trivadis
Part of Accenture

EJB Clustering mit JBoss

AUTOR

Sabine Winkler
Orientation in Objects GmbH

Veröffentlicht am: 20.1.2003

ABSTRACT

Skalierbarkeit ist eine wichtige Anforderung an J2EE Anwendungen. Die J2EE Architektur wirbt mit diesem Schlagwort und bietet mit der Enterprise Edition eine umfangreiche Technologie, die dies verspricht. Wieviele Benutzeranfragen und Transaktionen eine Anwendung sicher und skalierbar bearbeiten kann, können die Spezifikationen allein nicht beantworten. Hier richtet sich der Blick in die Welt der Application Server, die einen entscheidenden Anteil zur Realisierung dieser Anforderungen besteuern und Lösungen versprechen.

Dieser Artikel ist ein Einstieg in die Welt des EJB Clusterings mit JBoss. Weitere Informationen zum Thema **JBoss Clustering** finden Sie in der gleichnamigen Dokumentation der **JBoss Group**.

) Schulung)

) Beratung)

) Entwicklung)

) Artikel)

Trivadis Germany GmbH

Weinheimer Str. 68
D-68309 Mannheim

Tel. +49 (0) 6 21 - 7 18 39 - 0
Fax +49 (0) 6 21 - 7 18 39 - 50

www.oio.dekontakt@trivadis.com

CLUSTERING

Unter folgenden drei Gesichtspunkten lässt sich EJB Clustering betrachten:

- **Lastverteilung** *(Load balacing)*

  Kann durch den Einsatz von mehreren Servern in einem Cluster eine höhere Performanz erreicht werden?

- **Hohe Erreichbarkeit** *(High availability)*

  In welcher Zeit ist ein Service erreichbar? Was bedeutet "hohe Erreichbarkeit" aber?

- **Fehlertoleranz** *(Fault tolerance)*

  Wird Transaktionssicherheit und Datenkonsistenz gewährleistet, wenn Fehler auftreten, wie z. Bsp. der Ausfall eines Servers ?

DER JBOSS APPLICATION SERVER

Eine Alternative im Bereich der J2EE Application Server stellt das OpenSource Projekt JBoss mit dem JBoss Application Server dar (www.jboss.org). Mit der 3. Version wird zum ersten Mal die Möglichkeit des Clusterings angeboten. Die wichtigsten Clustering Features des JBoss sind:

- Automatisches Erkennen der Knoten in einem Cluster ohne zusätzliche Konfiguration
- Dynamische JNDI Erkennung
- Clusterweite Replikation des JNDI Baumes
- Hot Deploy clusterweit durch den "Farming" Service
- Lastverteilung und Ausfallsicherheit für JNDI, RMI, Entity Beans, Stateful / Stateless Session Beans

Im Zusammenhang mit Clustering wird beim JBoss Application Server der Begriff *Partition* verwendet. Eine Partition kann man äquivalent zu einem Cluster verstehen. Jeder JBoss Server stellt für sich eine Partition dar, d.h. er ist einer Partition per default zugeordnet. Befinden sich nun mehrere JBoss Server in einem Cluster, bedeutet dies, sie laufen innerhalb derselben Partition. So ist es möglich, dass beispielsweise eine JBoss Instanz gleichzeitig mehreren Partitionen zugeordnetet sein kann und somit gleichzeitig in verschiedenen Clustern einen Knoten darstellt.

