserver Integration
- Loadbalancing
- Clustering
- Resümee

Motivation

ORDIX AG

- Sessions an Worker gebunden
- Ausfall des Workers = Verlust der Session

Lösung:
Cluster mit Sessionreplikation

- Jeder Clusterknoten kennt jede Session

Vorrausetzungen

ORDIX AG

- Alle Session-Elemente sind Serializable
- Die Anwendung ist verteilbar deklariert
  - im Deploymentdeskriptor:
    `<distributable/>`
  - in Contextdefinition:
    `distributable="true"`

Funktionsweise des Clusters

ORDIX AG

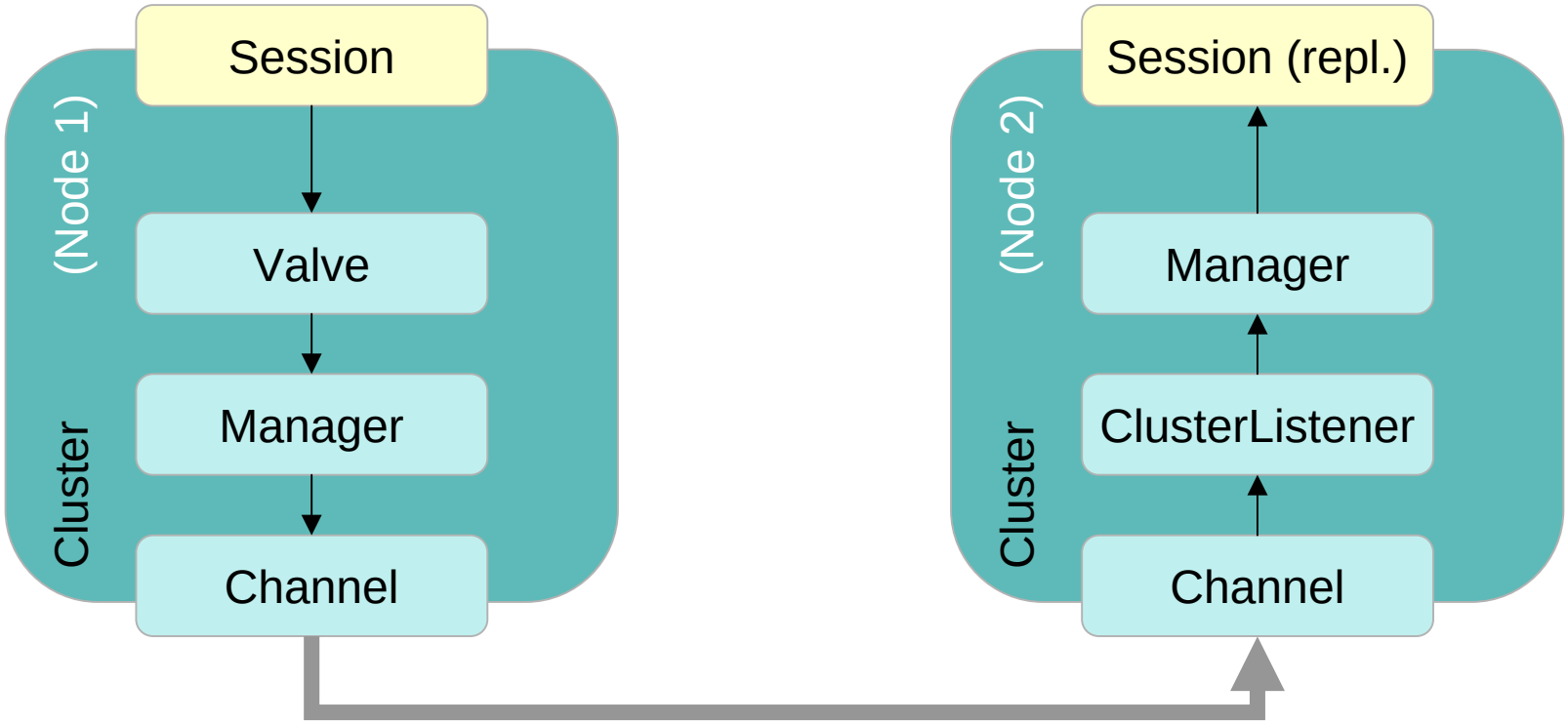
- Ziel: Nachrichtenaustausch zwischen Workern

