High Performance JPA mit Coherence

Markus Eisele
msg systems ag
Ismaning

Schlüsselworte
JPA, Coherence, EclipseLink

Einleitung
JPA ist die State-of-the-Art Technologie um aus Java heraus Objekte in relationalen Datenbanken zu hinterlegen. Manchmal ist die Performance die man erreichen kann dennoch nicht ausreichend. Der Vortrag stellt die Szenarien vor in denen Coherence als JPA Backend verwendet werden kann und gibt einen Überblick über Vor- und Nachteile.

Was ist Coherence?

Oracle Coherence ist eine In-Memory Data-Grid-Lösung. Mit ihrer Hilfe können Mission-Critical-Anwendungen sehr schnellen und skalierbaren Zugriff auf häufig verwendete Daten erlangen. Durch die automatische und dynamische Partitionierung von Daten im Speicher über mehrere Server hinweg ermöglicht Coherence durchgehende Verfügbarkeit und bietet transaktionale Integrität, auch im Falle eines Server-Ausfalls. Es ist eine gemeinsam genutzte Infrastruktur, die Lokalität der Daten kombiniert mit lokalen Rechenleistung, um Echtzeit-Daten-Analyse durchzuführen, in-memory Gitter Berechnungen und parallel Transaktions-und Event-Verarbeitung.

Strategie im Überblick
Data Grid Lösungen haben eins gemeinsam. Eine sehr einfache put()/get() API, welche Objekte in den Cache speichern bzw. daraus erneut lesen kann. Bei der Verwendung von Coherence im Kontext von JPA ergeben sich beim Anblick der einfachen API eine Menge von Fragen. Der Vortrag betrachtet die drei Möglichkeiten Coherence in Kombination mit der Java Persistence API einzusetzen. Dies geschieht am Beispiel der Referenz-Implementierung EclipseLink welche Teil des GlassFish Applikationsservers ist.
Strategie 1 ist die direkte Verwendung der Coherence API unter Verwendung einer Datenbank die über JPA angebunden ist. Dabei synchronisiert Coherence seinen Datenbestand regelmäßig via EclipseLink mit einer angebundenen DB.
Strategie 2 stellt die Verwendung von Coherence als Second Level Cache (L2) für EclipseLink dar. Damit kann eine optimierte Lesegeschwindigkeit beim Zugriff auf Daten via JPA erreicht werden.
Strategie 3 letztendlich ist die Kombination beider bereits vorgestellter Möglichkeiten und wird als Komplettlösung betrachtet.

