WebLogic - Der beste Application Server für die Oracle Datenbank

Thomas Robert
Oracle Deutschland B.V. & Co. KG
Niederlassung Hamburg

Schlüsselworte

WebLogic Server, Oracle Database, Hochverfügbarkeit, JDBC Replay, Application Continuity, Transaction Guard, Database Resident Connection Pool, Pluggable Database, Global Data Servíces

Einleitung

Oracle entwickelt kontinuierlich Funktionalitäten für die Datenbank, um den Zugriff von Anwendungen noch sicherer, performanter und ressourcenschonender zu ermöglichen. Diese im Datenbankserver bzw. im Oracle JDBC Treiber vorhandenen Features stehen zwar grundsätzlich allen Datenbank-Clients zur Verfügung und sind in der Regel transparent für den Anwendungscode, allerdings erfordert ihre Nutzung manchmal eine Anpassung an der Konfiguration der Java Connection Pools bzw. Datasources.

Der Oracle WebLogic Server ist der erste Application Server, der den deployten Applikationen eine weitgehend transparente Nutzung all dieser neuen Funktionalitäten bietet.

Dieser Vortrag stellt den Zugriff aus dem WebLogic Server auf Datenbank-Features wie Database Resident Connection Pool, Pluggable Database und Global Data Services vor und beschreibt die Nutzung von Active GridLink for RAC und Application Continuity (JDBC Replay).

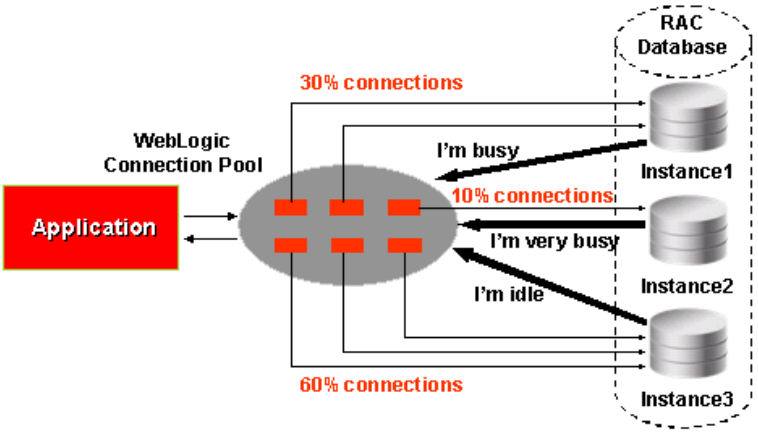
Überblick

Der Zugriff eines Java Clients auf eine SQL Datenbank ist über Standards wie SQL:2003 bzw. SQL:2008 sowie die entsprechenden JDBC Standards eigentlich normiert. Somit ist die Funktionalität unabhängig vom eingesetzten Datenbank Client und Application Server – könnte man denken. In der Tat gibt es aber eine Reihe von Verhaltensweisen, die über keinen Standard geregelt sind, aber Stabilität, Performance sowie Ressourcenverbrauch signifikant beeinflussen können. Insbesondere in den Oracle Datenbank Versionen 11g und 12c finden sich etliche Weiter- und Neuentwicklungen, die dem Programmierer und auch dem Administrator das Leben deutlich erleichtern können.

Active GridLink for RAC (AG4R)

Real Application Cluster (RAC) Datenbanken sollen für eine optimale Lastverteilung von Anwendungszugriffen über mehrere Datenbankrechner hinweg sorgen und bei Ausfall eines Rechners der Anwendung die Verbindung zu einer noch laufenden Cluster-Instanz ermöglichen. In der Vergangenheit funktionierte das, indem der Connection Pool im Application Server auf Fehlermeldungen der Datenbank gelauscht und beim Auftreten von Fehlern dann – reaktiv – seinen Connection Pool auf alternative Cluster Instanzen umgehängt hat. Um zu erfahren, dass eine RAC Instanz nach einem Fehler wieder zu dem Cluster dazu gekommen ist, musste der Connection Pool in festgelegten Intervallen die ihm bekannten Instanzen anfragen (pingen). Gänzliche neue RAC Instanzen konnten ohne Änderungen an der Connection Pool Konfiguration nicht bekannt gemacht werden.

