IDS 10.0 und Oracle 10g im Vergleich

von Jacques Roy

Wie können wir eine Datenbank auswählen, die Unternehmen einen Wettbewerbsvorteil und den größtmöglichen Return-On-Investment bietet?

In dem vorliegenden Artikel werden die technischen Merkmale, die eine gute OLTP-Datenbank ausmachen, sowie die Gründe für die Überlegenheit von IBM-Informix Dynamic Server Version 10.0 gegenüber Oracle 10g Release 2 behandelt. Eine der großen Herausforderungen dabei ist, für diesen Vergleich technische Informationen heranzuziehen, ohne sich in Details zu verzetteln. Ich versuche, diese Gefahr zu umgehen, indem ich die technischen Details zur Analyse der übergeordneten Anforderungen verwende. Dazu muss ich jedoch im Folgenden einige mit diesem Thema zusammenhängende Konzepte aufführen und erläutern.

OLTP-Anforderungen

Mit einem OLTP-System (OLTP – Online Transaction Processing) werden die tagtäglichen Operationen eines Unternehmens durchgeführt. In großen Unternehmen muss dieses System ein große Anzahl von Verbindungen unterstützen, z. B. zu Points-of-Sale oder Anwendern. Dadurch kann die Anzahl der Transaktionen sehr groß werden.

Viele dieser Systeme arbeiten in Echtzeit, so dass der Ausfall bzw. die Nichtverfügbarkeit einen Umsatzausfall zur Folge haben kann.
Es lassen sich folgende Anforderungen an OLTP-Systeme definieren:

Leistung: Das Datenbanksystem muss die Ressourcennutzung optimieren, um auf der Maschine, auf der es betrieben wird, die maximale Leistung zu erzielen.

Skalierbarkeit: Das System muss in der Lage sein, viele Anforderungen gleichzeitig sowie eine zunehmende Datenmenge zu verarbeiten. Die Arbeitsmenge, die bewältigt werden kann, muss proportional zu den verfügbaren Ressourcen (CPU, Hauptspeicher etc.) sein. Dies kann auch als Teil der Leistungsanforderungen angesehen werden.

Zuverlässigkeit: Das System muss verlässlich arbeiten und ununterbrochen einsatzfähig sein.

Verfügbarkeit: Das System muss rund um die Uhr verfügbar sein. Das bedeutet, dass geplante oder ungeplante Ausfälle minimiert werden müssen.

Verwaltbarkeit: Das System muss sich unkompliziert verwalten lassen. Der Personaleinsatz, den Sie für die Pflege des Datenbanksystems aufwenden, hat direkte Auswirkungen auf die Gesamtbetriebskosten.

Anwendungsentwicklung: Wie einfach ist die Entwicklung von Anwendungen für Ihre Datenbank? Wie standardisiert ist sie? Wie können Sie die Datenbank an Ihr Lösungsdesign anpassen, anstatt mit Rücksicht auf die Datenbank Abstriche an Ihrer Lösung vorzunehmen? Dies sind einige der Fragen, auf die wir eine Antwort finden müssen.

Leistung und Skalierbarkeit

Der Abschnitt, in dem über die Leistung gesprochen wird, ist der längste in dem ganzen Artikel, da für das Verständnis der Faktoren, die die Leistung auf Hardware-, Betriebssystem- und Datenbankserverebene beeinflussen, wichtige Hintergrundinformationen erforderlich sind. Selbstverständlich hat die Konzeption der Anwendung, die die auszuführenden Aktionen bestimmt, enormen Einfluss auf die Leistung. Softwaredesign wird jedoch im vorliegenden Artikel nicht besprochen.

Hintergrundinformationen zur Leistung

Lassen Sie uns vor der Besprechung der Datenbankleistung einen Blick auf die Einflussgrößen der Leistung werfen: die Merkmale der Komponenten eines Computersystems.

Wir übersehen gern, dass die Leistung eines Computers nicht allein durch die Taktfrequenz der CPU bestimmt wird. Wenn wir besser verstehen wollen, welche Faktoren sich auf die Systemleistung auswirken, müssen wir die wichtigsten Komponenten betrachten, nämlich CPU, Hauptspeicher, Systembus, Plattencontroller, Platte und Netz.

Nehmen wir ein aktuelles System wie den IBM System p5 570 (p570), der auf 16 Prozessoren, 512 GB Hauptspeicher und 79,2 TB Plattenspeicher skaliert werden kann:

CPU: Die maximale CPU-Geschwindigkeit beträgt derzeit 1,9 GHz. Zu jedem Kern gehören 2 CPUs mit jeweils 32 KB L1-Cache, 1,9 MB L2-Cache sowie 36 MB L3-Cache. Die maximale Bandbreite zwischen L2 und L3 beläuft sich auf 48 GB/Sek.

Hauptspeicher: Der p570 bietet eine Auswahl an Speichermodulen. Bestimmte CPU-Zyklen müssen zur Adressierung des Hardwarespeichers verwendet werden, was einen drastischen Abfall der Höchstleistung von 10,6 GB/Sek. bei DDR1-Modulen bzw. 24,98 GB/Sek. bei DDR2-Modulen zur Folge hat.

PCI-X-Bus: Bei allen Erweiterungen, beispielsweise zusätzlichen Plattencontrollern, muss das System den 133 MHz-/64-Bit-Bus verwenden, der einen Durchsatz von 1 GB/Sek. hat.

Platten: Die Geschwindigkeit der Plattenlaufwerke hängt davon ab, wie viel Zeit die Positionierung der Schreib-/Leseköpfe über dem gewünschten Zylinder (Suchzeit) und das Warten auf die Anzeige der Daten unter den Köpfen (Rotationslatenz) in Anspruch nimmt. Das schnellste verfügbare Laufwerk des p570 weist 15.000 Umdrehungen pro Minute (U/min.) auf. Die Rotationslatenz beträgt eine halbe Drehung, was eine Verzögerung von 0,002 Sekunden bedeutet. In dieser Zeit kann die genannte CPU 3,8 Millionen Anweisungen ausführen.

Netz: Der p570 unterstützt Netzcontroller, die mit einer Geschwindigkeit von 10/100/1000 Megabit pro Sekunde arbeiten. Ein Byte enthält 8 Bit. Wir können die maximale Übertragungsrate von 1000 Megabit pro Sekunde näherungsweise in 100 MB/Sek. umrechnen. Zudem müssen der System-Overhead für Netzpakete sowie mögliche Kollisionen in Rechnung gestellt werden, was die von diesem Gerät zu erwartende effektive Bandbreite weiter verringert.

Diese knappe Übersicht über die Merkmale von Computerkomponenten zeigt, dass Leistung mehrere Aspekte hat. Wir müssen die Verzögerungen beim Zugriff auf den Hauptspeicher begrenzen und dadurch die CPU-Nutzung optimieren. Die Hardware erleichtert dies mit den in den Power5-Prozessor integrierten drei Cache-Ebenen. Selbst Plattenlaufwerke sind mit Caches ausgestattet, um die Auswirkungen von Suchzeit und Rotationslatenz so gering wie möglich zu halten. Ein Warten der CPU auf den Plattenzugriff wirkt sich fatal auf die Leistung aus.

Schon lange werden Betriebssysteme zur Optimierung der Hardwarenutzung eingesetzt. Interessanterweise können viele der Optimierungen, die sich in Betriebssystemen finden lassen, auch auf Datenbankserver angewendet werden.

Betriebssysteme

Betriebssysteme haben verschiedene Aufgaben, unter anderem die Bereitstellung einer High-Level-Schnittstelle für die Hardware, die gemeinsame Nutzung von Ressourcen (CPU, Platte, Hauptspeicher etc.) durch mehrere Anwender, den Schutz der Anwender voreinander und die Optimierung der Ressourcennutzung.

Damit ein Anwenderprogramm auf einem Computer ausgeführt werden kann, muss es in den Hauptspeicher geladen werden, so dass die CPU auf die Anweisungen und Daten zugreifen kann. Ein Programm nutzt den physischen Speicher nicht direkt. Das Betriebssystem nutzt das Konzept des virtuellen Speichers, den es anschließend den physischen Speicherseiten zur Ausführung zuordnet. Der Programmzugriff auf andere Ressourcen wie Platten und Netze erfolgt mit Hilfe von Systemaufrufen.

Zur zeitlichen Planung der Ausführung von Anwenderprogrammen ordnet das Betriebssystem den einzelnen Programmen nach einem priorisierten Umlaufverfahren bestimmte Zeitfenster zu. So kann kein Programm die Maschine für sich allein nutzen, da das Betriebssystem die Möglichkeit hat, das Heft wieder selbst in die Hand zu nehmen und ein anderes Programm einzuplanen.

Schauen wir uns mit Blick auf die Leistungsoptimierung drei Funktionen von Betriebssystemen an: Dateisysteme, Threads und Prozessoraffinität.

Ein Dateisystem optimiert den Plattenzugriff mit Hilfe einer blockorientierten Ein-/Ausgabe und der Pufferung des Ergebnisses im Hauptspeicher. Die blockorientierte Ein-/Ausgabe verringert den Preis, der bei jedem Plattenzugriff zu zahlen ist: Suchzeit zwischen den Zylindern und Rotationslatenz. Das Unix-System beispielsweise verwendete ursprünglich eine Blockgröße von 1 KB. Später wurde die BSD-Gruppe mit dem schnellen Dateisystem FFS entwickelt. Dieses zeichnete sich vor allem durch die Verwendung einer (konfigurierbaren) Mindestblockgröße von 4 KB aus. Außerdem begrenzte es die Suchzeit dank des Konzepts der Zylindergruppe, bei dem die Verwaltungsinformationen zu einem Dateisystem auf jedem Zylinder und nicht an einer zentralen Position gespeichert wurden.

Dies bestätigt meine obige Aussage, dass es sich nicht empfiehlt, die CPU auf den Plattenzugriff warten zu lassen. Dateisysteme gehen noch einen Schritt weiter und stellen die Platteninformationen in den Cache. Dadurch werden zwei wichtige Ziele erreicht. Das Betriebssystem kann die Informationen in asynchroner Weise zurück auf die Platte schreiben, während die CPU andere Aufgaben ausführt, und die Anzahl der

E-/A-Vorgänge auf den Platten nimmt ab, da die Informationen bereitgestellt werden, ohne dass ein Zugriff auf die Platten erforderlich ist. Diese Optimierungsverfahren wird auch von Datenbankservern angewendet.