- Mitgliedschaft dynamisch über UDP Multicasting
- Gegenseitige Überwachung via Heartbeat
- Nachrichtenversand zwischen Knoten via TCP

Sessionreplikation ist eine Anwendungsform

Überblick Nachrichtenversand

ORDIX AG

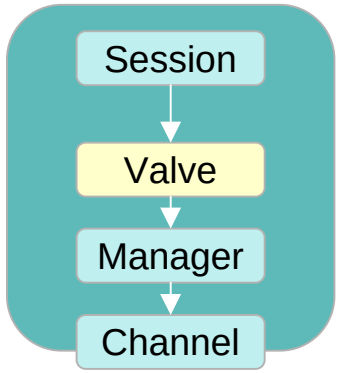


Valve

ORDIX AG

- Fügen sich wie „normale“ Valves in die äußere Requestpipeline ein

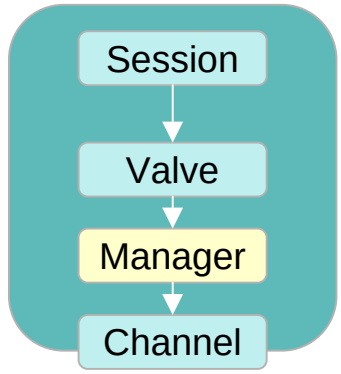
- ReplicationValve
  - Replikation via Manager anstoßen
  - Keine Replikation von statischem Content



Manager

ORDIX AG

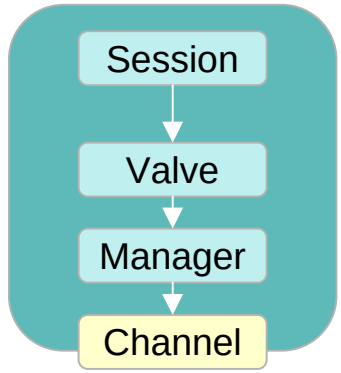
- Wählt „clusterfähigen“ SessionManager
  - DeltaManager
    - all-to-all Replikation
    - skaliert nicht gut, aber Gruppenbildung möglich
    - homogenes Layout erforderlich
  - BackupManager (ab Tomcat 6)
    - single-backup Replikation
    - heterogenes Layout möglich
- Kann pro Context individuell eingestellt werden



Channel

ORDIX AG

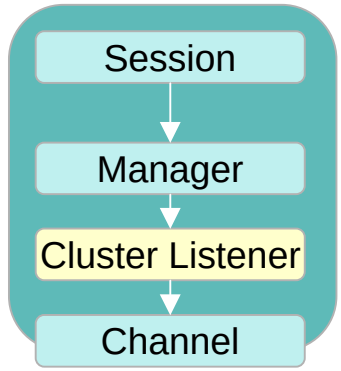
- Basis für Kommunikation innerhalb des Clusters
- Teil des Apache Tribes Framework
- Konfiguration durch diverse Unterelemente (s. u.)