In der Oracle Weblogic Suite sowie im WebLogic Server der Exalogic Elastic Cloud gibt es seit der Version 10.3.4 mit „Active GridLink for RAC“ (AG4R) einen ständigen Nachrichtenkanal zwischen den Datasources des Application Servers und den Oracle Notification Services der RAC Datenbank. Damit werden Performance- und Laufzeitinformationen von der Datenbank an die Connection Pools des Application Servers übermittelt, um die Lastverteilung auf die RAC Instanzen zu optimieren. Im Fehlerfall kann die Datenbank pro-aktiv die betroffenen Datasources informieren. Diese hängen dann rechtzeitig, möglichst bevor eine Anwendung wieder auf eine fehlerhafte Instanz zugreifen will, ihren Connection Pool auf die noch funktionierenden RAC Knoten um. Kommt eine RAC Instanz neu oder wieder zum Datenbank Cluster hinzu, werden die Datasources im Application Server ebenfalls aktiv über die zusätzlichen Ressourcen benachrichtigt.



Active GridLink for RAC sorgt darüber hinaus für Transaktionsaffinität. Dies bedeutet, dass eine Transaktion automatisch auf die RAC Instanz gebunden bleibt, auf der sie gestartet wurde. Und dies unabhängig davon, ob die Transaktion über einen oder mehrere WebLogic Server verteilt wird. Ähnliches gilt für Websession Affinität. Hierbei erfolgen alle Connections einer HTTP Session gegen die selbe RAC Instanz. Basierend auf dieser Affinität kann Active GridLink for RAC die SCAN Adressierung der RAC Datenbank nutzen. Dadurch ist die Konfiguration einer Datasource unabhängig von der Anzahl der RAC Knoten und deren Lokation.

Im Zusammenspiel mit Oracle Data Guard unterstützt Active GridLink for RAC auch Desaster Recovery Szenarien mit beliebig vielen Standby Datenbanken. Wird nach Ausfall der primären Site die Standby Datenbank aktiviert, schwenken die AG4R Connection Pools automatisch auf die neue Site um. Ein Restart der Application Server ist hierfür nicht erforderlich.

Application Continuity

In der Regel muss der Code einer Anwendung auf Fehler der Infrastruktur, in der die Anwendung betrieben wird, reagieren können. Bricht zum Beispiel eine Datenbankverbindung innerhalb einer Transaktion ab, so muss die Anwendung den Anwender entsprechend benachrichtigen. Dieser muss dann, nachdem die Verbindung wieder hergestellt wurde, die noch nicht gespeichert Daten erneut eingeben. Häufig ist dem Anwender jedoch gar nicht klar, welche Daten bereits gespeichert wurden und welche nicht.

Mit Application Continuity baut der JDBC Replay Treiber (eine spezielle Klasse des Oracle JDBC 12c Treibers) eine abgebrochene Datenbank-Session automatisch gegen eine alternative RAC (oder Data Guard) Instanz erneut auf und spielt die angefangene Transaktion bis zum Zeitpunkt des Abbruches nach. In diesem Fall bemerkt der Anwender möglicherweise gar nicht, dass es einen Fehler auf der Datenbank gegeben hat.