Moderne Betriebssysteme verfügen über Threads. Ein Thread ist ein untergeordnetes Objekt, das im Rahmen von Prozessen ausgeführt wird. Mit Hilfe von Threads kann die Programmierung vereinfacht und die Systemnutzung optimiert werden:

> *„In Solaris dauert die Erstellung eines Prozesses etwa 30 Mal so lang wie die Erstellung eines Threads, der Zugriff auf Synchronisationsvariablen dauert ca. 10 Mal so lang und ein Kontextwechsel dauert ungefähr 5 Mal so lang.“*
> Threads Primer, Seite 21

Mit Threads lassen sich die Leistung und Skalierbarkeit großer Anwendungen wie Datenbankserver verbessern. Dies wirkt sich auch auf anderen Gebieten aus, wie wir später sehen werden. Die Architektur von IDS ist auf einem Threadingmodell aufgebaut.

Mit den Jahren hat sich die Ausstattung von Computern von 1 CPU auf 64 oder sogar 128 CPUs gesteigert. Im Sinne der Optimierung der CPU-Nutzung wurde das Konzept der Prozessoraffinität entwickelt. Darunter versteht man die Neuterminierung eines Programms auf derselben CPU, auf der es zuletzt ausgeführt wurde. Dies begann als Funktion, die von Programmen optional aufgerufen wurde, und entwickelte sich schließlich zum Terminierungsalgorithmus des Betriebssystems. Bei dieser Funktion steigt die Wahrscheinlichkeit, dass sich die Anweisungen und Daten eines Programms bei dessen Neuterminierung noch immer im Cache der CPU befinden. Das zeigt, wie wichtig es ist, den Speicherzugriff zu optimieren, um die volle Leistung der CPUs zu nutzen.

Betriebssysteme optimieren also den Plattenzugriff durch die Pufferung der Daten im Hauptspeicher und den Speicherzugriff durch die Pufferung der Daten im CPU-Cache. Darüber hinaus bieten sie weitere Möglichkeiten, mit Hilfe von Threads den System-Overhead zu minimieren. Insgesamt tragen diese Funktionen dazu bei, Leistung und Skalierbarkeit auf hohem Niveau zu gewährleisten.

Datenbankserverarchitektur

Datenbankserver werden in einer speziellen Hardwarearchitektur ausgeführt. Es gibt Architekturen mit einem Einzelprozessor, mit symmetrischen Multiprozessoren (SMP), mit massiv-parallelen Prozessoren (MPP) etc.

IDS wird auf Einzelprozessor- und SMP-Maschinen ausgeführt. Oracle kann mit Hilfe von RAC (Real Application Cluster) auch auf MPP-Maschinen ausgeführt werden. Wir werden weiter unten sehen, warum dies kein Nachteil für IDS ist.

Die allgemeine Architektur aller wichtigen Datenbankserver weist viele Ähnlichkeiten auf. Sie enthält beispielsweise stets folgende Merkmale:

- Gemeinsame Speichernutzung zur Datenpufferung und zum Austausch von Informationen zwischen Prozessen
- Prozesse zur Verarbeitung der physischen und logischen Protokollierung
- Asynchrone E/A und E/A für Vorauslesen

Wir wollen hier nicht die gesamte Architektur im Detail besprechen, sondern uns auf einige zentrale Unterschiede zwischen IDS 10.0 und Oracle 10g konzentrieren. Wir werden diese Architekturen zum besseren Verständnis in einer Implementierung auf Unix-Systemen betrachten.

IDS-Architektur

Die Architektur von IDS 10.0 basiert auf einer Gruppe von Prozessen, die als virtuelle Prozessoren (VPs) bezeichnet werden. Es gibt mehrere Arten von virtuellen Prozessoren zur Ausführung bestimmter Aufgaben. Jeder VP ist multithreadfähig, d. h., er kann mehrere Aufgaben verarbeiten und sich je nach Situation durch Hinzufügen bzw. Entfernen von Threads dynamisch an den Systembedarf anpassen. Daraus ergibt sich eine limitierte Anzahl Unix-Prozesse, die für die Verwaltung des Systems bei jeder beliebigen Arbeitslast erforderlich sind. Dieses Threadingmodell wurde speziell zur Optimierung von Datenbankoperationen konzipiert. Über die Jahre wurde es im Sinn einer Leistungsoptimierung verbessert.

Die Threadingarchitektur von IDS erstreckt sich auch auf die Verarbeitung von Anwenderverbindungen und die parallele Ausführung von SQL-Abfragen bzw. anderen Datenbankoperationen. Jeder Thread bekommt ein Zeitfenster für die Ausführung in einem CPU-VP und wird anschließend neu terminiert. Auf diese Weise wird verhindert, dass ein Thread die Ressourcen für sich allein nutzt. Außerdem wird sichergestellt, dass der Betrieb selbst bei einer großen Anzahl von Anwendern und verschiedenen Abfragen reibungslos abläuft.

Da die Erstellung, Terminierung und Synchronisation von Threadoperationen effizienter als vergleichbare Operationen auf Prozessen ist, bietet IDS die optimale Grundlage für Leistung und Skalierbarkeit und erleichtert darüber hinaus die Verwaltung, wie wir später noch sehen werden.

Oracle-Architektur

Die Architektur von Oracle 10g ist quasi mit den Vorgängerreleases von Oracle identisch. Im Gegensatz zur Architektur von IDS ist Oracle 10g prozessbasiert. Der Anwender kann die Verbindung zu Oracle entweder über einen dedizierten Prozess oder mit Hilfe gemeinsam genutzter Serverprozesse herstellen. Gemeinsame Serverprozesse bieten eine gewisse Optimierungsmöglichkeit:

> *„Im Allgemeinen ist es besser, über einen Dispatcher und einen gemeinsamen Serverprozess verbunden zu sein... Ein gemeinsamer Serverprozess kann effizienter sein, da er die Anzahl der für die ausgeführte Instanz erforderlichen Prozesse gering hält.“*
> Oracle 10g Administrator’s Guide, Seite 4-1.

Die gemeinsam genutzten Serverprozesse dienen bei Oracle dazu, Prozesse auf der Basis bestimmter Konfigurationsparameter bedarfsgerecht hinzuzufügen und zu entfernen. Die zeitliche Einplanung der Abfragen geschieht wie folgt:

1. Ein Dispatcher empfängt eine Anforderung.
2. Er stellt die Anforderung in eine Anforderungswarteschlange.
3. Der nächste verfügbare gemeinsame Prozess nimmt eine Anforderung aus der Warteschlange und führt sie vollständig aus.
4. Der gemeinsame Prozess stellt das Ergebnis in die Antwortwarteschlange.
5. Ein Dispatcher nimmt die Antwort aus der Warteschlange und gibt sie dem Anwender zurück.

Dieses Terminierungsverfahren eignet sich nicht für eine Time-Sharing-Umgebung, sondern eher für die Stapelverarbeitung. Es können Situationen auftreten, in denen mehrere gemeinsame Prozesse durch lange Anforderungen gebunden sind. Daher wird in der Dokumentation zu Oracle 10g Folgendes ausgeführt:

> *„In den folgenden Situationen sollten Anwender und Administratoren die Verbindung zu einer Instanz jedoch explizit über einen dedizierten Serverprozess herstellen: …"*

Bei einem dedizierten Serverprozess wird für jede Verbindung ein eigener Prozess erstellt. Eine gemeinsam genutzte Serverinstanz kann auch Netzservices enthalten, die speziell zur Erstellung dedizierter Serververbindungen genutzt werden.

Kommentare zur Architektur

IDS 10.0 bietet auf Grund seiner Multithread-Architektur, die weniger Overhead generiert als Prozesse, eine hervorragende Grundlage für hohe Leistung. Oracle sieht das genauso:

> *„Das Betriebssystem braucht mitunter sehr viel Zeit für die zeitliche Einplanung und den Wechsel zwischen Prozessen. …*
> *Auf Grund der betriebssystemspezifischen Merkmale braucht Ihr System möglicherweise sehr lange für einen Kontextwechsel. Ein Kontextwechsel kann teuer werden, insbesondere bei einer großen SGA. …"*
> Oracle 10g Performance Tuning Guide, Seite 9-9

Laut dem obigen Zitat aus dem Thread Primer dauert unter Solaris der Kontextwechsel von Prozessen 5 Mal so lang wie von Threads. Bei anderen Betriebssystemen sind ähnliche Unterschiede zu erwarten. Mit zunehmender Systemauslastung wird die Wirkung des Kontextwechsels deutlicher.

Die zeitliche Einplanung von Anwenderabfragen ist ebenfalls von Bedeutung. Bei Oracle wirkt sie sich auf die Anzahl der gemeinsamen Prozesse aus, die für die Unterstützung des Systems erforderlich sind. Oracle konfiguriert die minimale und maximale Anzahl der gemeinsamen Prozesse. Werden mehr Prozesse benötigt, werden sie bis zum festgelegten Höchstwert erstellt, was einen erhöhten System-Overhead zur Folge hat. Ein nicht korrekt eingestellter Höchstwert kann erhebliche Leistungsprobleme verursachen, wenn Anforderungen von Anwendern zum Warten gezwungen werden, selbst wenn sie nur einen geringen Aufwand erfordern. Solchen Anwendern erscheint der Datenbankserver unter Umständen als nicht reaktionsfähig, wenn nicht sogar als funktionsunfähig.

Mit zunehmender Auslastung des Datenbanksystems muss mehr Verarbeitungsleistung zugewiesen werden. Wo IDS mehr Threads erfordert, benötigt Oracle mehr Prozesse.

Die Steigerung des Overheads (Prozess- bzw. Threaderstellung, Synchronisation, Kontextwechsel) geht bei IDS langsamer vonstatten als bei Oracle, ebenfalls auf Grund der Threadingarchitektur.

Selbstverständlich gibt es neben einer besseren Serverarchitektur noch weitere Überlegungen zu Leistung und Skalierbarkeit. Diese werden im nächsten Abschnitt besprochen.