ClusterListener

ORDIX AG

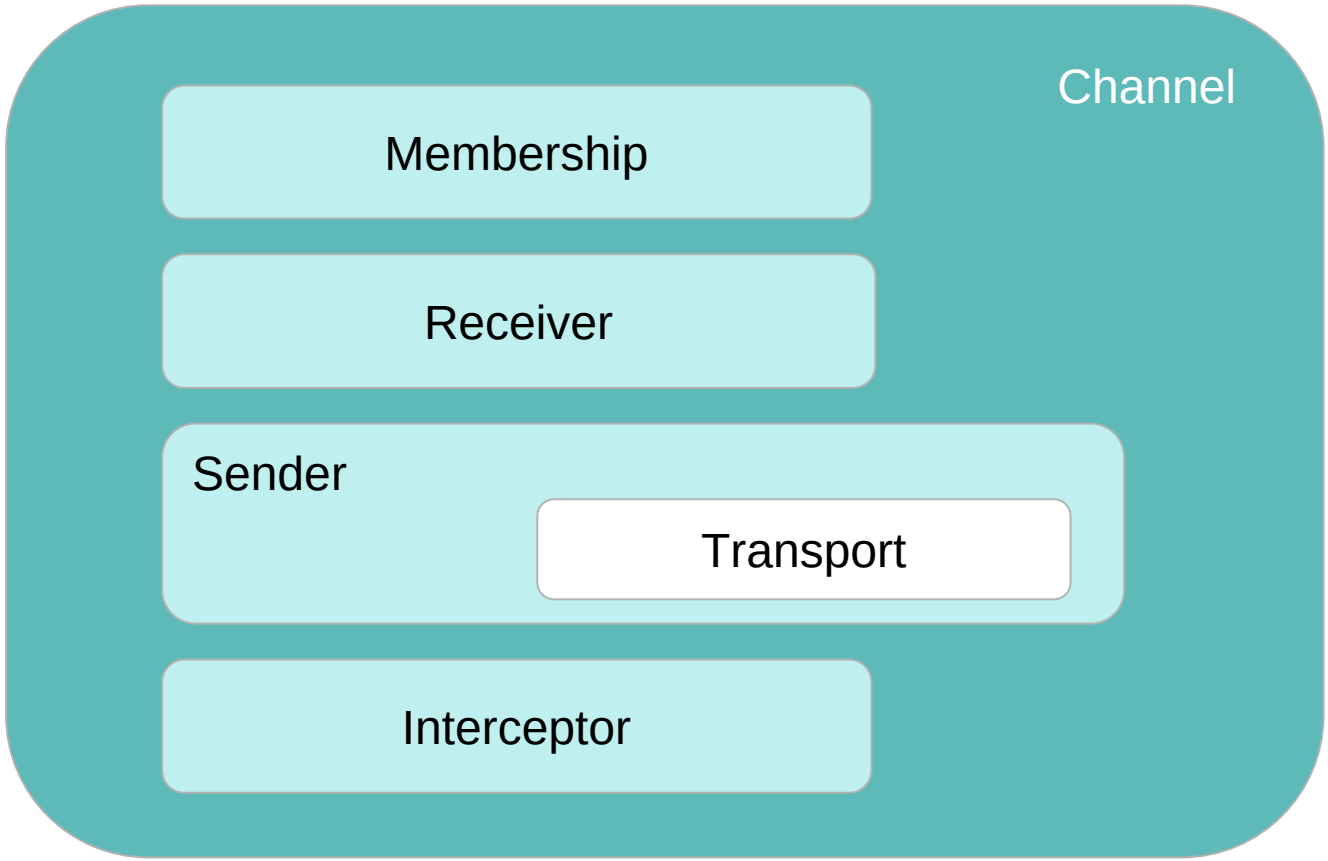
- Registrieren sich am Cluster
- Warten auf Nachrichten



- ClusterSessionListener
  - Bindeglied zwischen Cluster und Deltamanager
  - Weist Deltamanager an, lokale Sessions gemäß empfangener Nachrichten zu aktualisieren

Channel Aufbau

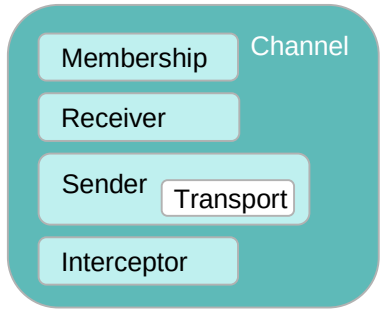
ORDIX AG



Channel

ORDIX AG

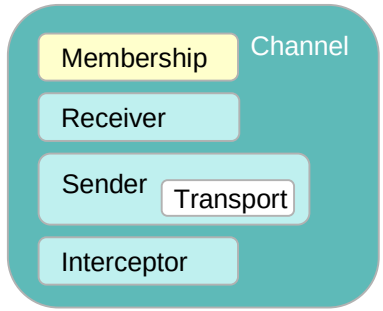
- Verschiedene Versandarten (channelSendOptions)
  - Asynchronous
  - Asynchronous Acknowledge
  - Synchronized Acknowledge