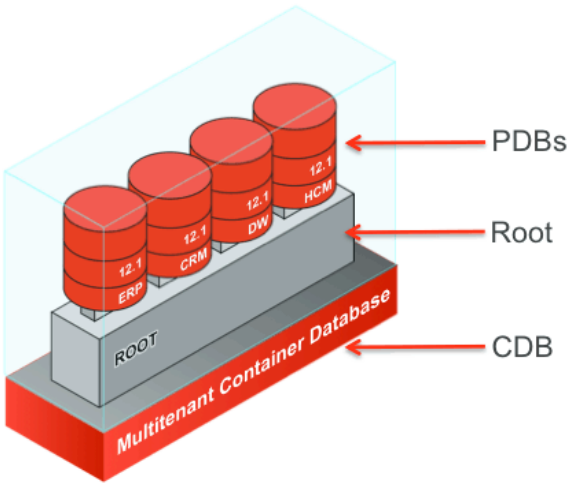
Optional kann der Programmierer einen Callback registrieren. Dieser ermöglicht es der Anwendung, im Falle eines automatischen Replays zuvor die Datenbank-Session identisch so zu initialisieren, wie sie bei der original Transaktion war.

In dieser ersten Version von Application Continuity existieren noch einige Voraussetzungen, die erfüllt sein müssen, damit ein automatischer Replay stattfinden kann. In dem Vortrag werden die notwendigen Bedingungen dargestellt und Beispiele für das Verhalten gezeigt.

Pluggable Databases

Oracle Database 12c führt eine mandantenfähige Datenbank auf der Basis von sogenannten „Pluggable Databases (PDBs)“ ein. Pluggable Databases sind Datenbanken, die innerhalb einer Container Datenbank laufen. Verschiedenste Ressourcen und Data Dictionary Objekte werden zwischen diesen PDBs geteilt, so dass Pluggable Databases in der Regel deutlich weniger Ressourcen verbrauchen, als die gleiche Anzahl an Standalone Datenbanken.

Pluggable Databases können einerseits schlicht für die Konsolidierung mehrerer Datenbanken benutzt werden. Darüber hinaus lassen sich mit ihnen aber auch die für eine Mandantenfähigkeit notwendige strikte Trennung von Ressourcen und Zugriffsrechten realisieren. Der Zugriff auf eine Pluggable Database ist komplett transparent für den Oracle WebLogic Server. Da alle Service Namen aller Pluggable Databases einer Container Database verschieden sein müssen, genügt die korrekte Angabe des Service Namens im Connect String des Connection Pools. Im WebLogic Server muss man also gar nicht wissen, ob es sich um eine Pluggable Database handelt, oder nicht.