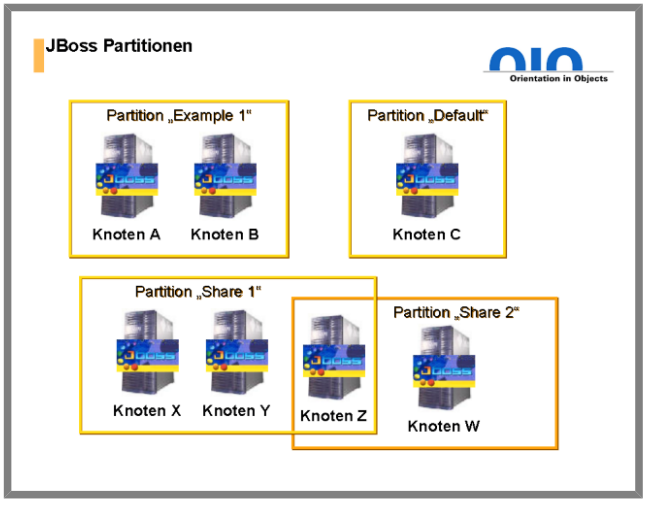

Abbildung 1: Verschiedene Möglichkeiten von Partitionierungen mit JBoss Instanzen

Durch die Partitionierung können im gleichen Netzwerk verschiedene Cluster betrieben werden. Jeder Partition wird ein eigener Name zugeordnet. Somit ist eine Unterscheidung möglich. In der aktuellen Version des JBoss ist die Implementierung der Partitionen durch das JavaGroups Framework realisiert. (http://javagroups.sourceforge.net)

Um eine spezifischere Verteilung von Komponenten in größeren Clustern vornehmen zu können, beinhaltet das Konzept der Paritionierung Subpartitionen. In der momentan verfügbaren Version des JBoss ist die Möglichkeit einer Subpartitionierung noch nicht implementiert. Allerdings soll in künftigen Versionen diese Funktionalität realisiert werden.

FAILOVER ZWISCHEN THEORIE UND PRAXIS

Datenkonsistenz - Wie kann diese gewährleistet werden, wenn ein Server Knoten ausfällt ?

CLIENT MANAGED FAILOVER

Stellt der Client ein Anfrage an einen Server Knoten, der gerade ausgefallen ist, wäre eine Lösung, dass der Client von sich aus das Wissen mitbringen würde, sich zu einem anderen, laufenden Knoten zu verbinden (s. Abbildung 2). Fraglich hierbei ist die fehlende Transparenz für den Client. Wie soll das Wissen über weitere verfügbare Knoten clientseitig implementiert werden ?

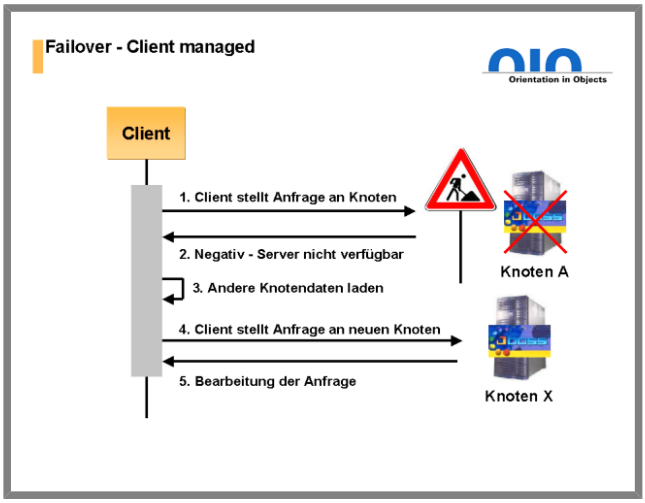

Abbildung 2: Clientseitige Behandlung bei Ausfall eines Cluster Knotens

DISPATCHER MANAGED FAILOVER

Eine Dispatcher Komponente zwischen Client und Cluster könnte die Aufgabe der Weiterleitung der Client Anfrage beim Ausfall eines Servers ebenso übernehmen. Der Client sendet seine Anfragen direkt an den Dispatcher und dieser leitet die Anfrage an die Knoten im Cluster weiter. Fällt ein Knoten im Cluster aus, wie Abbildung 3 zeigt, muss der Dispatcher die Anfrage intern an einen anderen Knoten zustellen und die Gewährleistung übernehmen, dass auf jeden Fall die Bearbeitung der Client Anfrage durchgeführt wird.

