Tuning Oracle Web-Applications in WLS12c

Markus Klenke
TEAM GmbH
Paderborn

Schlüsselworte
Oracle ADF, Deployment, Weblogic Server, Performance

Einleitung

Entwickelt man die ersten prototypischen Applikationen mit JSF oder Oracle ADF wird häufig nicht direkt auf die Laufzeit und Performanz der Applikation geschaut. Fehlende Optimierungen fallen meist erst bei größeren Applikationen, höheren Nutzerzahlen oder dem allgemeinen Produktivgang auf. Es kommt nicht in Frage die Applikation komplett neu aufzubauen, daher wird versucht an allen möglichen Stellschrauben zu drehen, um Leistung aus der Anwendung und dem Server herauszukitzeln.
Doch nicht nur die Anwendung kann ein Performance Schwachpunkt in der Systemarchitektur sein. Auch beim Deployment und bei der Serverparametrisierung können selbst kleinste Einstellungen zwischen Sieg und Niederlage entscheiden.
Dieser Vortrag soll verschiedene dieser Stellschrauben an der Applikation, dem Deployment und dem WLS 12c aufzeigen und im Fall einer ADF Applikation Best Practice Tipps mitgeben.

Vom Prototypen zum echten Projekt – der erste Schrecken

Geschäftsapplikationen mit einem neuen Framework oder einer neuen Programmiersprache zu erstellen folgt meistens einem ähnlichen Schema:
Schulungen erhalten, Prototypen bauen, Prototypen einstampfen, eigentliches Projekt beginnen.
Dieser Ansatz klingt nach einem guten Plan, ist er in den meisten Fällen auch. Leider kann ein Prototyp bei weitem nicht alle Anforderungen an das Echtprojekt abbilden. In vielen Fällen wird mit Mock-Daten gearbeitet, höchstens mit einer oder zwei Personen getestet oder mal eben schnell die erste Lösung für ein Problem gewählt. Beginnt dann die eigentliche Implementierung des Projektes, treten bei erstem Lastverhalten die Schweißperlen auf der Stirn der Entwickler/Administratoren zum Vorschein. Die Applikation ist auf einmal sehr langsam, bei 50 Benutzern bricht der Server zusammen, Benutzer sperren sich gegenseitig Datensätze etc. Woran kann das liegen und wo kann man nach diesen Problemfällen schauen. Tendenziell gibt es wenigstens drei Bereiche für potenzielle Applikationsschwachpunkte: Die Applikation selbst, das Deployment und den Web-Server. Zusätzlich gibt es diverse Datenbank-Optimierungen, welche dieser Beitrag allerdings nicht abdeckt.

Am Anfang war die Webapplikation

Das Oracle Application Development Framework, kurz ADF, ist das zentrale Java Framework von Oracle zur Erstellung von Enterprise Applikationen. Technisch gesehen bietet es diverse Interfaces, welche beliebige Datenquellen über eine Abstraktionsschicht an eine Model-View-Controller-Muster getriebene Web-Applikation bindet. Durch diese Muster-Implementierung ergeben sich innerhalb der ADF Applikation mehrere Schichten, welche für eine Performance Analyse begutachtet werden müssen. Eine Übersicht über die verschiedenen Schichten ist in Abbildung 1 dargestellt.