Weitere Überlegungen zu Leistung und Skalierbarkeit

Leistung und Skalierbarkeit beginnen mit einer Architektur, die den System-Overhead minimiert. Danach folgt die Nutzungsoptimierung der übrigen Ressourcen. Sie umfasst die parallele Ausführung, Indexierung und Datenpartitionierung. Daneben könnten weitere Punkte aufgeführt werden, doch es würde hier zu lange dauern, die Unterschiede zu erläutern. Die drei oben genannten Faktoren sind die wichtigsten und lassen sich gut zum Aufzeigen der Unterschiede verwenden.

Parallele Ausführung

Unter paralleler Ausführung versteht man die Fähigkeit, eine Anforderung in mehrere gleichzeitig ausführbare Abschnitte zu unterteilen. Dies ist nützlich, wenn wir das Ergebnis einer komplexen, anspruchsvollen Abfrage möglichst schnell benötigen oder Indizes erstellen müssen. Dank dieser Funktion können wir mehrere Leseoperationen von mehreren Plattenlaufwerken ausführen und mehrere CPUs verwenden.

IDS wurde von vornherein so konzipiert, dass eine parallele Ausführung unterstützt wird. Dabei werden Abfragen in mehrere Ausführungsthreads unterteilt, die durch interne, als „Exchanges“ bezeichnete Mechanismen miteinander verbunden sind. Dadurch wird die Unabhängigkeit zwischen den verschiedenen spezialisierten Threads (Scannen, Sortieren, Gruppieren etc.) gewährleistet. Exchanges teilen die Last auf mehrere Worker-Threads so auf, dass jeder dieselbe Arbeitsmenge hat. Auf diese Weise werden die Systemressourcen optimal genutzt.

IDS unterstützt eine Reihe paralleler Operationen, nämlich das Erstellen von Indizes, Sortieren, Wiederherstellen, Scannen, Verknüpfen, Zusammenfassen, Gruppieren sowie das Ausführen von anwenderspezifischen Routinen (UDR).

Die Abfrageparallelität von IDS wird durch mehrere Faktoren beeinflusst. Dazu zählen die Speichermenge, die parallelen Abfragen zugeordnet wird, die Anzahl der parallelen Abfragen, die gleichzeitig ablaufen dürfen, sowie die Anzahl der Datenbankbereiche, auf die bei einer bestimmten Abfrage zugegriffen wird.

Oracle stellt in seiner Enterprise Edition ebenfalls parallele Abfragen bereit. Beim Start einer Oracle-Instanz wird ein Pool von Prozessen erstellt, die für parallele Operationen nützlich sind. Der Grad der Parallelität wird durch Hinweise sowie durch Sitzungs-, Tabellendefinitions- und Indexdefinitionseinstellungen bestimmt. Mit anderen Worten: Die Parallelität richtet sich bei Oracle nach der Anzahl der parallelen Prozesse, die Sie für eine Operation einrichten.

„…parallele Serverprozesse bleiben im gesamten Verlauf der Ausführung einer Anweisung zugeordnet. Nach Abschluss der Verarbeitung der Anweisung stehen diese Prozesse wieder zur Verarbeitung weiterer Anweisungen zur Verfügung.“
Oracle 10g Administrator’s Guide, Seite 4-15.

Auch hier verwendet Oracle Prozesse anstelle von Threads. Dabei werden mehr Systemressourcen benötigt als bei Threads, wie wir bereits gesehen haben. Die Parallelität wird bei Oracle durch den Datenbankadministrator oder den Anwender bestimmt, der die Anzahl der Prozesse festlegt, die zur Parallelverarbeitung der Abfrage zu verwenden sind, nicht durch den Datenbankserver, der die Anforderungen auf Basis der für parallele Operationen verfügbaren Ressourcen (Hauptspeicher, CPUs, Plattenanzahl) festlegt. Dies erschwert die Nutzung der Parallelitätsfunktionen von Oracle und erhöht ihre Fehleranfälligkeit bei der Optimierung.

Datenpartitionierung

Die Datenpartitionierung bietet die Möglichkeit zur Verteilung der Daten auf mehrere Plattenlaufwerke. Da die einzelnen Platten parallel durchsucht werden können, erhöht sich der Durchsatz.

IDS unterstützt Partitionierung seit mehreren Jahren. Die Datenverteilung erfolgt entweder im Umlaufverfahren oder nach Bereich (Ausdruck).

Beim Umlaufverfahren werden die Daten gleichmäßig auf eine Gruppe von Platten verteilt. Sie können dabei alle Plattenressourcen nutzen, ohne die Verteilung der Daten kennen zu müssen. Dieses Verfahren ist auch in Umgebungen nützlich, in denen Sie alle bzw. einen Großteil der Daten verarbeiten müssen, z. B. in Data Marts/Data Warehouses oder bei der Berichterstellung. Bei der bereichsorientierten Partitionierung werden die Daten mit Hilfe von Ausdrücken auf logische Speicherbereiche verteilt, die als Datenbankbereiche (dbspaces) bezeichnet werden. Da der Server den zur Verteilung der Daten verwendeten Ausdruck kennt, kann er dank dieser Information bei einer Abfrage Speicherfragmente ausschließen und sich beim Zugriff auf diejenigen Platten beschränken, auf denen sich die angeforderten Zeilen befinden können. So wird die Anzahl der auszuwertenden Zeilen erheblich reduziert, die Menge der benötigten Ressourcen verringert und die Leistung gesteigert.

IDS 10.0 bietet zusätzlich die Möglichkeit, mehrere Partitionen in einem Datenbankbereich zu speichern. Dies ist insbesondere bei einer differenzierteren Partitionierung von Vorteil, bei der die Anzahl der Partitionen die der physischen Plattenlaufwerke bei weitem übersteigt. Beispielsweise könnten Sie vor der Situation stehen, Daten in einem gleitenden Zeitfenster von 15 Monaten tageweise zu partitionieren. Möglicherweise wollen Sie die 15 verschiedenen Platten beibehalten und die Daten auf jeder Platte weiter nach Tagen aufteilen. Jede Abfrage, die einen Tag oder einen Bereich von Tagen (z. B. eine Woche) angibt, würde dabei eine erhebliche Verringerung der Datenmenge bewirken, die von der Platte gelesen werden müsste.

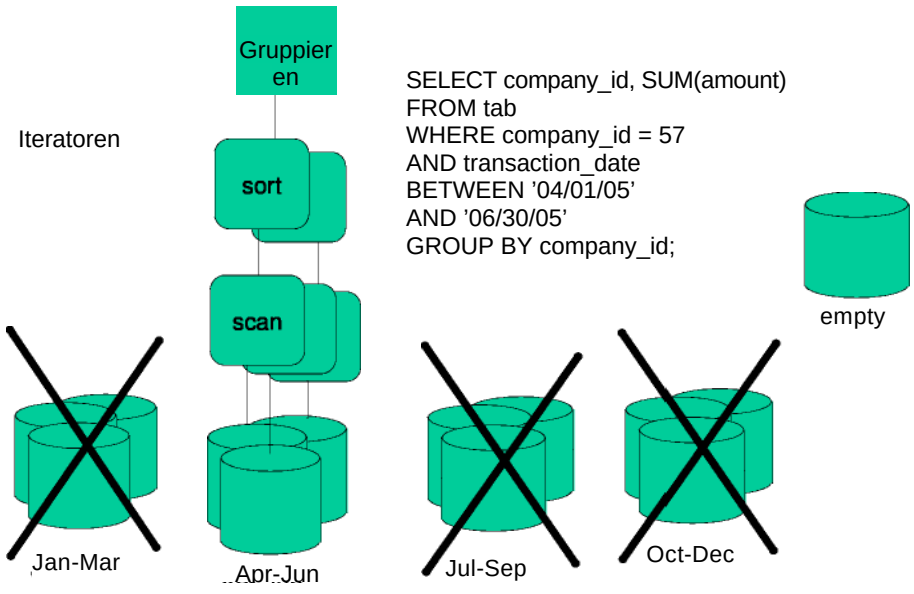
Diese Funktion ist wichtig, weil eine Verringerung der von der Platte zu lesenden Datenmenge potenzielle Wartezeiten der Platte senkt. Außerdem bewirkt sie eine bessere Nutzung des Puffercaches, weil nur die relevanten Informationen in diesen Cache geladen werden und somit kein Speicherplatz belegt wird, der anderweitig besser verwendet

werden kann. Auch Indizes können der Bereichspartitionierung folgen, d. h., jede Partition enthält einen kleinen Index, der sich durchsuchen lässt. Dies ist ein weiterer potenzieller Leistungsvorteil.

Ab IDS 9.21 (2001) wurde diese Funktion weiter verbessert. Sie können nun Tabellen ändern oder Tabellenfragmente an- bzw. abhängen. Da die Fragmente dem Partitionierungsausdruck folgen, können alte Fragmente entfernt und neue Fragmente hinzugefügt werden, ohne die Indizes neu erstellen zu müssen.

Bei der Verwendung eines gleitenden Datenfensters wie oben haben wir mehrere Vorteile. Wir könnten einen neuen Monat mit Daten in einer neuen Tabelle laden, den geeigneten Index dafür erstellen und sie schnell an die Haupttabelle anhängen. Ebenso schnell könnten wir einen Datenbereich aus der Haupttabelle entfernen, die Daten sichern und den Speicherplatz für die weitere Nutzung freigeben. Das Ergebnis sind eine höhere Verfügbarkeit und effizientere Operationen beim Entfernen eines Wertebereichs.

Die folgende Abbildung veranschaulicht die Verwendung der Bereichspartitionierung bei einer SQL-Abfrage.



Oracle hat sich dazu entschlossen, den Ansatz von IDS zu übernehmen und Unterstützung für die Partitionierung bereitzustellen. Das Produkt unterstützt drei Partitionierungsverfahren: Bereichs-, Hash- und kombinierte Partitionierung. Bei der Hashpartitionierung wird mit einem Hash-Schlüssel bestimmt, in welcher Partition die Zeile gespeichert wird. Dabei ergeben sich nur im Falle von Gleichheitsbedingungen Vorteile auf Grund der Vermeidung von Fragmenten. Bei der kombinierten Partitionierung werden die Daten nach dem Bereichsverfahren partitioniert und innerhalb der einzelnen Partitionen nach dem Hashverfahren weiter unterteilt.

Die Partitionierungsstrukturen von Oracle unterstützen nur Verfahren, bei denen Sie Ihre Daten analysieren müssen, um sicherzustellen, dass sie gleichmäßig auf den Partitionen verteilt werden können. Bei der kombinierten Partitionierung können Partitionen innerhalb einer Partition erstellt werden. Dies führt in aller Regel zu einer vermehrten Bewegung des Schreib-/Lesekopfs und damit zur Senkung des Durchsatzes der Platteneinheit. Somit hat diese Option nur einen beschränkten Nutzen.