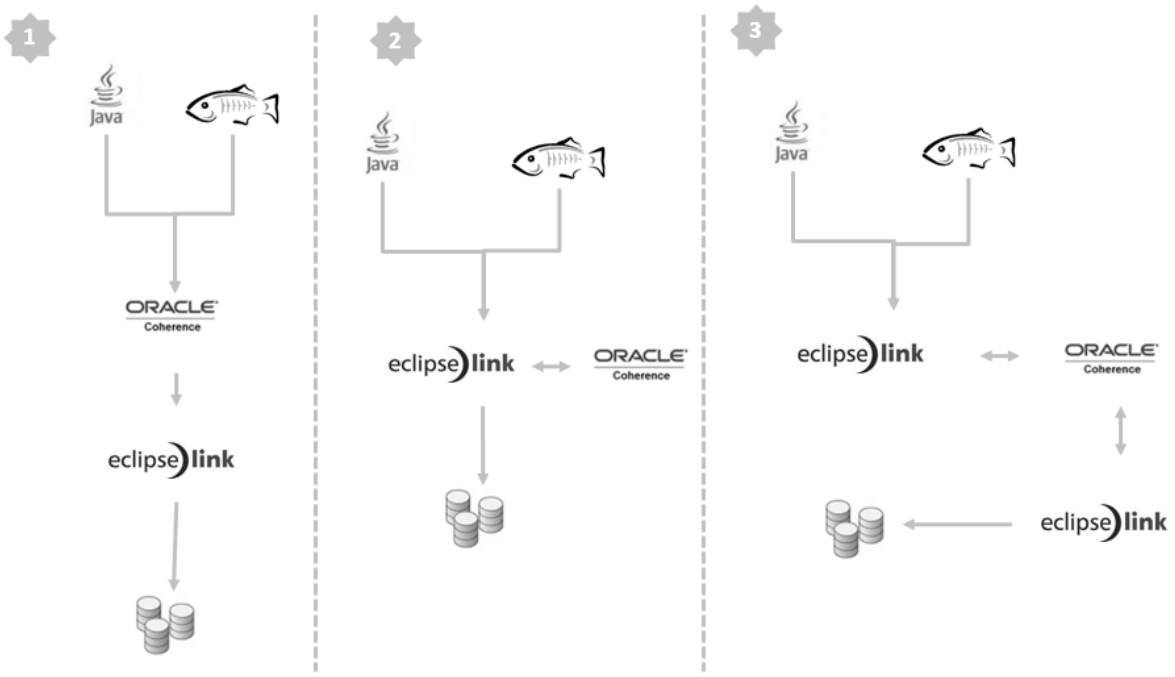

Abbildung 1 - Wege zur JPA on the Grid

JPA backed Caches
Bei dieser Strategie wird über die klassischen Coherence-API's programmatisch auf den Cache zugegriffen. Dieser ist wieder über JPA mit einer Datenbank verbunden. Über eine CacheLoader und CacheStore Implementierung, welche für EclipseLink optimiert ist, wird dieser "traditionelle" Ansatz dabei mit der Datenbank verbunden. Die beiden Implementierungen (EclipseLinkJPACacheLoader oder EclipseLinkJPACacheStore) findet man in der toplink-grid.jar-Datei. Darüber hinaus wird noch die JPA run-time-Konfigurationsdatei persistence.xml und zusätzlich die JPA-Mapping-Datei orm.xml benötigt. Die Coherence-Cache Konfigurationsdatei coherence-cache-config.xml muss ebenfalls angegeben werden, um die Standard-Coherence-Einstellungen zu überschreiben und die Cachespeicher-Schema zu definieren.

JPA mit Second Level Cache (L2)
Die als „Grid-Cache" bekannte Konfiguration verwendet Coherence als JPA shared (L2) Cache. Dabei wird bei Abfragen auf Primärschlüsseln zuerst versucht, diese aus Coherence zu laden. Erst wenn dies nicht erfolgreich ist, wird eine Datenbank-Abfrage durchgeführt. Dabei wird Coherence mit dem Ergebnis der Abfrage automatisch aktualisiert. Nicht-Primärschlüssel Abfragen hingegen gehen direkt gegen die Datenbank. Dabei werden die gelieferten Ergebnisse mithilfe von Coherence überprüft um ggf. Objekt Instanziierungskosten für bereits zwischengespeicherte Objekte zu vermeiden. Bei Abfragen nach Objekten, welche noch nicht im Coherence Cache liegen wird Coherence entsprechend aktualisiert um bei erneuten Abfragen einen höheren Durchsatz zu erreichen. Schreib-Operationen hingegen aktualisieren die Datenbank direkt.

JPA L2 und JPA backed Cache
Die Kombination aus den bereits vorgestellten Ansätzen ist unter dem Namen "Grid-Entity" bekannt und eignet sich besonders für Anwendungen, welche schnellen, lesenden Zugriff auf große

Datenmengen mit vergleichsweise wenigen Updates benötigen. Dieses kann per „Write-Behind“ noch weiter optimiert werden um durch asynchrone Datenbank-Updates die Geschwindigkeit weiter zu verbessern.

Kontaktadresse:
Markus Eisele
msg systems ag
Robert-Bürkle-Str. 1
D-85737 Ismaning

Telefon:	+49 (0) 89-96 101 0
Fax:	+49 (0) 89-96 101
E-Mail	markus.eisele@msg-systems.com
Internet:	www.msg-systems.com