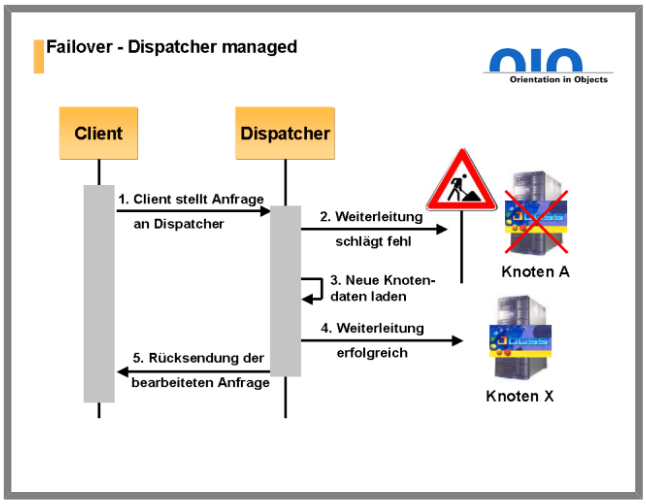

Abbildung 3: Fehlerumgehung durch eine Dispatcher Komponente

Zunächst scheint dies eine gelungene Lösung im Vergleich zum client managed Failover darzustellen. Das "Innovative" stellt gleichzeitig jedoch das Risiko dieses Ansatzes dar - der Dispatcher. Mit ihm steht und fällt der Ansatz, d.h. fällt die Dispatcher Komponente aus, kann ebenso keine Sicherheit für Datenkonsistenz gewährleistet werden.

SMART PROXY - DIE JBOSS LÖSUNG FÜR FAILOVER

JBoss macht sich die Fähigkeiten von RMI zunutze, um auf den Ausfall eines Knotens reagieren zu können. In einer eigenen Implementierung, HA-RMI *(HA = high available),* wird bei jeder Anfrage des Clients eine Proxy Klasse mit heruntergeladen. Dieser Proxy beinhaltet eine Liste über alle momentan verfügbaren Zielknoten im Cluster und Informationen zu einem weiteren Feature, der dynamisch veränderbaren *load balancing policy.* D.h. die Strategien zur Lastverteilung können zur Laufzeit verändert werden.

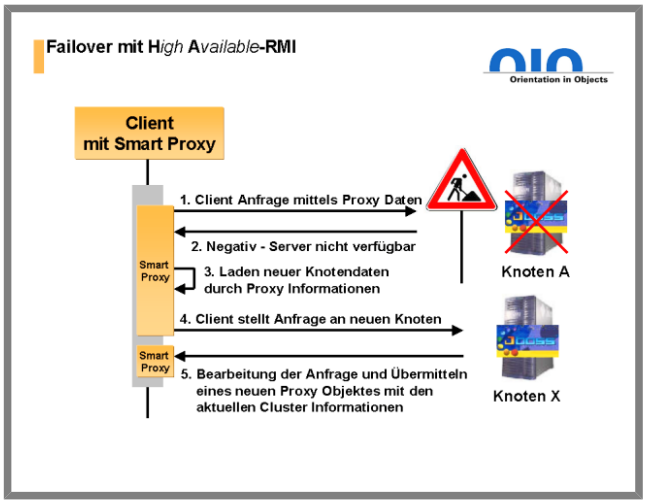

Abbildung 4: Das Smart-Proxy Konzept vom JBoss

Verändert sich die Topologie des Clusters, wird bei der nächsten Client Anfrage eine neue Liste mittels des übertragenen Proxy Objektes gesendet, die den aktuellen Zustand widerspiegelt (s. Abbildung 4).