Oracle bietet eine Möglichkeit zum An- und Abhängen neuer Partitionen. Mit der Anweisung „ALTER TABLE . . . DROP PARTITION“ werden alle Partitionen des globalen Indexes als nicht verwendbar gekennzeichnet. Somit bringt jede Neuorganisation der Partitionierung einer Tabelle erhebliche Ausfallzeit mit sich, da die Indizes neu erstellt werden müssen.

Alles in allem bietet Oracle weniger nützliche Partitionierungsoptionen und keine Funktion zum An- bzw. Abhängen von Partitionen, die in einem Geschäftsumfeld oft von zentraler Bedeutung ist.

Indexierung

Alle Datenbankprodukte bieten Indexierungsfunktionen. IDS unterstützt B-Tree-Indizes, Clusterindizes und R-Tree-Indizes. Außerdem unterstützt IDS eine API, die die Implementierung neuer Indexierungsverfahren wie Textsuchen (beispielsweise im Excalibur Text-DataBlade) unterstützt. Der B-Tree-Index kann auch zur Erstellung eines Indexes für das Ergebnis einer anwenderspezifischen Routine (UDR) verwendet werden. Dies wird als funktionaler Index bezeichnet.

Oracle unterstützt folgende Indextypen: B-Tree, Cluster, Reverse Key, Hash, Bitmap und Bitmap Join, Functional, R-Tree sowie spezialisierte Indizes. Die beiden ersten Typen gelten als Arbeitstiere der Datenbanksysteme und existieren schon sehr lange.

Der Reverse Key-Index unterscheidet sich vom standardmäßigen B-Tree-Index dadurch, dass er das Byte jeder indexierten Spalte umkehrt. Oracle gibt an, dass damit Leistungseinbußen in Oracle RAC vermieden werden können, allerdings auf Kosten einer geringeren Funktionalität.

> *„…kann dazu beitragen, Leistungseinbußen bei Real Application Clusters zu vermeiden… Bei Verwendung von Reverse Key-Indizes geht die Möglichkeit verloren, Abfragen mit Bereichseinschränkungen mit Hilfe des Indexes zu beantworten.“*
> Oracle 10g Database concepts, Seite 5-29, 5-30

Ein Hashindex ist ein anderer Indextyp mit eingeschränkter Verwendbarkeit. Er bietet bei Gleichheitsoperationen eine gute Leistung, eignet sich jedoch nicht für Bereichsoperationen.

Ein Bitmapindex wird in einer Data Warehouse-Umgebung eingesetzt und ist nützlich, wenn in den indexierten Spalten sehr wenige verschiedene Werte verwendet werden.

> *„Bitmapindizes eignen sich nicht für OLTP-Anwendungen, bei denen die Daten mit einer großen Anzahl gleichzeitig ablaufender Transaktionen geändert werden“*
> Oracle 10g Database concepts, Seite 5-31

Oracle musste den Ansatz von Informix übernehmen und den R-Tree-Index implementieren. Dieser Indextyp bietet effiziente Suchläufe für mehrdimensionale Probleme. Er wird vornehmlich bei räumlichen und geodätischen Anwendungen eingesetzt. Oracle hat die Unterstützung seines Quadtree-Indexes für räumliche Anwendungen eingestellt.

> *„… von der Verwendung des Quadtree-Indexes wird abgeraten, und es wird sehr empfohlen, den R-Tree-Index zu verwenden.“*
> Oracle 10g, Spatial user’s guide and reference, Seite 1-9

Interessanterweise scheint das Oracle R-Tree-Indexierungsverfahren nicht als Index in Oracle integriert zu sein:

> *„Ein R-Tree-Index wird in der räumlichen Indextabelle gespeichert… Der R-Tree-Index unterhält auch ein Sequenzobjekt, um zu gewährleisten, dass Anwender den Index gleichzeitig aktualisieren können.“*
> Oracle 10g, Spatial user’s guide and reference, Seite 1-10

Räumliche Daten werden für viele Unternehmen immer wichtiger. Sie können als grundlegendes Indexierungsverfahren für zahlreiche OLTP-Anwendungen betrachtet werden. Das obige Zitat von Oracle zeigt, dass der Hersteller beschlossen hat, Kompromisse einzugehen, um zu Informix aufzuschließen, anstatt eine optimierte Lösung anzubieten.

Der Abschnitt zur Indexierung macht deutlich, dass Oracle unter anderem deshalb mehr Indexierungsoptionen als IDS bereitstellt, weil damit einige Unzulänglichkeiten von Oracle behoben werden sollen. Außerdem zeigt er, dass zumindest der R-Tree-Index nicht ordnungsgemäß in Oracle integriert wurde.

IDS bietet klare Optionen für die Indexierung. Es gibt keinerlei Unklarheiten über die jeweilige Verwendung der einzelnen Verfahren. Alle Indexierungsverfahren sind in die Steuerkomponente integriert und bieten die bestmögliche Leistung.

Skalierbarkeit der Hardware

Manche Benutzer sind hinsichtlich der Skalierbarkeit eines Einzelsystems skeptisch. Sie fragen sich, ob nicht ein Produkt sinnvoller wäre, bei dem sich der Datenbankserver über mehrere physische Maschinen erstrecken kann. Dies gehört zur Verkaufspolitik von Oracle

RAC. Da Oracle RAC ein integraler Bestandteil einer Oracle-Lösung ist, werden wir dieses Thema später behandeln.

TPC-C ist ein Standardbenchmark, der Informationen über die Leistung von Systemen bereitstellt, die in einer OLTP-Umgebung verwendet werden. Interessanterweise wurden die beiden besten Ergebnisse dieses TPC-C-Benchmarks (bis zur Verfassung dieses Artikels) auf Einzelsystemen erzielt. Das höchste TPC-C-Ergebnis mit einem Wert von 3,2 Millionen tpmC (Transaktionen pro Minute) erreichte ein IBM System p5 595 mit 64 Prozessoren. Dieses System hatte 2 TB Hauptspeicher und 243 TB Plattenspeicher.

Eine derartige Leistung wird weltweit nur von einer Handvoll sehr großer Unternehmen benötigt. Angesichts des Mooreschen Gesetzes (die Prozessorgeschwindigkeit verdoppelt sich alle 18-24 Monate) ist zu erwarten, dass die Leistung von Einzelsystemen auch weiterhin in der gleichen Größenordnung wächst. Wenn man also davon ausgeht, dass die aktuelle Hardwareleistung für die derzeitige Umgebung eines Unternehmens mehr als ausreichend ist, dürfte dies unter Berücksichtigung des Mooreschen Gesetzes auch in Zukunft so bleiben. Darüber hinaus lässt sich eine Umgebung mit einer einzelnen Maschine viel leichter verwalten als eine Datenbank, die mehrere physische Maschinen umfasst.

Zuverlässigkeit

Es ist äußerst schwer, die Zuverlässigkeit von Produkten zu vergleichen, da die meisten bereitgestellten Informationen als subjektiv betrachtet werden könnten. Wir könnten auf die „unbreakable“ Kampagne von Oracle hinweisen, die viele Oracle-Sites zum Ausfall brachte, um zu belegen, dass das Produkt nicht „unbreakable“ (unverwundbar/einbruchssicher) war.

IDS 10.0 setzt durch eine weitere Intensivierung der Labortests auf die kontinuierliche Verbesserung der Zuverlässigkeit des Produkts. Außerdem sind viele Kunden mit der Zuverlässigkeit von IDS sehr zufrieden. Die Steuerbehörde von El Salvador, einer der IDS-Kunden, berichtet beispielsweise:

> *„Die Möglichkeit, Steuerdokumente 10 Jahre lang auf einer sicheren und zuverlässigen Plattform aufzubewahren, ist für die Handlungsfähigkeit von El Salvador immens wichtig.“*

Es gibt zahlreiche anekdotenhafte Berichte über Systeme, die beinahe in Vergessenheit geraten wären, weil sie so reibungslos laufen. Andere Systeme werden nur heruntergefahren, weil ihre Betreiber beschlossen haben, ihre Computersysteme einmal pro Quartal zu warten. Ein Großkunde schließlich berichtete, dass eine große Anzahl der Instanzen 99,998 % der Zeit betriebsbereit sei, einschließlich der Wartungszeit. Dies entspricht einer Ausfallzeit von etwas mehr als 5 Minuten pro Jahr Verfügbarkeit – ein eindrucksvoller Beleg für Zuverlässigkeit.

IDS ist ein führender Hersteller in den Branchen Handel, Telekommunikation, Elektronik sowie Bank- und Gesundheitswesen. Für all diese Kunden ist Zuverlässigkeit bei der Ausführung ihrer geschäftskritischen Anwendungen ein unerlässlicher Faktor, und IDS erfüllt diese Anforderung.

Verfügbarkeit

Unter Verfügbarkeit versteht man die Möglichkeit, ein System online zu verwalten (geplante Ausfallzeiten zu minimieren), jede Art von Nichtverfügbarkeit zu verringern und bei Störungen eine schnelle Wiederherstellung zu erreichen.

IDS 10.0 bietet viele Onlineoperationen, darunter die bereits erwähnte konkurrenzlose An- und Abhängefunktion von IDS sowie die Erstellung und Entfernung von Indizes. Daneben stellt IDS viele Systemmanagement- und Optimierungsoperationen bereit, die auch im laufenden Betrieb des Datenbanksystems durchgeführt werden können. Kunden nennen hier Funktionen wie die Onlinesicherung, die schnelle Wiederherstellung nach Stromausfällen, die zeitnahe Beseitigung von Problemen mit der Steuerkomponente oder Anwendungen sowie das überaus bequeme Abrufen von Daten, die zur Früherkennung potenzieller Leistungsprobleme oder schwerwiegender Störungen noch vor deren Auftreten erforderlich sind. Diese Funktionen tragen dazu bei, dass die Leistung von IDS konstant hoch bleibt und die Datenbank noch verfügbarer wird.