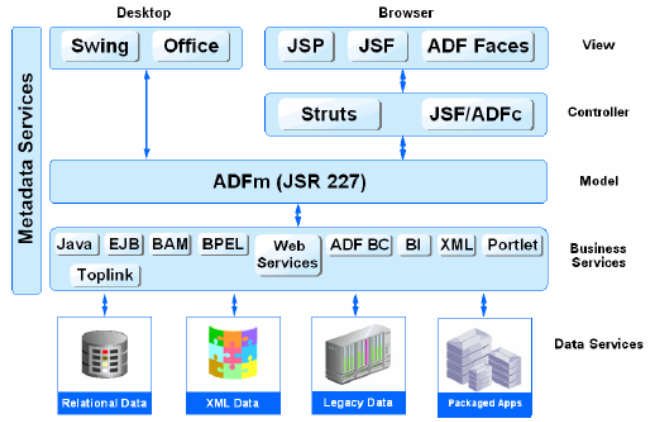

Abb. 1: Oracle ADF Architektur

Da die Datenanbindung nicht Teil dieses Vortrages ist, befassen wir uns zunächst mit der Business Services Schicht. Diese bietet eine Transaktionsgesteuerte Cache Plattform und stellt die Daten aus diversen Data Services durch Java Objekte für weitere schichten oder Anwendungen zur Verfügung. Technisch gesehen gibt es in der Oracle Standardimplementierung dieser Schicht drei wichtige Elemente mit unterschiedlichen Funktionalitäten: Entity Objekte zum Vorhalten von Daten aus der Datenschicht, View Objekte zum akquirieren von Entity Objekten und Application Modules zur Anbindung der vorherigen Elemente an die Oberfläche. Den Größten Gewinn von Performance kann man bei den letzten Beiden Komponenten erreichen.

View Objekte verhalten sich in Ihrer Funktionalität ähnlich wie Datenbank Views oder Named Queries aus der EJB Welt; sie kapseln Anfragen an eine Datenbank und führen diese zur Laufzeit aus. View Objekt können bei Anbindung an eine Oracle Datenbank Optimizer Hints genau so benutzen, wie die Datenbank (Beispiel: Möglichst schnelle Rückgabe der ersten Zeile). In vielen Fällen ist es absolut unnötig, alle Werte aus der Rückgabemenge sofort zu laden, da entweder eine Formulardarstellung oder eine Tabellendarstellung mit weniger als n Zeilen auf der Oberfläche vorliegt. Um nicht nur die Datenbank Query sondern auch das Instanziierungsverhalten der Objekte zu optimieren bietet ADF an den View Objekten weitere Einstellungen. So ist es zum Beispiel möglich, ein View Objekt rein für die Erstellung eines Datensatzes zu benutzen. Die Query dient daher nur der Information, wo dieser Satz später in der Datenbank abgespeichert wird.

Application Modules sind die Services der Business Components, welche die Datenabfragen und Rückgabewerte an die Außenwelt propagieren. Diese Objekte organisieren zusätzlich die Instanziierung von View Objekten. Im Normalfall wird für jeden Anwendungsbenutzer mindestens ein Application Module erzeugt, um für die Nutzungszeit der Applikation eine Verbindung mit der Datenbank aufrecht zu erhalten. Es gibt allerdings View Objekte, bei denen es wenig Sinn macht, sie für jeden Benutzer neu zu instanziieren. Ein Beispiel hierfür ist ein View Objekt, was beispielsweise Übersetzungstexte aus der Datenbank liest. Diese Texte ändern sich für gewöhnlich nicht zwischen den Release-Zyklen. Um diese Objekte als Singleton zu erstellen, kann das Application Module als „Shared Application Module“ deklariert werden. Dieser Modus sorgt dafür, dass nur ein einziges Modul von diesem Typ erstellt wird und von allen Benutzern gleichermaßen verwendet. (Man sollte dazu sagen, dass die Query des View Objektes zur Laufzeit für jeden Benutzer individuell verändert werden kann, z.B. für Sprachpräferenzen). Das sorgt dafür, dass die abgefragten Elemente auch nur ein einziges Mal gecached werden, was gerade bei Anwendungen mit vielen Benutzern einen enormen Speichervorteil mit sich bringt. Zusätzlich wird die Geschwindigkeit der Anwendung gesteigert, da bei schon gecacheten Objekten die Query auf der Middleware ausgeführt wird und nicht auf der Datenbank.

Sind die Daten bereitgestellt, werden diese typischerweise direkt an die Oberfläche angebunden. Wie die Applikation mit den Daten umgeht und wie Benutzereingaben interpretiert werden, wird typischerweise über Managed Beans gesteuert. Dies sind Java Klassen, welche einen bestimmten Lebenszyklus (Scope) vom Framework vorgegeben bekommen.