Dieser Entwurf stellt eine transparente Lösung dar, bringt jedoch eine Client-seitige Abhängigkeit zu RMI mit sich. Sofern der Client diese Kommunikation nicht nutzt, sind ihm die Clustering Features des JBoss nicht zugänglich. Lösungsansätze zu dieser Problematik liegen bisher nur im Theoretischen vor. Eine Implementierung wird seitens JBoss zur Zeit nicht unterstützt.

CLUSTERED NAMING SERVICE - HA JNDI

Um Failover und Load balancing im JBoss zu nutzen, verbindet sich der Client zu HA-JNDI, einem clusterweiten, globalen JNDI Context. HA-JNDI ermöglicht die automatische Replikation von allen gebunden Objekten innerhalb des Clusters. Die Objekte bleiben verfügbar, unabhängig davon, ob ein Knoten ausfällt oder nicht. HA-JNDI arbeitet jeweils mit dem lokalen JNDI der einzelnen Knoten zusammen. Sind beispielsweise Objekte nur im lokalen JNDI Tree gebunden, können diese ebenso über HA-JNDI erfragt werden, da eine Anfrage automatisch an das lokale JNDI delegiert wird (s. Abbildung 5).

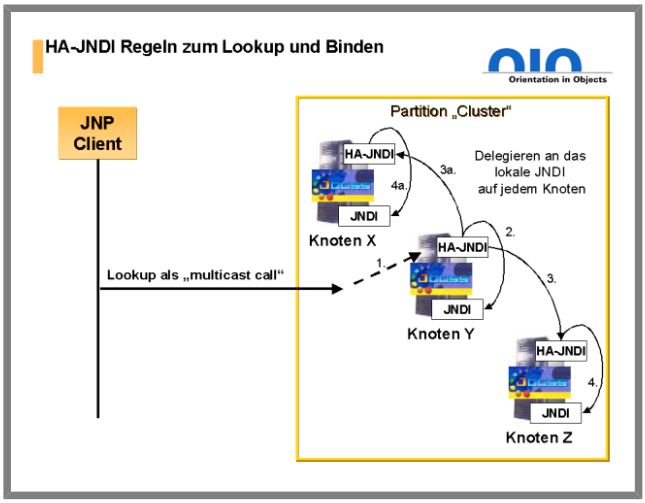

Abbildung 5: Möglicher Ablauf eines Lookup innerhalb des Clusters

Der Client kann nun auf einfache Weise mit HA-JNDI arbeiten. Wird in den *jndi.properties* für die Angabe der *java.naming.provider.url* die Property leer gelassen, produziert der JNP Client einen "*multicast call*" im Netzwerk und erhält automatisch einen HA-JNDI Server. Eine weitere Möglichkeit besteht darin, bei der oben genannten Property die einzelnen Serveradressen anzugeben. Der JNP Client probiert in diesem Fall, der Reihe nach Verbindung mit den angegebenen Servern aufzunehmen. Beim ersten erfolgreichen Verbinden erhält er vom Server wieder die aktuelle Cluster Topologie mittels der Proxy Klasse mitgeteilt. Für den Fall, dass kein Server in der angegebenen Property verfügbar ist, produziert der JNP Client widerum einen "*multicast call*" zur automatischen Erkennung.

ZUSAMMENFASSUNG

Die Entwicklung zeigt, dass der JBoss als OpenSource Projekt das EJB Clustering für jeden Anwender, unabhängig von Fragen des Geldes und speziellem Know How, ermöglicht. Dieser Artikel hat einen Einblick in die Möglichkeiten des EJB Clusterings mit JBoss gegeben und kurz aufgezeigt, wie der JBoss Application Server die einzelnen Anforderungen des EJB Clusterings technisch realisiert.

- Starten und konfigurieren eines JBoss Clusters

- Clustering von Enterprise Java Beans im JBoss Application Server
- Der Farming Service des JBoss
- Einrichten einer EJB "Farm"

Tip: Unsere neuesten Veröffentlichungen sowie Informationen zu anderen Themen im Bereich *Java und XML* können Sie ebenso durch unseren Newsletter erfahren.