Außerdem unterstützt IDS eine leistungsfähige Peer-to-Peer-Replikationsfunktion, die die gemeinsame Nutzung einer beliebigen Teilmenge von Daten durch Hunderte von Maschinen möglich macht, die über die ganze Welt verstreut sein können. Es gibt tatsächlich einige Installationen, die in solchen Umgebungen laufen. IDS ist das einzige Datenbanksystem, das eine Replikation in dieser Größenordnung bietet.

Für die Wiederherstellung nach einem Katastrophenfall mit hoher Verfügbarkeit bietet IDS die Funktion HDR (High-availability Disaster Recovery), die sich durch eine einfache Einrichtung und Verwaltung auszeichnet. Sie ermöglicht die Replikation einer Datenbankinstanz an einem anderen Standort. Die zweite Maschine kann als schreibgeschützter Server für Abfragen und Berichte genutzt werden. Da die zweite Maschine eine geringere Auslastung als die Primärmaschine aufweist, können die verfügbaren CPU-Zyklen für nicht geschäftskritische Operationen genutzt werden, die bei einer Katastrophe auf der Primärmaschine unterbrochen werden können. Dadurch bleiben selbst im Katastrophenfall 100 % der Ressourcen des Unternehmens den Produktionsanwendungen zugeordnet. Die Funktion HDR ist im Lieferumfang von IDS Enterprise Edition enthalten und für IDS Workgroup Edition als Option verfügbar.

Oracle stellt einige Onlineoperationen bereit. Gehen wir davon aus, dass sie mit IDS vergleichbar sind (in jedem Fall mit Ausnahme der oben erwähnten Partitionierungsfunktion), und konzentrieren wir uns auf die Oracle-Lösung zur Wiederherstellung nach einem Katastrophenfall. Oracle führt eine Reihe von Lösungen an. Was Hochverfügbarkeit angeht, beschränkt sich diese auf Oracle RAC und Data Guard.

Das Tool zur Hochverfügbarkeit, das hauptsächlich von Oracle beworben wird, ist RAC. Im nächsten Abschnitt wird Oracle RAC detailliert unter den Gesichtspunkten Leistung/Skalierbarkeit und Hochverfügbarkeit besprochen. Dabei werden die Funktionsweise und Einschränkungen dieses Produkts beschrieben. Außerdem wird erläutert, warum sich Data Guard besser zur Wiederherstellung nach einem Katastrophenfall eignet.

Data Guard, die andere von Oracle bereitgestellte Lösung, bietet eine Standby-Einrichtung für die Primärmaschine. Die Standby-Maschine kann die Replikation während Abfragen unterbrechen und somit als schreibgeschützte Maschine verwendet werden:

> *„Data Guard unterhält eine physische Standby-Datenbank durch die Ausführung einer Redo Apply-Funktion. Wenn eine physische Standby-Datenbank nicht gerade durch eine Wiederherstellung belegt ist, kann sie im schreibgeschützten Modus geöffnet werden, …“*
> Oracle 10g, Data Guard Concepts and Administration, Seite 2-1

Dies erschwert die Nutzung der Standby-Maschine. Data Guard kann auch in einem logischen Bereitschaftsmodus zur Replikation eines Teils der Primärdatenbank genutzt werden. Dabei lassen sich bis zu neun Standby-Maschinen replizieren, ein Bruchteil der Möglichkeiten, die IDS Enterprise Replication (Anmerkung: IDS Enterprise Replication ist hier korrekt.) mit der Peer-to-Peer-Replikation in Topologien bietet, die Hunderte von Systemen unterstützen.

Insgesamt scheint IDS in puncto Verfügbarkeit zumindest einen kleinen Vorsprung vor Oracle zu haben. Nachdem wir damit die Bereiche Leistung/Skalierbarkeit und Verfügbarkeit besprochen haben, können wir uns jetzt Oracle RAC zuwenden, einem zentralen Bestandteil der Datenbanklösung von Oracle.

Oracle Real Application Cluster (RAC)

RAC ist in Bezug auf Skalierbarkeit und Hochverfügbarkeit die Nummer 1 unter den Oracle-Lösungen. Oracle setzt auf das Konzept, mit Hilfe mehrerer einfacher Computer ein System auf Unternehmensniveau zu erstellen. Schauen wir uns die Architektur dieser Lösung an, um herauszufinden, wie weit dieses Konzept aufgeht.

Architektur von RAC

Oracle RAC unterstützt eine einheitliche Datenbanksicht über mehrere Computersysteme (theoretisch bis zu 100). Dies ist eine „shared Disk“-Umgebung. Somit kann jeder Server im Cluster auf sämtliche Daten auf allen Platten zugreifen. Zur Unterstützung dieser Umgebung muss Oracle die Datenbankoperationen zwischen den Servern koordinieren. Dies wird durch die Integration folgender Zusatzfunktionen erreicht:

- Globaler Cache-Service-Prozess
- Globaler Enqueue-Service-Prozess
- Globales Ressourcen-Directory, das auf alle aktiven Instanzen verteilt ist
- Cache-Fusion zur Übertragung von Datenblöcken zwischen Instanzen

Diese Komponenten werden zusätzlich zu den Komponenten benötigt, die für eine Instanz erforderlich sind, die nicht in einer RAC-Umgebung betrieben wird. Zudem müssen Sie vor der Installation von RAC noch die Oracle Clusterware installieren.

Skalierbarkeit von RAC

Oracle empfiehlt aus Gründen der Skalierbarkeit die Implementierung von RAC auf mehreren, mit einfachen Prozessoren ausgestatteten Maschinen. In diesem Abschnitt wird die Skalierbarkeit bei diesem Ansatz im Vergleich mit der Skalierbarkeit bei einer Einzelmaschine untersucht. Nehmen wir ein Cluster aus vier Maschinen mit einer einzigen CPU, je 1 GB Hauptspeicher und einigen Plattenlaufwerken und vergleichen es mit einer 4-CPU-Maschine, die 4 GB Hauptspeicher und die entsprechende Anzahl Plattencontroller und Plattenlaufwerke besitzt. Wie schneiden diese beiden Ansätze beim Vergleich der Skalierbarkeit ab? Ich bin der Ansicht, dass RAC der falsche Ansatz für Skalierbarkeit ist, und werde das nachstehend darlegen.

System-Overhead

Im oben genannten Konfigurationsbeispiel sehen wir, dass bei RAC vier Instanzen des Betriebssystems und der zugehörigen Prozesse statt einer einzigen verwendet werden. Außerdem werden vier Datenbankinstanzen mit all ihren Prozessen und Datenstrukturen statt einer einzigen verwendet.

Für die Koordination zwischen den Instanzen werden zudem die oben erwähnten Zusatzprozesse und -strukturen benötigt. Oracle weist darauf mit Warnungen wie der folgenden Aussage hin:

> *„Auf Grund der Cache-Fusion gelten bei RAC höhere Anforderungen an die SGA-Größe als bei Oracle-Datenbanken, die nur aus einer einzigen Instanz bestehen.“*
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, Seite 1-6

SGA steht für System Global Area. Dies ist eine Struktur, die im gemeinsam genutzten Speicher verwaltet wird.

Für die Ausführung eines Betriebssystems ist Speicher erforderlich. Die Mindestkonfiguration für Linux beispielsweise liegt bei ca. 32 MB; 64 MB wird empfohlen. Dazu kommen noch die Voraussetzungen für eine Datenbankserverinstanz. Im Oracle-Handbuch zur Schnellinstallation unter Linux werden als Hardwarevoraussetzungen 1.024 MB Hauptspeicher bei 400 MB Plattenspeicherplatz im Verzeichnis /tmp und 1,5-3,5 GB Plattenspeicherplatz für die Softwareinstallation angegeben.

Für die Implementierung von RAC sind mehrere Instanzen des Betriebssystems, mehrere Instanzen der Oracle-Datenbanksoftware (und Oracle Clusterware) sowie zusätzlicher Speicher auf den einzelnen Maschinen für Cache-Fusion erforderlich. Es ist offensichtlich, dass die bereitgestellten 4 GB Hauptspeicher im Cluster nicht den 4 GB auf einer Einzelmaschine entsprechen. Der Unterschied der tatsächlich verfügbaren Speichermenge beträgt Dutzende von Megabyte.

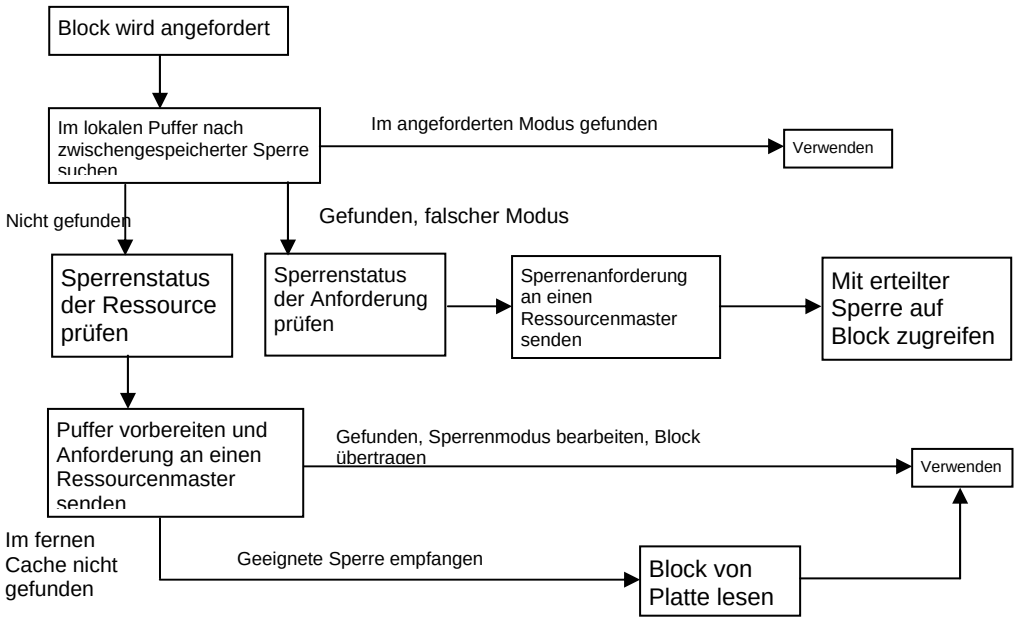
Weiter oben im Abschnitt zur Leistung haben wir gesehen, dass Computerkomponenten verschiedene Leistungsmerkmale aufweisen. Sie können die Systemleistung verbessern, indem Sie mehr Speicher verwenden und den Plattenzugriff (E-/A-Vorgänge) reduzieren. Oracle RAC nimmt Ihnen im Endeffekt einen Teil der Möglichkeiten zur Leistungssteigerung.

