InMemory eine sinnvolle Erweiterung zur Datenanalyse?

wir betreiben für den Kunden eine Exadata Quarter Rack X3-2. Es werden 8 Datenbanken mit jeweils einzelnen Instanzen auf den beiden DB-Server betrieben. Es kommen neue Applikationen hinzu, auf der anderen Seite wachsen die alten. Aus diesem Grund war in der Vergangenheit Datenwachstum immer ein Thema. letztendlich führte das auch zu einer Erweiterung der Umgebung durch ein Quarter Rack Storage Expansion X4, wobei zunächst nur zwei Cellserver aktiviert wurden.

Im Jahr 2015 kamen dann neben den Platzproblemen dann auch Performanceprobleme hinzu. wir haben die uns bekannten Applikationen analysiert, konnten aber keinen dramatischen Anstieg in Form von CPU- oder IO-Ressourcen feststellen. Trotzdem lief vor allem der erste Datenbankserver viele Stunden des Tages unter Volllast.

Was war die Ursache? Der Kunde hatte eine neue Applikation auf eine produktive Datenbank gebracht, in der Verbraucherdaten auf Tagesbasis verarbeitet und gesichert wurden. Bisher waren nur Daten auf Monats- und Wochenbasis verarbeitet worden. Das Business aber wollte Daten auf Tagesbasis analysieren. das gestartete Projekt sollte die Tagesdaten erst einmal nur für 4 Wochen halten, aus den 4 Wochen wurden 3 Monate, aus den 3 Monaten wurden 6, aus den 6 Monaten wurden Performanceprobleme.

Die Tagesdaten wurden per ETL in der Datenbank ausgewertet und gesichert. Die Anwender griffen per Business Objects auf die Daten zu und wurden in der Kombination der abgefragten Spalten nicht eingeschränkt. Die Daten wurden nach Zeit und Region partitioniert, und per HCC komprimiert, aber selten wurden diese beiden Partitionskriterien in den Abfragen genutzt. Das Ergebnis waren in der Regel Full Table Scans auf einer ca. 5 TB großen Tabelle, oft verknüpft mit Dimensionstabellen. Mit ca.bis zu 3 parallelen Abfragen kam der DB-Server noch in angemessener Zeit zum Ergebnis. Aber mehrere parallele Anfragen endeten in CPU-Performanceengpässen.

Nach einer Analyse der Abfragen haben wird dann versucht, auf den 2 meist genutzten Spalten einen Index aufzubauen. Ein Index benötigte knapp 1 TB Platz, aber durch jeden Index verlangsamte sich der ETL-Prozess um je 2 Stunden. Indizes waren also keine Lösung.

Um die „Alt-„Applikationen zu entlasten, wurden im ersten Schritt die Daten auf die Testumgebung, eine Eighth Rack Exdata, verlagert. Die Performancezahlen waren hier ähnlich. Es waren weniger CPU-Ressourcen vorhanden, aber mehr oder weniger keine „Konkurrenten“. Von der Testumgebung sind die Daten dann auf eine Analytics-Appliance eines anderen Herstellers migriert worden, weil die Testumgebung auf die Dauer auch nicht genügend Ressourcen hatte.

In der Zwischenzeit hatte der Kunde beschlossen, aus der Gesamtmenge der Daten bestimmte Sichten herauszufiltern und in kleineren Applikationen den Kundengruppen zur Verfügung zu stellen. Gleichzeitig mit dieser Änderung der Herangehensweise sollte auch die Antwortzeit der Datenabfrage minimiert werden. Jedes einzelne Dashboard hatte 10 Sekunden als Limit in der Antwortzeit. Im ersten Schritt musste der ETL-Prozess neu implementiert werden.