Membership

ORDIX AG

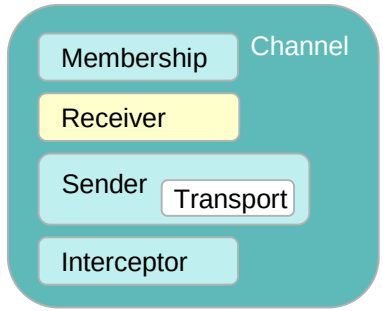
- Dezentrale Cluster-Mitgliedschaft
- Versendet Heartbeat via Multicast
- Heartbeat enthält Kontaktdaten (Receiver)
- Attribute auf allen Knoten gleich
  - McastAdresse und Port
  - frequency – Heartbeatfrequenz in ms
  - dropTime – Timeout in ms für toten Nachbar
- Hat keine Rolle beim Nachrichtenversand



Receiver

ORDIX AG

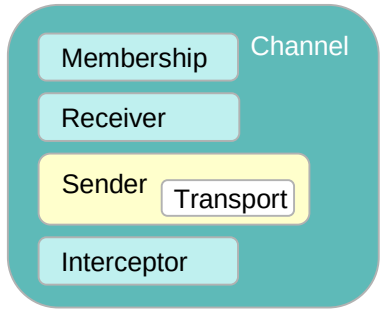
- Empängt Clusternachrichten via TCP
- non-/blocking Varianten
- Anzahl Threads für Listener individuell
  - Anzahl Knoten – 1
  - Bei parallelem Nachrichtenversand ggf. mehr



Sender

ORDIX AG

- Nachrichtenversand in den Cluster
- „Parallel Concurrent Messaging“
  - Parallel → 1 Nachricht an n Empfänger
  - Concurrent → n Nachrichten an 1 Empfänger

