Tuning the Mobile Server

Philipp Loer
ORDIX AG
Paderborn

Schlüsselworte

Oracle Database Mobile Server 11g, Performance-Tuning, Partitionierung

Einleitung

In diesem Vortrag werden einige Tuning-Möglichkeiten des Oracle Database Mobile Server 11g präsentiert.

Der Oracle Database Mobile Server 11g ermöglicht eine Replikation von Materialized Views auf verschiedene mobile Geräte (wie z.B. Smartphones). Hierdurch können diese im Falle einer fehlenden Netzwerkverbindung zum Datenbankserver auf ihrer eigenen, lokalen Datenbank weiterarbeiten. Wird die Netzwerkverbindung wiederhergestellt, so muss ein Datenabgleich zwischen beiden Datenbanken stattfinden. Hierbei werden die zwischenzeitlich auf dem Server geänderten Daten auf den Client repliziert. Umgekehrt repliziert aber auch der Client die auf ihm geänderten Daten auf den Server.

In diesem Vortrag werden zunächst kurz die Funktionsweise des Mobile Server und insbesondere der Ablauf eines Datenabgleichs beschrieben. Anschließend werden verschiedene Möglichkeiten aufgezeigt, wie diese asynchrone Replikation beschleunigt werden kann. Es wird vorgestellt, welcher Performance-Gewinn u.a. durch den Einsatz von Shared Maps, Partitionierung oder Denormalisierung erreicht werden kann.

Tuning the Mobile Server

„I love it when a plan comes together."

- Colonel John „Hannibal" Smith, „The A-Team"-

Eine Eigenschaft guter Software ist, dass sie korrekt funktioniert. Aber eine korrekt funktionierende Software mit ungenügender Performance wird von Kunden und Anwendern nicht akzeptiert. Einige der beschriebenen Optimierungsmöglichkeiten sind in bestehenden Softwareumgebungen nur unter hohem Aufwand umzusetzen. Andere verringern die Zeit des Datenabgleichs, sorgen aber für eine erhöhte Last auf der Datenbank. Daher ist es wichtig, bereits im Vorfeld eines Softwareentwicklungsprojektes den Performance-Gesichtspunkten die erforderliche Aufmerksamkeit zu schenken. Dazu ist es erforderlich sowohl die benötigte als auch die erreichbare Performance so früh wie möglich zu planen.

Der Vortrag beschränkt sich auf die Performance-Analyse des Oracle Database Mobile Server. Im besonderen Fokus steht die Untersuchung der Funktionsweise des Mobile Server sowie der Operationen in den Datenbanken auf Client- und Server-Seite. Eine weitergehende Analyse des Performance-Einflusses von Systemkomponenten, wie z.B. der Leistung einzelner Hardwarekomponenten ist ausdrücklich nicht Teil dieses Vortrags.

Der Aufbau des mobile Server

Die Architektur des Oracle Database Mobile Server besteht aus drei Schichten: Der Client-Schicht, der Server-Datenbank und dem Mobile Server in der mittleren Schicht.

Server-Datenbank: Die Server-Datenbank ist eine normale Oracle Datenbank. Im Gegensatz zu anderen Anbietern mobiler Datenbanksysteme (wie z.B. IBM oder Sybase) ist diese Datenbank zwingend eine Datenbank in der Standard oder Enterprise Edition aus dem Hause Oracle.

Mobile Server: Die zweite Systemkomponente ist der Mobile Server. Diese Systemkomponente organisiert die Replikation und Synchronisation der Daten zwischen den einzelnen Clients und der Server-Datenbank.

Mobile Client: Der Mobile Client ist das Pendant des Mobile Server auf Client-Seite. Aufgabe des Mobile Client ist es, eine Verbindung zum Mobile Server aufzubauen. Darüber hinaus beinhaltet er die folgenden Teilkomponenten:

- Sync Engine: Die erste Teilkomponente des Mobile Client ist die Sync Engine. Ihre Aufgabe ist es, die seit der letzten Synchronisation in der Client-Datenbank gemachten Änderungen an den Mobile Server zu senden. Darüber hinaus nimmt sie die Änderungen in der Server-Datenbank entgegen und führt diese in der Client-Datenbank durch.

- Device Management Agent: Aufgabe des Device Management Agent ist es, die Datensynchronisation zu starten. Dies kann aufgrund einer Anforderung aus der Clientanwendung geschehen. Darüber hinaus kann der Device Management Agent aber auch selbstständig eine Synchronisation starten, z.B. falls der Client gerade gestartet wurde, eine Netzwerkverbindung zur Verfügung steht oder eine bestimmte Anzahl an Datensätzen auf Client-Seite geändert bzw. hinzugefügt wurde. Zweite Aufgabe des Device Management Agent ist es, eine durch den Mobile Server initiierte Softwareaktualisierung der Mobile Application durchzuführen.