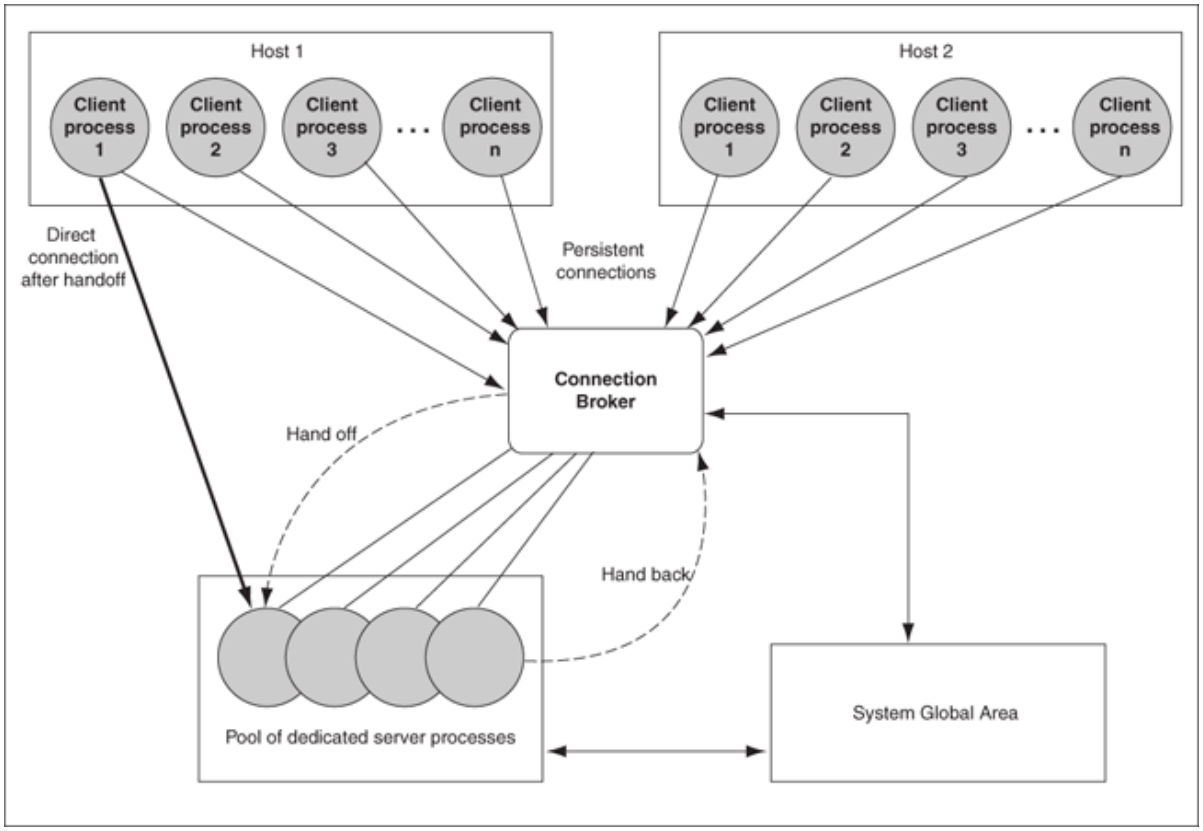


Andererseits möchte man aber in manchen Fällen vielleicht aus einer einzigen Datasource auf unterschiedliche Pluggable Databases zugreifen – z.B. wenn eine mandantenfähige Applikation je nach angemeldetem Mandanten eine andere PDB benutzen soll. Im einfachsten Fall lässt sich dies anwendungsseitig realisieren, in dem der Code nach einem „datasource.getConnection()“ als Erstes das Statement “ALTER SESSION SET CONTAINER = <mandant>“ absetzt und jeder Mandant so seine eigene PDB bekommt. Dieses „Umhängen“ der Verbindung ist für die Datenbank aber relativ aufwändig. Aus diesem Grund kann es sinnvoll sein, über sog. „Connection Labels“ die Verbindungen eines Connection Pools auf unterschiedliche PDBs offen zu halten. Der Connection Label Callback liefert dann die Connection auf die richtige PDB an den getConnection() Request der Applikation zurück – je nachdem, welcher Mandant diesen Request gerade absetzt. Gibt es noch keine Verbindung zu der angeforderten PDB, wird diese neu erstellt, dem Pool zugefügt und beim nächsten Zugriff dieses Mandaten wieder genutzt.

Database Resident Connection Pool

Viele Anwendungen in Application Servern erfordern eine große Anzahl von Datenbank Connections, auch wenn diese so gut wie nie alle gemeinsam genutzt werden. Insbesondere in Application Server Clustern werden oft sehr viele Datenbankverbindungen offen gehalten, da jeder Managed Server für sein Maximum an Connections konfiguriert werden muss, aber so gut wie nie alle Managed Server dieses Maximum *gleichzeitig* ausnutzen. Diese große Zahl an Datenbankverbindungen – und die damit verbundene große Zahl an Dedicated Server Prozessen – führt zu einem unnötig hohen Resourcenverbrauch auf der Datenbank (Anzahl Prozesse, Shared Memory Nutzung, usw.). Database Resident Connection Pools ermöglichen eine bessere, weil ressourcenschonende Nutzung von Datenbank Sessions und Prozessen. Im Gegensatz zum Shared Server Konzept, das bereits seit Oracle