Cache-Fusion-Overhead

Alle Datenbanksysteme nutzen den Hauptspeicher, um Daten nahe beim Prozessor zu halten, damit sie nicht ständig auf die Platte zugreifen müssen. Da der Hauptspeicher weitaus leistungsfähiger als die Platte ist, trägt diese Funktion wesentlich zur Gesamtleistung und Skalierbarkeit von Datenbanksystemen bei.

Jede Oracle-Instanz in einer RAC-Umgebung wird auf einer eigenen Maschine ausgeführt und besitzt einen eigenen Cache. Da alle Instanzen zu der einheitlichen logischen Sicht der Datenbank gehören, benötigt Oracle einen Mechanismus zur Verfolgung der Daten im jeweiligen Cache und zur gemeinsamen Nutzung der Daten zwischen Maschinen. In früheren Oracle-Releases war der Distributed Lock Manager (DLM) die zentrale Komponente dieser Funktion. Oracle hat einige Verbesserungen vorgenommen und verwendet jetzt einen Mechanismus, der als Cache-Fusion bezeichnet wird.

Bei Cache-Fusion kann die Datenbankinstanz über eine Netzverbindung einen Datenblock aus dem Puffercache einer anderen Instanz des Clusters abrufen. Diese Operation erübrigt sich auf einem System mit einer einzigen Instanz, wo sich der Datenblock entweder im Hauptspeicher oder auf der Platte befindet. Bei RAC ist dieser Prozess weitaus komplexer. Er wird im nachstehenden Diagramm verdeutlicht:



Diesem Diagramm können wir Folgendes entnehmen: Wenn sich der Block im lokalen Cache im richtigen Modus befindet, kann er sofort verwendet werden. Dieser Fall ähnelt dem Prozess bei einer einzigen Serverinstanz. In allen anderen Fällen müssen wir mit einem Ressourcenmanager über die Sperrenbearbeitung sprechen. Sobald die Sperre bearbeitet wurde, können wir den Block so verwenden, als befände er sich im lokalen Cache. Andernfalls müssen wir herausfinden, ob er sich in einem fernen Cache befindet, den Sperrenmodus prüfen und den Block auf die lokale Maschine übertragen. Befindet er sich nicht in einem fernen Cache, müssen wir uns eine geeignete Sperre für ihn verschaffen und ihn von der Platte abrufen.

Die Übertragung von Cache zu Cache ist besser als der Weg über die Platte (weil dadurch Rotationslatenz und Suchzeit vermieden werden), bedeutet jedoch trotzdem einen zusätzlichen Verarbeitungsaufwand, der auf einem Einzelsystem nicht anfällt. Diese Übertragung erfolgt über eine Netzverbindung. Denken Sie daran, dass die Netzgeschwindigkeit mindestens um eine Größenordnung niedriger ist als die des Hauptspeichers und dass die CPU noch schneller arbeitet. Oracle erteilt eine Warnung zu dieser Übertragung:

> *„Das Interconnect und das Protokoll für die Kommunikation zwischen den Clusterknoten können sich auf die Leistung von Cache-Fusion auswirken. Außerdem bestimmen die Interconnect-Bandbreite, ihre Latenz und die Effizienz des Kommunikationsprotokolls die Geschwindigkeit, mit der Cache-Fusion Blockübertragungen verarbeitet.“*
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, Seite 12-1

Die Skalierbarkeit von Oracle RAC hängt direkt davon ab, wie oft Sie den Zugriff auf einen fernen Cache vermeiden können. Im besten Fall wird immer nur der lokale Cache genutzt. In diesem Fall wird nicht Oracle RAC verwendet, sondern quasi eine Einzelmaschine, wobei trotzdem der oben beschriebene System-Overhead anfällt.

Im ungünstigsten Fall müssen Sie jedes Mal einen Block aus einem fernen Cache abrufen. Dabei müssen Sie die globale Sperre empfangen und den Block übertragen. Wir haben oben gesehen, dass die Netzgeschwindigkeit viel niedriger ist als die des

Hauptspeichers und des Prozessors. Die Blockübertragungsdauer ist für die Computergeschwindigkeit ein sehr wichtiger Faktor. Das Endergebnis wäre die Verlangsamung Ihres Gesamtsystems von der CPU- auf die Netzgeschwindigkeit.

Bei einer ausgewogeneren Berechnung ergibt sich die Wahrscheinlichkeit, einen Block im lokalen Cache zu finden, aus der Division von 100 durch die Knotenanzahl. In einer RAC-Umgebung mit 4 Knoten würden 25% der Blöcke lokal gefunden. Somit wäre in durchschnittlich 75% der Fälle eine Blockübertragung erforderlich. Dadurch verringert sich letztlich die Systemleistung von der CPU-/Hauptspeichergeschwindigkeit auf die Netzgeschwindigkeit.

Dieses Verhältnis wird von vielen Faktoren beeinflusst. Wir haben oben gesehen, dass Oracle zum Schutz vor Leistungseinbußen in RAC einen Reverse Key-Index bereitstellt. Wenn mehrere Maschinen dieselben Index- oder Datenblöcke benötigen, müssen die RAC-Knoten unter Umständen lange warten, bis die Blöcke verfügbar sind und auf ihren Knoten übertragen werden.

Mit einer aufwändigen Überwachung und Optimierung können Sie möglicherweise herausfinden, wie Sie die Daten aufteilen müssen, um zu erreichen, dass die einzelnen Bereiche weitgehend unabhängig voneinander sind und dass nur ein einziger Knoten des Clusters auf einen bestimmten Bereich zugreift. Damit werden die Daten de facto so partitioniert, dass die einzelnen Knoten unabhängig voneinander sind. Dies macht jedoch den Sinn der Skalierbarkeit mit RAC zunichte, da nun jede Maschine praktisch eine unabhängige Datenbankinstanz darstellt. Oracle deutet dies an:

> *„Wenn Sie partitionierte Tabellen und Indizes für OLTP-Umgebungen in Hashwerte umwandeln, können Sie die Leistung Ihrer RAC-Datenbank erheblich steigern. Beachten Sie, dass ein Index mit Hash-Partitionierung nicht für Abfragen mit Bereichseinschränkungen verwendet werden kann.“*
> Oracle 10g Clusterware and RAC Administration and Deployment Guide, Seite 1-12

Wenn Sie eine Tabelle mit der Kostenstelle als Hashkey partitionieren, dann könnten Sie die Kostenstellen den Knoten fest zuordnen und so sicherstellen, dass kein anderer Knoten auf die entsprechenden Daten zugreift, solange keine Operation durchgeführt wird, die eine Bereichsabfrage über den Index erfordert. Das bedeutet aber auch, dass für jede Kostenstelle nur die Verarbeitungsleistung und der Speicher eines bestimmten Knotens verfügbar ist. Sobald Sie einer Kostenstelle mehrere Knoten zuordnen müssen, stehen Sie wieder vor dem oben genannten Problem.

Einschränkungen bei der Abfragenverarbeitung

Datenbanksysteme sind in der Lage, Abfragen in mehrere Teile zu zerlegen und diese parallel zu verarbeiten. Dadurch wird bei den entsprechenden Abfragen ein höherer Durchsatz erreicht. In einem OLTP-System sind unter Umständen bestimmte Berichte zu erstellen, für die große Datenmengen verarbeitet werden müssen. Dies kann das Sortieren der Daten, das Zusammenfassen von Ergebnissen und andere Verarbeitungsschritte umfassen.

Mit dem, was wir inzwischen über Cache-Fusion wissen, lässt sich unschwer erkennen, dass die Ausführung einer Abfrage zur Berichterstellung viele Kopieroperationen von

Blöcken zwischen den einzelnen Maschinen auslösen und damit die Leistung der anderen Knoten beeinträchtigen könnte. Es wäre dann besser, die Berichte außerhalb der Arbeitszeiten zu erstellen, wenn die OLTP-Anwender offline sind. Andererseits dürfte es nicht leicht sein, bei einem Betrieb rund um die Uhr, der sich über mehrere Zeitzonen erstreckt, das geeignete Zeitfenster für die Berichterstellung zu finden.

Anspruchsvolle Abfragen laufen schneller, wenn sie viele Ressourcen nutzen können. Bei Oracle RAC ist die Verarbeitungsleistung auf einen einzigen Knoten beschränkt. Wenn wir auf das Beispiel der einfachen Prozessoren in einer RAC-Umgebung mit vier Knoten zurückkommen, so würde sich die Berichtabfrage dort auf einen einzigen Prozessor und 1 GB Hauptspeicher beschränken, anstatt vier Prozessoren und 4 GB Speicher nutzen zu können. Diese Beschränkung kann noch deutlicher hervortreten, wenn große Datenmengen sortiert werden müssen. Sobald im Hauptspeicher kein Platz mehr verfügbar ist, müssen die Platten zur Speicherung von Teilergebnissen verwendet werden. Dies verlangsamt die Verarbeitung erheblich.

Möglicherweise müssen Sie mehr als einen Bericht erstellen lassen. In diesem Fall könnten Sie jede Berichtabfrage auf einem anderen Knoten ausführen, um mehr Prozessoren und Hauptspeicher nutzen zu können. Wenn Sie das tun, stehen Sie wieder vor dem Problem, dass Blöcke zwischen den Maschinen kopiert werden müssen, wodurch erheblich mehr Zeit für die Ausführung dieser Berichte erforderlich wird.

Skalierbarkeit von RAC – Zusammenfassung

Wir haben oben gesehen, dass Oracle RAC auf Grund des erheblichen Overheads beim System- und Datenzugriff große Schwachstellen bei der Skalierbarkeit aufweist. Durch das Hinzufügen von Knoten werden diese Probleme noch verschärft. Alle Bemühungen zur Verringerung dieses Overheads würden eine aufwändige Überwachung und Optimierung erfordern. Alle Änderungen der Verarbeitungsumgebung würden eine zusätzliche Überwachung und Optimierung erfordern. Im Endeffekt würde die Skalierbarkeit noch immer hinter der eines Einzelsystems zurückstehen.

Im Sinne einer optimalen Leistung und Skalierbarkeit sowie einer besseren Ressourcennutzung empfiehlt es sich daher, Oracle RAC nicht zu verwenden.