- Mobile Application: Die Mobile Application ist die Software auf Client-Seite. Sie greift auf die Daten der Client-Datenbank zu und kann darüber hinaus eine Synchronisation mit der Server-Datenbank durch die Sync Engine anfordern.

- Client-Datenbank: Auf der Seite des Client kann entweder sqlite oder Berkeley DB gewählt werden. Im Gegensatz zu Berkeley DB ist die Nutzung von sqlite kostenlos. Für sqlite ist jedoch über die offizielle Oracle-Dokumentation keine weitere Unterstützung seitens der Oracle Corporation verfügbar.

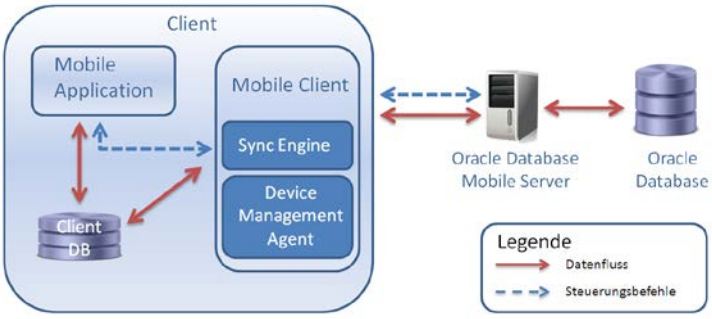

Abb. 1: Architektur

Tuning-Beispiel: Partitionierung der Mapping-Tabelle

In diesem Vortrag werden mehrere Methoden der Performance-Optimierung vorgestellt. Für einen ersten Eindruck erfolgt hier nun beispielhaft die Beschreibung einer ausgewählten Optimierungsmöglichkeit.

Bei einem Datenabgleich schickt der Mobile Server alle zwischenzeitlich geänderten Datensätze an den Client. Um dies zu ermöglichen wird für jeden Datensatz eine Versionsnummer gespeichert. Diese wird bei jeder Änderung des Datensatzes inkrementiert. Zusätzlich speichert der Mobile Server in einer sogenannten Mapping-Tabelle, welche Version des Datensatzes zuletzt an den Client geschickt wurde. Durch einen Vergleich dieser beiden Versionsnummern ist es dem Mobile Server möglich zu ermitteln, welche Datensätze an den Client zu schicken sind. Nimmt man beispielsweise an, dass die abzugleichende Tabelle 100.000 Datensätze enthält, und dass diese mit 1.000 Clients abgeglichen werden muss, so enthält die entsprechende Mapping-Tabelle 1.000 x 100.000 = 100 Mio. Datensätze.

Im Folgenden wird gezeigt, welcher Performance-Gewinn durch eine Partitionierung der Mapping-Tabelle erreicht werden kann.

Technischer Hintergrund

Durch Partitionierung kann eine Tabelle horizontal geteilt werden. Die Partitionen enthalten jeweils einen Teil der Datensätze und sind durch einen einheitlichen Tabellennamen ansprechbar. Hierdurch ist es möglich, eine bestehende Tabelle ohne Änderung einer Applikation zu partitionieren. Die Abbildung 2 verdeutlicht dieses Prinzip. Jeder Datensatz wird hierbei einer bestimmten Partition zugewiesen. Über die bei der Anlage der Tabelle anzugebenden Schlüsselspalten werden die Datensätze je nach gewähltem Partitionierungsverfahren einer bestimmten Partition zugewiesen. Datensätze mit identischen Schlüsselwerten befinden sich immer in der gleichen Partition. Oracle-Datenbanken unterstützen die folgenden Partitionierungsverfahren:

- List-Partitionierung: Bei der List-Partitionierung muss für jeden Schlüsselwert angegeben werden, welcher Partition der Datensatz zugewiesen werden soll.
- Range-Partitionierung: Die Range-Partitionierung ermöglicht es, statt eines oder mehrerer expliziter Schlüsselwerte einen Wertebereich anzugeben. Realisiert wird dies durch die Definition einer Obergrenze für jede Partition. Die Untergrenze für eine Partition ergibt sich durch die Obergrenze der angrenzenden Partition.
- System-Partitionierung: Die System-Partitionierung ist seit Oracle 11g Release 1 verfügbar. Sie ermöglicht der Applikation, den Datensatz einer bestimmten Partition zuzuweisen.
- Hash-Partitionierung: Bei der Hash-Partitionierung wird der Datensatz durch Verwendung eines Hash-Algorithmus zugewiesen. Über den Hash-Wert des Schlüsselattributs kann hier die zugehörige Partition ermittelt werden.