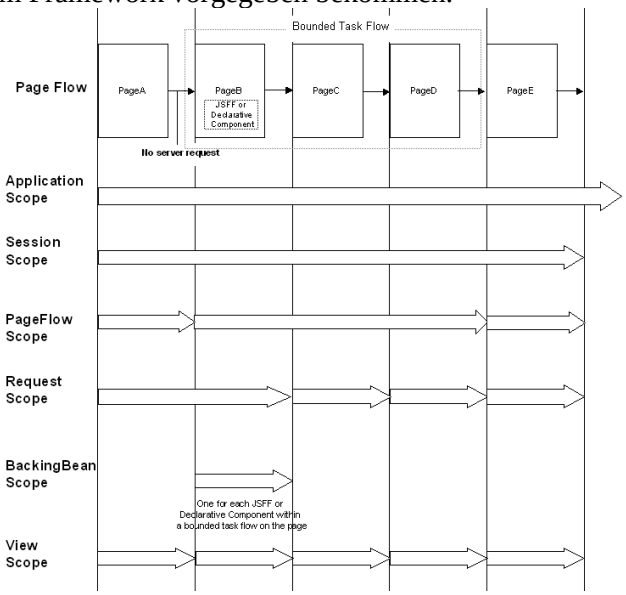

Abb. 2: Managed Bean Scopes in Oracle ADF

Je nach Anforderung ist es notwendig, einen möglichst kleinen oder großen Zyklus auszuwählen, um die Performance der Applikation hoch zu halten. Auch hier kann gerade in Anwendungen, welche von unerfahrenen Entwicklern erstellt werden, ein großer Performance Boost gefunden werden.Ist eine Bean in einem kurzen Scope angesiedelt, so wird diese entsprechend häufig aufgeräumt und neu instanziiert. Eine Bean in einem Scope über alle Sessions hinweg würde hingegen ein einziges Mal beim Starten der Applikation instanziiert werden und dann für alle Benutzer der Applikation gültig sein.
Auf diesem Level der Applikation finden sich meistens zwei Extrema, welche zu einer Performance-Verschlechterung oder einer Systeminstabilität führen können. So kommt es immer mal vor, dass Beans mit einem hohen Kostenfaktor (lange Instanziierung, komplexe Berechnung, etc.) in einem zu kleinen Scope liegen, oder anders herum Java Objekte mit einer großen Datenmenge für eine temporäre Berechnung im einem zu großen Scope. Der Vortrag zeigt zwei Beispiele für schlecht gewählte Managed Bean Scopes und zeigt, wie einfach die Performance der Anwendung verbessert werden kann, indem Beans in den passenden Scope angebunden werden.

Alle angesprochenen Punkte werden in einer (zu Darstellungszwecken übermäßig) unperformanten Anwendung dargestellt und mit Argumenten begründet optimiert, so dass die Anwendung am Ende der Demo einen gewaltigen Performance boost erhält.

Von der Entwicklung zur Produktion – Deployment Pläne
Ist die Applikation selbst so gut es geht optimiert, können Änderungen an Applikationsparametern zusätzlichen Performancegewinn einbringen. Im Entwicklungsmodus werden die Oberflächenkomponenten von ADF meist unkomprimiert gerendert, was für Oberflächentests auf Style Klassen Basis notwendig ist. Für den Produktiveinsatz ist dieser Modus allerdings vollkommen unnötig und verlangsamt nur die Renderzeit der Seite. Muss man jetzt für den Entwicklungsmodus und den Produktivmodus zwei Applikationen simultan entwickeln? Glücklicherweise nicht, es besteht die Möglichkeit einzelne XML Strukturen der Applikation während des Deployments mit Hilfe eines

Deployment Plans auf eine Umgebung anzupassen. So kann beispielsweise der Parameter zur Kompression der Oberflächenkomponenten zu diesem Zeitpunkt auf true gestellt werden, damit auf der Produktion die Kompression der Klassennamen durchgeführt wird. Es können insbesondere auch Parameter verändert werden, die das Applikationsverhalten alternieren. So könnte im Produktionsmodus eine Abfrage nicht mehr gegen eine Datenbank, sondern gegen einen parallel gestarteten WebService gesteuert werden, um Performance zu gewinnen. Sollte genug Zeit vorhanden sein, wird ein Deployment Plan exemplarisch gezeigt und es wird dargestellt, wozu dieses von Haus aus mitgebrachte Feature vom Weblogic Server fähig ist.