Hochverfügbarkeit von RAC

Wir haben gerade gesehen, dass RAC die falsche Lösung für Skalierbarkeit ist. Damit bleibt noch das Versprechen der Hochverfügbarkeit von RAC übrig. Der beste Ansatz für unsere Besprechung ist wahrscheinlich, wenn wir uns auf zwei Knoten beschränken. Da jedoch bei einer solchen Konfiguration ein Ausfall noch gravierendere Folgen hätte, wollen wir uns an das vorige Beispiel mit vier einfachen Maschinen halten.

Zunächst ist anzumerken, dass RAC zwar bei Knotenausfällen eingreift, bei Plattenausfällen dagegen schweigt. Wenn man eine Umgebung mit hoher Verfügbarkeit gewährleisten will, muss man Vorsorge für Plattenausfälle treffen.

Ein weiteres Problem ist die Gefahr von Standortkatastrophen. Fällt ein Standort durch Brand, Erdbeben oder aus einem anderen Grund komplett aus, verlieren Sie die gesamte RAC-Umgebung. Für manche Unternehmen ist der mögliche Verlust eines Rechen-

zentrums eine Überlegung, die in Betracht gezogen werden muss. In einem solchen Fall ist RAC nicht ausreichend. Sie müssten Oracle Data Guard zu der Umgebung hinzufügen. Wie wir bereits gesehen haben, ist Data Guard der mit IDS HDR angebotenen Lösung unterlegen.

Im Rest dieses Abschnitts beschränken wir uns auf die Beschreibung der Funktionen von RAC, ohne auf die in den vorangegangenen Absätzen erwähnten Einschränkungen hinsichtlich der Hochverfügbarkeit einzugehen.

Oracle behauptet, dass der Datenbankserver beim Ausfall eines Knotens weiterhin aktiv ist und Sie Ihre Anwendungen weiter ausführen können. Ich möchte zwei Punkte ansprechen, die nach einem Ausfall wichtig sind: die Wiederherstellung und die Leistung nach der Wiederherstellung.

Wiederherstellung

Oracle RAC muss beim Ausfall eines Knoten mehrere Aktionen ausführen:

- Knotenausfall erkennen
- Remastering der von dem ausgefallenen Knoten gesteuerten Datenblöcke durchführen
- Protokollübernahme und Seitensperrung
- Die Redo- und Undo-Wiederherstellung durchführen

Die Wiederherstellung des ausgefallenen Knotens erfolgt durch einen der noch funktionsfähigen Knoten. Zumindest während der Protokollübernahme und Seitensperrung ist die Datenbank blockiert, da die noch funktionsfähigen Instanzen nicht wissen, welche Seiten wiederhergestellt werden müssen, solange nicht alle Seiten gesperrt sind.

Das zeigt, dass jedes Datenbankprodukt bei einem Ausfall eine gewisse Zeit zur Wiederherstellung benötigt, so dass die Datenbank über einen bestimmten Zeitraum nicht verfügbar ist. Oracle RAC bietet keine schnellere Wiederherstellung als die HDR-Funktion von IDS.

Leistung nach der Wiederherstellung

Wir kennen bereits die Leistungsprobleme dieser Architektur: Die vier Maschinen erreichen nicht das Leistungsniveau der entsprechenden Einzelmaschine. Was geschieht nach der Wiederherstellung nach einem Ausfall? Der Betrieb erfolgt auf einer niedrigeren Leistungsstufe. Wir müssen uns auf eine solche Situation vorbereiten, um nicht ohne die entsprechende Leistung und Antwortzeit dazustehen, die für die geschäftskritischen Anwendungen erforderlich ist. Wir haben also zwei Möglichkeiten: Entweder wir konfigurieren die Systeme mit übergroßen Reserven, oder wir nehmen einige Anwendungen aus dem Betrieb.

Wenn Sie für Ihr Cluster eine Leistungsoptimierung vorgenommen haben, kann die Wiederherstellung nach einem Ausfall in eine Katastrophe münden. Stellen Sie sich vor, Sie hätten Ihr System so optimiert, dass jeder Knoten ein bestimmte Arbeitslast verarbeitet und nur ein Minimum an Blöcken zwischen den Maschinen kopiert werden muß. Was passiert beim Ausfall eines Knotens? Ein einzelner Knoten kann die Arbeitslast, die der ausgefallene Knoten verarbeitet hat, nicht bewältigen (außer wenn jeder Knoten eine Auslastung von 50 % oder weniger hat). Die Anwendung, die auf dem ausgefallenen

Knoten ausgeführt wurde, muss auf mehrere Knoten verteilt werden. Auf Grund dieser Umverteilung müssen die übrigen Knoten Daten untereinander kopieren. Die zusätzlichen Puffervorgänge auf den aktiven Knoten und die zusätzlichen Kopiervorgänge von Blöcken zwischen den Maschinen erhöhen den Overhead des gesamten Clusters. Dadurch wird die Gesamtleistung noch stärker beeinträchtigt. Das Ergebnis könnte so aussehen, dass Oracle RAC zwar betriebsbereit ist, die Arbeitslast der Produktionsanwendungen jedoch nicht bewältigt. Die Antwortzeit wäre dabei womöglich nur unwesentlich besser als bei der Nichtverfügbarkeit des Systems.

Wenn Sie RAC mit Blick auf Hochverfügbarkeit einsetzen, müssen Sie Ihre Umgebung überdimensionieren, um einen Knotenausfall und den zusätzlichen Overhead aufzufangen, wenn Sie nicht Gefahr laufen wollen, nach der Wiederherstellung in einer eingeschränkten Umgebung zu arbeiten.

Wie sieht es mit Aktiv-Aktiv aus?

Das am häufigsten genannte Argument für die Hochverfügbarkeit von Oracle RAC ist die Tatsache, dass dieses Produkt im Aktiv-Aktiv-Modus betrieben wird: Alle Knoten arbeiten gleichzeitig an derselben Datenbank. Dies ist nach meinem Eindruck in erster Linie ein gefühlsmäßiges Argument. Gegen Gefühle kann ich nichts machen, aber ich kann zahlreiche Punkte anführen, die zeigen, dass dies kein Vorteil ist.

In der gesamten Besprechung von RAC haben wir gesehen, dass die Zuweisung zusätzlicher Ressourcen erforderlich ist, um mit der Skalierbarkeit einer Einzelmaschine gleichzuziehen: 4 x 1 CPU ist nicht gleich 4 CPUs, und 4 x 1 GB Hauptspeicher ist nicht gleich 4 GB Hauptspeicher. Außerdem müssen weitere Ressourcen für den Umgang mit Ausfällen zugewiesen werden. All dies klingt für mich nach einer verkappten Aktiv-Passiv-Umgebung. Es kommt noch schlimmer.

Wir haben gesehen, dass eine Abfrage (eigentlich eine Anwendersitzung) auf einem speziellen Knoten ausgeführt wird. Diese Abfrage kann nur auf die Verarbeitungsressourcen eines einzigen Knotens zugreifen. Dies können wir in gewisser Weise als Aktiv-Passiv-Umgebung für die einzelnen Abfragen begreifen: ein Knoten aktiv, drei Knoten passiv. Wenn die Abfrage mehr CPUs und Hauptspeicher hätte nutzen können: Pech gehabt.

Schließlich müssen wir uns klarmachen, dass das Vorhandensein einer Standby-Maschine in der IDS HDR-Umgebung nicht bedeutet, dass die Hardware nicht für andere Aufgaben verwendet werden kann, wenn alles reibungslos läuft. Sie können beispielsweise die Berichterstellung auf die Standby-Instanz auslagern und damit Zyklen für die interaktiven Anwender verfügbar machen. Dadurch steigt letztlich die Anzahl der unterstützbaren Anwender, insbesondere in einer unterbrechungsfrei verfügbaren Umgebung.

Die Standby-Maschine hat Verarbeitungsleistung übrig, da sie nicht so viel zu tun hat wie der Primärserver. Sie können diesen Leistungsüberschuss für weniger wichtige Aufgaben einsetzen, die beim Ausfall der Primärmaschine unterbrochen werden können.

Warum also sollten Sie Ihre „passive“ Verarbeitung kaschieren, indem Sie sie auf mehrere Maschinen verteilen? Konzentrieren Sie doch besser alles auf einer Maschine, und nutzen Sie diese entsprechend. So können Sie die Verarbeitung auf Ihren Produktionssystemen

optimieren und sich dank der intelligenten Nutzung der übrigen Verarbeitungsleistung einen Wettbewerbsvorteil verschaffen.

RAC – Zusammenfassung

In diesem Abschnitt haben Sie die technischen Argumente für die Unterlegenheit von Oracle RAC als Lösung für Skalierbarkeit und Hochverfügbarkeit erfahren. Der Overhead für die Koordination zwischen den Knoten verringert die Skalierbarkeit erheblich. Zudem werden Ressourcen wie CPU und Hauptspeicher fragmentiert und damit ihre optimale Nutzung stark eingeschränkt.

Auch in Bezug auf die Hochverfügbarkeit ist RAC unterlegen. Das Produkt bietet keinerlei Vorteile gegenüber einer IDS-Lösung bei der Wiederherstellung. Außerdem vergeudet es Ressourcen, die besser zur Unterstützung der Unternehmensprozesse eingesetzt werden könnten.

Es kommt noch schlimmer: Eine RAC-Umgebung lässt sich schwerer verwalten, da sie ständig überwacht und optimiert werden muss.

Verwaltbarkeit

Ein zentraler Faktor für eine gute Verwaltbarkeit ist ein Datenbanksystem, das auf einer soliden Architektur gegründet ist und sich dadurch an den Bedarf von Produktionsanwendungen anpassen lässt. Dies umfasst die unkomplizierte Anpassung an die Anzahl der Anwender, die Anzahl der Abfragen und die Anforderungen dieser Abfragen. Die Multithread-Architektur von IDS bietet eine solche Grundlage für die dynamische Zuweisung von Ressourcen.

IDS lässt sich einfach konfigurieren, wenn man die Anzahl der auf dem System verfügbaren CPUs, den für das Datenbanksystem vorgesehenen Hauptspeicher sowie eine grobe Schätzung der Anzahl der Anwender berücksichtigt. Die Überwachung kann mit Server Studio, über eine Webschnittstelle (ISA), Befehlszeilen und über SQL-Abfragen an Systemkataloge erfolgen.