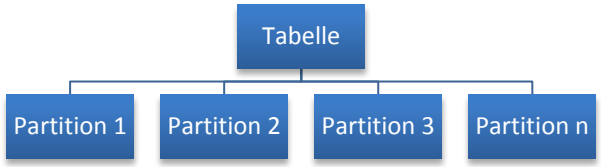

Abb. 2: Partitionierung

Begründung

Der Mobile Server ermittelt während eines Datenabgleichs die an den Client zu sendenden Datensätze durch einen Vergleich der Versionsnummern. Eine Partitionierung des Mapping-Tabelle würde folgende Vorteile bringen:

- Verbesserter Ausführungsplan: Die Partitionierung ermöglicht einen Full Partition Scan
- Bessere Blockfüllung: Bei einer nach der Client-ID partitionierten Mapping-Tabelle würden die Datensätze mit gleicher Client-ID auf weniger Datenbankblöcke verteilt werden. Daher müssten weniger Blöcke gelesen werden und die Größe des benötigten Arbeitsspeichers würde reduziert.

Bei einem Datenabgleich werden alle Datensätze mit einer bestimmten Client-ID benötigt. Daher sollte die Client-ID als Partitionierungsschlüssel gewählt werden. Da diese über die GUI des Mobile Server frei vergeben werden kann, ist eine Verwendung der List- oder Range-Partitionierung nicht zu empfehlen. Diese müsste für jede neue Client-ID angepasst werden. Um die oben genannten Vorteile zu erreichen und den Wartungsaufwand bei neuen Client-IDs zu reduzieren, ist eine Hash-Partitionierung folglich die geeignetste.

Test

Um die Wirksamkeit dieses Optimierungsansatzes zu beweisen dient der folgende Test: Es wurde eine Tabelle mit 10.000 Datensätzen angelegt. Diese Tabelle wurde anschließend auf 1.000 Clients repliziert.

Die Mapping-Tabelle hat somit eine Größe von 1.000 x 10.000 = 10 Mio. Datensätzen. Anschließend werden sechs Tests durchgeführt. Es wird jeweils die auf dem Server geänderte Datenmenge variiert und die Mapping-Tabelle unpartitioniert sowie mit 256 und 512 Hash-Partitionen getestet.

Testergebnisse

Die Untersuchung brachte das in der Tabelle dargestellte Ergebnis. Hierbei ist zu beachten, dass die Zeit für die Abfrage der Mapping-Tabelle des Server für diese Anfrage von knapp über zweitausend Millisekunden um ca. 95 Prozent auf ca. 80 bis 100 Millisekunden deutlich reduziert werden konnte. Diese Zeitersparnis schlug sich auch im gesamten Datenabgleich nieder. Hier konnte im Durchschnitt deutlich über eine Sekunde eingespart werden.

Nr.	Geänderte Datensätze	Partitionierung	Mapping-Abfrage	Gesamtzeit
1	0	Keine	2049 ms	4,8 s
2	0	256 Hash-Partitionen	179 ms	3,3 s
3	0	512 Hash-Partitionen	177 ms	3,1 s
4	5000	Keine	2076 ms	17,4 s
5	5000	256 Hash-Partitionen	205 ms	15,9 s
6	5000	512 Hash-Partitionen	207 ms	16,1 s

Der vorliegende Test hat bewiesen, dass durch eine Partitionierung der Mapping-Tabelle eine Performance-Steigerung erreicht werden kann. Weitere Tests ergaben jedoch, dass ein messbarer Performance-Gewinn erst ab der hier dargestellten Größe von 10 Millionen Zeilen erreicht wird. Für kleinere Mapping-Tabellen sollte daher auf eine Partitionierung verzichtet werden.

In den Tests eins und vier waren die 10.000 Mapping-Datensätze eines Client auf 10.000 Blöcke verteilt. Hier musste der Optimizer einen Full Table Scan auf der Mapping-Tabelle durchführen. In den anderen Testfällen lagen die Datensätze der Mapping-Tabelle eines Client in einer Partition. Der Optimizer kann daher statt eines Full Table Scan einen Full Partition Scan wählen. Hierdurch konnte die Anzahl der gelesenen Blöcke von ursprünglich 35.902 auf nur noch 291 Blöcke reduziert werden. Die Ausführungszeit sank von 2,01 Sekunden auf nur noch 0,02 Sekunden.

Für die Partitionierung wird aus lizenzrechtlichen Gründen als Datenbankmanagementsystem die Enterprise Edition mit Partitionierungsoption benötigt, wodurch im Gegensatz zu anderen Methoden dieses Vortrags zusätzliche Kosten entstehen können.

Kontaktadresse:

Philipp Loer

ORDIX AG

Westernmauer 12-16

D-33098 Paderborn

Telefon:	+49 (0) 5251-1063-0
Fax:	+49 (0) 1805 673490
E-Mail:	info@ordix.de
Internet:	www.ordix.de