9i existiert, erhält der Client („Client“ meint hier eine Connection aus dem WLS Connection Pool) einen *Dedicated* Server Prozess zugeordnet. Dieser „gehört“ dem Client, bis er diese Connection wieder an den Connection Pool zurück gibt. Danach kann sie nur an *den selben* Datenbank-User weitergegeben werden – allerdings auch an eine andere Applikation. Der selbe Dedicated Server Prozess kann also mehrere Client-Connections (auch aus unterschiedlichen WebLogic Managed Servern unterschiedlicher Rechner) bedienen. Im herkömmlichen Modell (ohne DRCP) ist ein Dedicated Server dagegen „idle“, solange die ihm zugehörige Connection im Connection Pool nicht von Anwendungscode benutzt wird.



Im WebLogic Server 12c lässt sich die Nutzung von Database Resident Connection Pools einfach konfigurieren. Voraussetzung ist nur die Nutzung des JDBC 12c Treibers.

Global Data Services

In großen, global verteilten Unternehmen sind häufig auch die Datenbanken über verschiedene Unternehmensstandorte verteilt. Replikationsmechanismen wie Oracle GoldenGate oder Oracle (Active) Data Guard sorgen für die Synchronisierung der Daten. So haben die Anwender der verschiedenen Unternehmensstandorte Zugriff auf jeweils lokal gehaltene Daten.

Im Fehlerfall wäre es nun aber wünschenswert, dass die Anwendungen automatisch auch ein anderes Replikat der Datenbank verwenden können, selbst wenn die Latenzzeiten durch den Zugriff auf einen entfernten Standort zu möglicherweise schlechteren Antwortzeiten führt. Global Data Services ermöglichen das Einrichten von Active GridLink for RAC Data Sources, die zusätzlich noch eine Information über die Region beinhalten, in der sich der Datenbank Service *möglichst* befinden soll. Im Fehlerfall, wenn die lokale Datenbank nicht (mehr) verfügbar ist, erfolgt der Failover auf einen anderen Standort, basierend auf der Fast Connection Failover Technologie in Active GridLink for RAC.

Dieses Konzept basiert auf der Annahme identischer Datenbanken in den unterschiedlichen Standorten, die über Multi-Master-Replikation synchronisiert werden. Nur so kann auf allen Datenbanken der verschiedenen Standorte nicht nur gelesen sondern die Daten auch modifiziert werden.

Alternativ (allerdings unter Anpassung des Anwendungscodes) ist es möglich, *einen* Datenbankstandort für DML-Operationen *aller* Clients vorzusehen, während reine Leseoperationen jeweils auf den lokalen Instanzen stattfinden. Im Fehlerfall können sowohl der (eine) Service für die schreibende Datenbank als auch eine ausgefallene Read-Only Datenbank auf einen anderen Standort verlagert werden. Der Reconnect der WebLogic Data Sources erfolgt transparent für die Anwendung, ohne dass Konfigurationsänderungen notwendig sind oder ein Restart der Applikationsserver erfolgen muss.

Zusammenfassung

Die Oracle Datenbank implementiert eine Reihe von Funktionalitäten, die es Anwendungsentwicklern erlauben, ausfallsichere und skalierbare Applikationen zu schreiben. Der Oracle WebLogic Server befreit den Entwickler weitestgehend von der Notwendigkeit, diese Oracle spezifischen Features im Code zu implementieren. Dies hält den Anwendungscode portabel und befreit den Entwickler von der Implementierung herstellerspezifischer Anpassungen. Darüber hinaus lassen sich nützliche Erweiterungen in die Anwendungen bringen, die den Resourcenverbrauch bzw. die Performance weiter optimieren.

Kontaktadresse:
Thomas Robert
Oracle Deutschland B.V. & Co. KG
Kühnehöfe 5
D-22761 Hamburg

Telefon:	+49 40 890 91 188
Fax:	+49 40 890 91 188
E-Mail	Thomas.Robert@oracle.com
Internet:	www.oracle.com