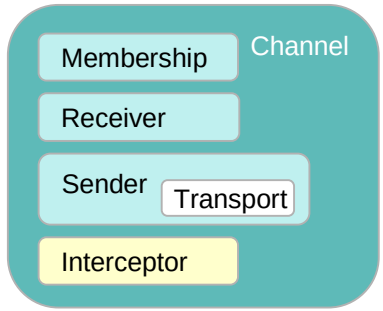


- Poolsize steuert die maximalen Concurrent Messages pro Knoten

Interceptor

ORDIX AG

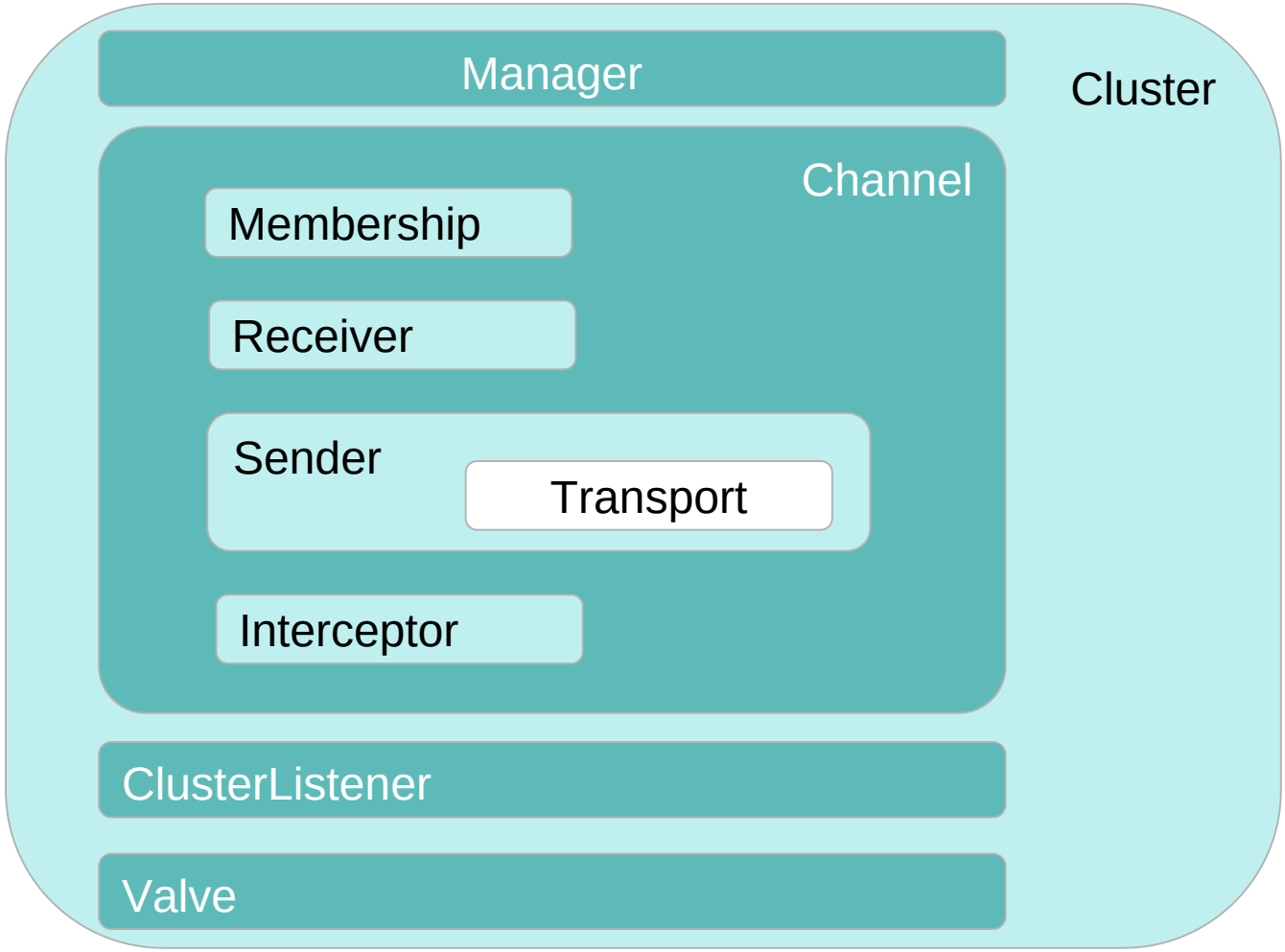
- Manipulation von Nachrichten (Rx/Tx)
- Interceptor bilden einen Stack
- Reihenfolge in der server.xml ist entscheidend



- Beispiel: TcpFailureDetector
  - UDP Timeout erreicht
  - Prüfung via TCP, ob der Knoten tatsächlich tot ist

Gesamtübersicht Clusteraufbau

ORDIX AG



Performance vs. Replikation

ORDIX AG

- Einbußen sind moderat, aber spürbar
- Abhängig von vielen Parametern, u. a.
  - Anzahl der Knoten
  - Anzahl der Nachrichten
  - Größe der Nachrichten (Session-Elemente)
- Individuelle Anpassungen an Anwendung erforderlich