Optimierung auf dem Server – JVM und Connection Pools

Auf dem Server angekommen gibt es immer noch diverse Möglichkeiten die Applikationsperformance zu beeinflussen. Primärelemente dafür sind die Konfiguration der JVM des Servers und die Konfiguration der Datenquell-Pools. Auf Grund des beschränkten Platzes, werden in diesem Beitrag nur die JVM Performance Optimierungen angesprochen. Bei den Einstellungen der Java virtuellen Maschine scheiden sich die Geister. Daher sind die folgenden Einstellungen den Erfahrungen aus der ADF Entwicklung entnommen, und sollten nicht als Allheilwerkzeug für die Entwicklung jeglicher Webapplikationen gesehen werden. Wie im vorherigen beschrieben muss die JVM großteilig mit Objekten aus den Managed Beans interagieren. Daher sollten die Speicherkonfigurationsparameter auf genau dieses Szenario optimiert werden. Generell benötigen Starts der Applikation mit einem noch nicht aktiven Benutzer einen relativ großen Anteil an bootstrap Speicher. Das heißt, bei einer Neuanmeldung wird der größte Teil des gesamten vom Benutzer benötigten Speichers sofort benötigt. Es macht also nicht wirklich Sinn die initiale Heap-Size deutlich kleiner zu halten als die maximale Heap-Size. Als Best Practice für ADF Applikationen hat sich ergeben, Die JVM Parameter Xms (initialer Heap Size) = Xmx (maximaler Heap Size) zu setzen, um das Nachladen von Speicherblöcken auf dem Arbeitsspeicher des Servers zu vermeiden.
Ein weiteres wichtiges Thema im Bereich Speicherverwaltung ist die Konfiguration der Garbage Collection (GC). Für ADF sind insbesondere die Parameter XX:NewRatio und XX:SurvivorRatio interessant, da sich diese mit den jeweiligen Teilbereichen des permanenten Heap auseinandersetzen, der Young Generation und der Old Generation (Siehe Abbildung 4).

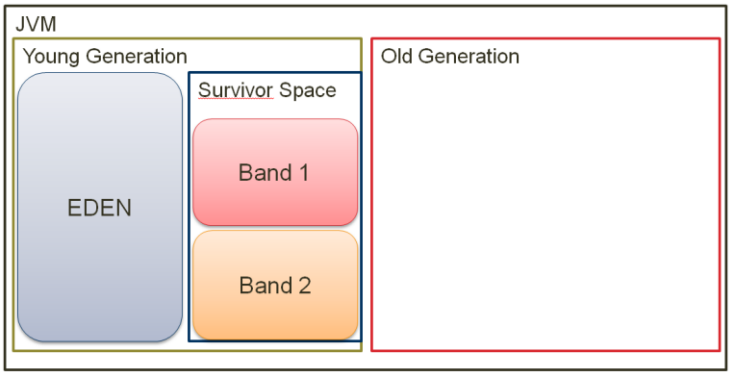

Abb. 3: Schematische Darstellung einer Java VM

Welche Speichereinstellungen sich bezogen auf ADF Anwendungen besonders eignen und welche man besser auslassen sollte, wird ebenfalls Teil der Demo sein.

Ein weiterer Bereich der bezüglich Performance sehr gut im Auge behalten warden sollte, sind die JDBC Connection Pools auf dem Application Server. Diese Erstellen nun den Kreisschluss in diesem Vortrag, da der Server sich um die erstellten Datenbank Verbindungen kümmert. Es kann also theoretisch sein, dass sich zwar eine sehr performante Anwendung und ein gutes Speichermanagement

auf dem Server befindet, das neue Bottleneck allerdings die Anbindung an die Datenbank selbst ist. In vielen Produktivsystemen wird durch ungünstig gesetzte Parameter im JDBC Pool ein enormer Traffic durch Instanziierung und Deinstanziierung von Datenbank-Verbindungen erzeugt, welcher die Anwendungen konstant verlangsamt. In einem noch schlimmeren Fall kann es sogar dazu kommen, dass die Anwendung auf Grund von nicht Verfügbarkeit von Datenbank Verbindungen für einzelne Anwender nicht benutzbar ist. Auch dieser Punkt soll abschließend in der Demo noch einmal veranschaulicht werden.

Kontaktadresse:

Markus Klenke
TEAM GmbH
Hermann-Löns Str. 88
33104 Paderborn

Telefon: +49 (0) 5254 – 8008/55
E-Mail mke@team-pb.de
Internet: www.team-pb.de