Für die Verwaltungsfreundlichkeit von IDS gibt es mehrere Gründe. Erstens vereinfacht die Architektur die Verwaltung von Verbindungen und Abfragen durch die bedarfsgesteuerte, dynamische Zuweisung von Ressourcen. Zweitens lassen sich die meisten Aufgaben leicht automatisieren, was dem Datenbankadministrator die Möglichkeit gibt, zu agieren, statt zu reagieren. So kann er mehr Zeit darauf verwenden, gespeicherte Prozeduren, Funktionen, Trigger, Leistungsanalysen und Anwendungsdesigns zu entwickeln, neue Funktionen zu prüfen und die optimale Nutzung zu ermitteln, diese Funktionen zu implementieren, Anwender zu schulen und natürlich Unterstützung verschiedener Art zu bieten.

Beispiele für die Verwaltungsfreundlichkeit sind Großkunden aus dem internationalen Einzelhandel, die Tausende von IDS-Instanzen von weniger als 20 Datenbankadministratoren verwalten lassen. Transportation Clearing House (TCH), ein anderer Kunde, meint dazu:

„Dank IDS konnten wir unseren Bedarf an Kundendienstpersonal und die zugehörigen Kosten erheblich senken. So konnten wir bei stabilen Personal- und Gemeinkosten eine Wachstumsrate von 500 % erzielen.“

Bei Oracle müssen Sie festlegen, wie viele Prozesse eine Instanz haben kann und wie viele Sitzungen (Clientverbindungen) diese Prozesse verarbeiten können. Darüber hinaus gibt es Parameter für die aktuelle Anzahl der gemeinsam genutzten Serverprozesse, die maximale Anzahl der gemeinsam genutzten Serverprozesse, die aktuelle Anzahl der Dispatcher und die maximale Anzahl der Dispatcher. All diese Werte stehen im Verhältnis zur Anzahl der auszuführenden Prozesse.

„In Oracle Database 10g wurden die Parameter in Basiskategorien und erweiterte Kategorien unterteilt. Die Administratoren können sich bei ihren täglichen Interaktionen auf 28 Basisparameter beschränken. Die erweiterten Parameter sind speziell geschulten Datenbankadministratoren vorbehalten und dienen dazu, das Verhalten der Oracle Datenbank an besondere Anforderungen anzupassen…“
Oracle Database 10g: The Self-Managing Database

Dass Datenbankadministratoren jeden Tag Interaktionen mit 28 Basisparametern ausführen müssen, klingt nicht gerade nach einer verwaltungsfreundlichen Datenbank, die obendrein noch als „selbstverwaltend“ angepriesen wird, doch für Oracle 10g ist dies bereits eine große Verbesserung.

Wir würden uns schnell in Details verlieren, wollten wir sämtliche Funktionen der Verwaltbarkeit von IDS 10.0 und Oracle 10g vergleichen. Deshalb wollen wir uns auf Sicherung und Wiederherstellung beschränken. Oracle bietet zu diesem Thema folgende Handbücher an:

- Backup and Recovery Basics (226 Seiten)
- Backup and Recovery Advanced User’s Guide (458 Seiten)
- Recovery Manager Quick Start Guide (26 Seiten)
- Recovery Manager Reference (332 Seiten)

Das Handbuch „IDS 10.0 Backup and Restore Guide“ hat dagegen nur 391 Seiten und enthält eine Beschreibung von zwei Dienstprogrammen für die Sicherung sowie der Funktion zur Wiederherstellung auf Tabellenebene mit Zeitangabe. Allein daran lässt sich schon ermessen, wie viel Aufwand nötig ist, um diesen Bereich des Datenbankmanagements zu begreifen.

Anwendungsentwicklung

IDS bietet Unterstützung für die Standardschnittstellen ESQL/C, ODBC und JDBC. IDS basiert auf einer Tradition der Softwareentwicklung. Am Anfang des Produkts stand die Sprache Informix 4GL, die bis heute auf der ganzen Welt oft genutzt wird. Inzwischen bietet IBM mit der Sprache EGL einen Weg zur Modernisierung. Außerdem kann der Kunde andere Produkte verwenden, die vollständig kompatibel mit Informix 4GL sind, beispielsweise die Produkte des Unternehmens Four J’s.

Hinsichtlich der Entwicklungstools für IDS setzt IBM auf die Unterstützung weit verbreiteter Tools. Bei Java sind dies Tools auf der Basis des Open-Source IDE Eclipse.

Außerdem unterstützt IBM die Integration in Tools wie Microsoft Visual Studio und die .Net-Umgebung.

Oracle behauptet zwar, sich an Standards zu orientieren, entwickelt aber gleichzeitig eigene Tools und definiert sogar eigene Typen in der Datenbank, anstatt die Standardtypen zu verwenden. Oracle rät beispielsweise von der Verwendung des Standardtyps `VARCHAR` ab:

> *„Verwenden Sie nicht den Datentyp `VARCHAR`, sondern stattdessen den Datentyp `VARCHAR2`. Zwar ist der Datentyp `VARCHAR` derzeit mit `VARCHAR2` identisch…"*
> Oracle 10g SQL Reference, Seite 2-10

Ein weiterer wichtiger Aspekt bei der Entwicklung ist der Bedienungskomfort bei der Nutzung der Datenbank. IDS ist führend in der Erweiterbarkeit von Datenbanken. Dadurch können die Systemarchitekten die Datenbank an ihr Geschäftsumfeld anpassen, anstatt ihre Konzeption auf die Datenbank zuschneiden zu müssen. Dieser Unterschied zwischen IDS und Oracle wird vor allem bei den erweiterten Funktionen deutlich. Eine räumliche Abfrage sieht bei IDS beispielsweise wie folgt aus:

```
SELECT A.Feature_ID
FROM A
WHERE
ST_Overlaps(A.shape,
  ST_GeomFromText(
    ' POLYGON (x1 y1, x2 y2, x3 y3, x4 y4 …) ',
    5     -- OpenGIS requirement
  )
);
```

Bei dieser Abfrage werden Standardtypen und -operationen entsprechend dem GIS-Standard verwendet, und die Steuerkomponente der Datenbank kann selbst das optimale Verfahren zur Lösung der Abfrage wählen.

Oracle behauptet, dieselben Funktionen bereitzustellen, einschließlich R-Tree-Indizes, wie oben beschrieben. Das Produkt muss jedoch eine weitaus komplexere räumliche Abfrage als IDS bereitstellen, in der proprietäre Typen und sogar Angaben zur Auflösung der Abfrage enthalten sind:

```
SELECT A.Feature_ID
FROM TARGET A
WHERE
sdo_relate(A.shape,
 mdsys.sdo_geometry(
   2003, NULL, NULL,
   mdsys.sdo_elem_info_array(1,1003,1),
   mdsys.sdo_ordinate_array(
     x1, y1, x2, y2, x3, y3, x4, y4, …
   )
 ),
 'mask=anyinteract querytype=window'
```

```
) = 'TRUE'
```

Wir sehen, dass für die Abfrage von Oracle mehr Informationen erforderlich sind als zur Definition eines Polygons. Außerdem werden zusätzliche Angaben zur Art der Auflösung der Abfrage benötigt. Dadurch erhöht sich die Komplexität für die Entwickler. Im Ergebnis bedeutet dies längere Entwicklungszeiten, eine schwierigere Wartung und potenziell mehr Probleme bei der Leistungsoptimierung.

Über die Anwendungsentwicklung könnte man ganze Bücher schreiben. Das obige Beispiel zeigt die unterschiedlichen Ansätze: IDS bemüht sich um Einfachheit und gute Integration, während Oracle die Funktionalität ohne Berücksichtigung des Bedienungskomforts bereitstellt.

Zusammenfassung

Von einer höheren Warte betrachtet, sehen alle Datenbanksysteme gleich aus. Sie erledigen ihre Arbeit. Die entscheidende Frage dabei ist jedoch: zu welchem Preis? Unternehmen verschaffen sich einen Wettbewerbsvorteil, indem Sie ihre Prozesse auf sämtlichen Ebenen des Unternehmens optimieren. Dies umfasst den Umfang der für die Ausführung der geschäftskritischen Anwendungen erforderlichen Hardware, die Anzahl der zur Verwaltung dieser Systeme benötigten Mitarbeiter und die Verfügbarkeit dieser Systeme.

Im vorliegenden Dokument haben wir gesehen, dass Informix eine solidere Grundlage für die Optimierung der Leistung von Computersystemen bietet. Dies führt zu einer Steigerung der Leistung und Skalierbarkeit.

Was die nicht so leicht abgrenzbaren Bereiche Zuverlässigkeit und Verfügbarkeit angeht, so haben wir gesehen, dass IDS ein bewährtes Produkt ist, das keinem anderen nachsteht. Die Funktion von IDS zur Wiederherstellung nach einem Katastrophenfall mit hoher Verfügbarkeit (HDR) ist de facto allen von Oracle angebotenen Lösungen überlegen, einschließlich RAC und Data Guard. Wir konnten sogar aufzeigen, dass Oracle RAC weder für Skalierbarkeit noch für Hochverfügbarkeit geeignet ist.

Was das Systemmanagement angeht, so wurde IDS für eine automatische Verwaltung konzipiert, die sich ausgezeichnet für Umgebungen eignet, deren Datenbankadministratoren über eingeschränkte Erfahrungen verfügen.

Mit IDS 10 an Stelle von Oracle 10g können Sie Ihre Computersysteme besser optimieren und die Effizienz Ihres Unternehmens steigern. Dies bringt echte Wettbewerbsvorteile, die Ihnen helfen, in Ihrer Branche die Führung zu übernehmen oder zu halten.

Referenzen

- *IBM @server p5 570 Technical Overview and Introduction*
  IBM Redbook
- *The Design of the Unix Operating System*
  Maurice J. Bach ISBN 0-13-201799-7

- *The Design and Implementation of the 4.3BSD UNIX Operating System*
  Samuel J. Leffler u. a. ISBN 0-201-06196-1
- *A Guide to Multithreaded Programming, Thread Primer*
  Bill Lewis, Daniel J. Berg ISBN 0-13-443698-9
- *IDS 10.0 Documentation*
- *Oracle10gR2 Documentation*
- *Oracle Database 10g: The Self-Managing Database*
  (Oracle White Paper)