Request Zuordnung bei Ausfall

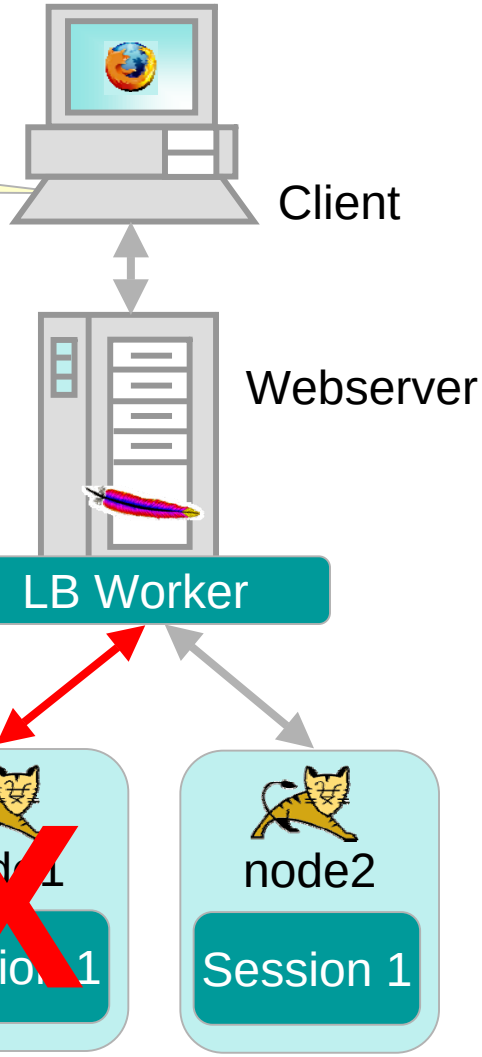
ORDIX AG

Request:
http://site1/myApp?jsessionid=session1.**node1**

Versuch Request an Node 1 weiterzuleiten, dann Failover auf Node 2

Node 2 kennt Session von Node 1 und bearbeitet den Request

(und führt ggf. Rebinding der Sitzung durch)



Agenda

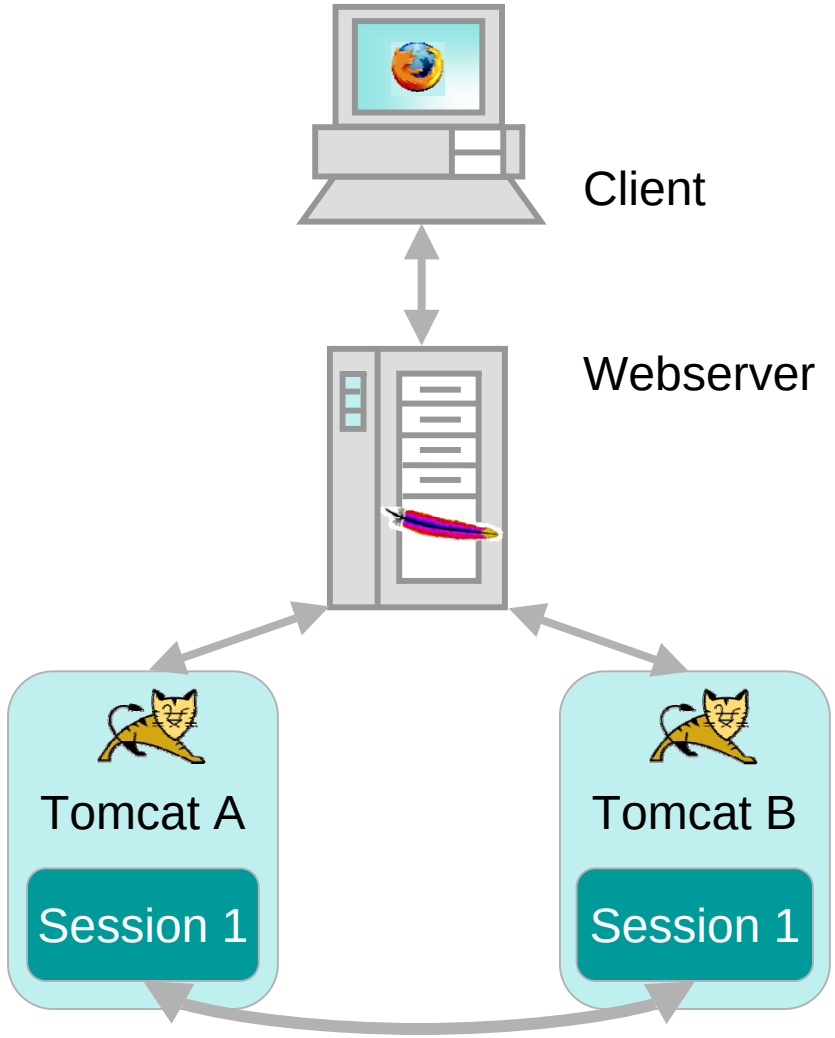
ORDIX AG

- Zielsetzung des Vortrags
- Webserver Integration
- Loadbalancing
- Clustering
- Resümee

Resümee

ORDIX AG

- JK Connector verteilt Requests
- Sessions werden auf zugehörigen Knoten geleitet
- Cluster Knoten bilden Multicast Verbund
- Knoten benachrichtigen sich gegenseitig über geänderte Sessions
- Bei Ausfall eines Knotens wird der Request auf einen anderen Knoten geleitet



ORDIX AG

Vielen Dank
für Ihre Aufmerksamkeit!

einfach.gut.beraten.