In diesem Zusammenhang kam die InMemory-Option der 12c ins Spiel. dazu wurde auf der Testumgebung eine Instanz mit 175 GB SGA aufgebaut und dort eine Applikation entwickelt, die als Frontend spezielle Datenanalysesoftware nutzte. Nach erfolgreichen Tests wurde die Erweiterung der Produktion geplant um umgesetzt. Die DB-Server wurden jeweils um 256 GB Hauptspeicher erweitert und auf einem der DB-Server eine Instanz mit 240 GB SGA (210 GB InMemory-Size) aufgebaut. In der Zwischenzeit wurde die Applikation auf der Testumgebung produktiv.

Parallel dazu wurden die Daten dann parallel in die (produktive) Test- als auch in die Proditions-DB geladen. Die Anwender beklagten sich dann über eine „langsamere“ Produktions-DB. Darauf führten wir intensive Performanceuntersuchungen durch. Die Anwender stellten Sequenzen mit Abfragen zusammen, die wir auf beiden Umgebungen hinsichtlich Ressourcenbedarf und Ausführungsplänen miteinander verglichen haben. Hierbei viel auf, dass die Produktions-DB häufig dynamisch Statistikdaten

(dynmamic_sampling) sammelte. Als Maßnahme haben wir Histotgramme auf allen Spalten, auf beiden Umgebungen angelegt, wodurch sich die Ausführungszeiten in etwa anglichen.

Parallel haben wir dann Lasttests in keinerem Maßstab durchgeführt. Ungefähr 10 Anwender führten gleichzeitig Dashboard-Anfragen auf unterschiedlichen Datenbeständen durch. Dabei fiel dann Abfragen auf der Datenbank auf, die sich niemand erklären konnte, die aber über Minuten Last auf der DB und damit auf dem Server erzeugten. Nach Analysen konnte die Datenanalysesoftware als verursacher ermittelt werden und das Absetzen derartiger Statements eliminiert werden.

Im nächsten Schritt haben wir dann die Testsequenz mit unterschiedlichen Einstellungen von Datenbankparametern getestet. Im Wesentlichen sind her die Parameter Richtung Vektorverarbeitung, neu mit der 12c zu Verfügung gestellt, zu nennen.

Alle Anfragen laufen heute parallel ab, wobei der Parallelitätsgrad manuell auf Tabellenebene eingestellt ist. Wir haben auch Tests mit parallel_degree_policy=AUTO getestet, die Ergebnisse aber brachten keine positiven Verbesserungen, so dass wir heute mit manueller Parallelitätssteuerung arbeiten.

An einem Statement haben wir uns auch „die Zähne ausgebissen“. Das Statement zeichnete sich durch folgenden Konstrukt in einer Joinbedingung aus:

ON ((spalte 1 = spalte 2) OR ((spalte3 is not null) and (spalte 4 is not null))

Dies ließ der Optimizer immer mit einem NESTED LOOP ausführen, was je nach zu verarbeitender Menge dann auch zu nicht akzeptablen Ausführungszeiten führte. Hier haben wir bis heute keine Läösung.

In der Summe liegen heute die Abfragen unter einer Sekunde, meist im Millisekundenbereich. Insgesamt zeigt die Performanceauslastung im OEM eine Nutzung von weniger als eine CPU über den Tag an. Die Ausnahme zeigt der tägliche Datenload in den frühen Morgenstunden von 4 CPU über ca. 2 Stunden. Aktuell haben wir ca. 95 GB Hauptspeicher durch diese Applikation belegt.

Zurzeit testen wir die zweite Dashboardanalyse, die parallel zur ersten gerade produktiv gegangen ist. Auch hier haben Performancetests gemacht und haben bei einigen Statements ein negatives Verhalten in der Vektorverarbeitung festgestellt. Die Parameter haben wir dann auf Sessionebene angepasst.

Von der anfangs eingesetzten langsamen „eierlegenden Wollmilchsau“ sind wir auf hochperformante Datenanalysen auf kleineren Datenbeständen geschwenkt. Eine Frage bleibt heute für mich noch unbeantwortet: hätte es eine Lösung für die „eierlegende Wollmilchsau“ geben können? Sicherlich nicht bezüglich der geforderten Antwortzeiten, aber mit mehr Flexibilität Richtung Datenanalysemöglichkeiten?