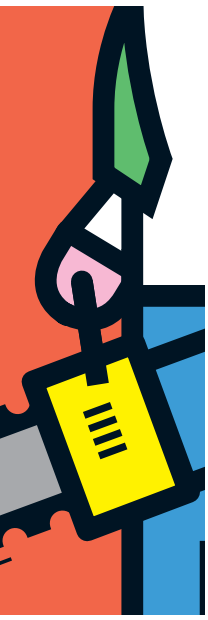


Op zoek naar de juiste balans tussen beschikbaarheid en TCO

ILM in een database-omgeving

Toon Loonen en Arwin van den Bos

De kosten van storage van zeer grote databases kunnen hoog oplopen. Terwijl we voor 100 euro al een 200 tot 500 GB interne of USB disk kopen voor onze PC thuis, zijn de kosten voor de kale opslagruimte voor een database in een professionele omgeving een orde van grootte (globaal een factor 10) hoger.

Op een SAN, waar deze ruimte meer flexibel beschikbaar wordt gesteld en ook voor redundantie en caching wordt gezorgd, is dit nog een orde van grootte hoger. Inclusief het beheer (backup- en restore-faciliteiten) en de omgeving (gebouw, elektriciteit, koeling etcetera) kunnen deze prijzen oplopen tot ruim boven de 100.000 euro per TB per jaar, dit ook afhankelijk van de beheervorm (eigen beheer of uitbesteding) en het contract (support 24x7 of kantooruren, etcetera). Dit is ruim een factor 1000 hoger dan de prijs van het schijfje dat we bij de PC boer op de hoek kopen.

Daarnaast groeit de hoeveelheid opgeslagen data jaarlijks enorm. Beide zorgt dit voor zeer hoge kosten voor de dataopslag bij databases van enige omvang, zeg groter dan 1 TB.

Information Lifecycle Management (ILM) kan helpen om deze kosten te beheersen. Het doel van ILM is het vinden van de juiste balans tussen de beschikbaarheidseisen van de gegevens en de kosten van de opslag, beveiliging en beheer.

Oplossingen

Diverse hardware-leveranciers, zoals HP, IBM en EMC, leveren complete ILM-oplossingen voor grote databases of rekencentra. De gewenste opslag wordt hierbij ingedeeld in drie categorieën:

- Online storage (OLS): de schijfruimte waaraan de hoogste eisen worden gesteld wat betreft snelheid (voor lezen en schrijven) en beschikbaarheid/betrouwbaarheid (door middel van kwaliteit van de schijven en/of redundantie/mirroring).
- Nearline Storage (NLS): de schijfruimte waaraan de eisen voor snelheid en beschikbaarheid/betrouwbaarheid wat minder hoog zijn.
- Offline storage/archive (in het algemeen tape of CD/DVD maar eventueel ook schijfruimte, NLS, waar gegevens buiten de database bewaard worden): de opslag die pas na bijvoorbeeld een minuut (met een taperobot) of zelfs een uur (bij handmatige handelingen en het opzoeken van een tape of CD in een kast) beschikbaar moet zijn, bijvoorbeeld in verband met wettelijke bewaartermijnen of historische overzichten.

Voor een specifieke situatie was de kostenverhouding tussen tape, NLS en OLS: 1 op 10 op 30. De bedragen en cijfers in dit artikel moeten als een globale indicatie worden gezien. Het gaat hier in eerste instantie om de verhoudingen en de mogelijkheid om op deze kosten te besparen.

Het snelheidsverschil tussen NLS en OLS was bij de genoemde situatie globaal:

- een factor 1 op 2 voor random lezen en schrijven;
- een factor 1 op 1,2 voor sequentieel lezen en schrijven.

Het is dus heel belangrijk en kostenefficiënt om te kijken welke bestanden op welk type medium worden opgeslagen. De algemene regel is hierbij: hoge snelheid vereist, veel wijzigingen, random benaderen, dan OLS, of anders NLS.

Uitwerking

Dit wordt concreet uitgewerkt, met voorbeelden en terminologie van een Oracle database; dit is gemakkelijk te vertalen naar elk ander relationeel database product. Het mag duidelijk zijn dat we de database zelf op OLS zullen plaatsen. Ook de programmatuur (relatief een kleine hoeveelheid op schijf) en de online logfiles komen op OLS. Verder kunnen andere relatief kleine bestanden, zoals bestanden met initialisatieparameters, alertlogs, auditlogs en andere logging, zonder problemen op OLS worden geplaatst, omdat de ruimte en dus kostenbesparing erg klein is als deze bestanden op NLS worden gezet. De Oracle archive files komen natuurlijk op NLS. Als de backup eerst naar schijf wordt geschreven en van schijf naar tape, dan komt de backup ook op NLS.

OS/UNIX bestanden, zoals rapporten uit de database en inter-

face files (Oracle import/export, XML, CSV of enig ander formaat) die worden ingelezen in de database of vanuit de database naar andere systemen gestuurd worden, kunnen ook goed op NLS staan. Dit geldt ook voor archiefbestanden met gegevens die buiten de database op schijf worden bewaard, bijvoorbeeld PDF, CSV of XML files met financiële gegevens (facturen) die volgens wettelijke voorschriften vijf jaar of langer bewaard moeten worden. Hierbij kan zelfs overwogen worden om deze na bijvoorbeeld een jaar van NLS door te schuiven naar CD of tape, zie ook [1].

Een voorbeeld

In de IT wordt vaak de 20-80 regel genoemd: 20 procent van het werk doe je voor 80 procent van de processing en andersom. In een telecom-omgeving heb ik dit wat extremer gezien: in een database zat 99 procent van de gebruikte ruimte in 1 procent van de tabellen. Het betrof hier met name CDR's: de gegevens over de telefoongesprekken. Deze 1 procent van de tabellen is natuurlijk interessant om te partitioneren. Deze gegevens worden na een korte tijd weinig meer gelezen en zelden gewijzigd.
Door de tabellen waarin deze gegevens staan te partitioneren op bijvoorbeeld maandelijkse partities, kunnen we na enkele maanden de betreffende partities van OLS naar NLS verplaatsen en de betreffende tablespace(s) op read only zetten. Vervolgens maken we een backup van deze tablespaces.

Stel dat we drie maanden gegevens op OLS bewaren en 1 jaar op NLS bij een database van 5 TB. Daarna worden de gegevens nog enkele jaren buiten de database gearchiveerd op tape of CD. Stel ook dat 1 TB 100.000 euro per jaar kost op OLS en eenderde daarvan op NLS. De besparing door NLS in plaats van OLS te gebruiken is dus tweederde maal 100.000 euro per TB per jaar.
Op OLS hebben we dan de 1 procent ruimte van de 99 procent 'andere tabellen' plus eenvijfde deel van de grote gepartitioneerde tabellen. Dit is ongeveer 1 TB. De 12 maanden op NLS gebruiken dan nog 4 TB. De besparing is dan vier maal tweederde maal 100.000 euro; is 266.666 euro per jaar. Als ook een backup op disk naar NLS in plaats van OLS wordt geschreven, kan de besparing gemakkelijk oplopen naar een half miljoen euro per jaar. Zie ook afbeelding 1. Als de tablespaces op NLS ook op read-only staan en niet met de periodieke backup meegenomen worden, dan kan op de doorlooptijd van de backup ook 80 procent bespaard worden. Voor databases van deze omvang is dit een aanzienlijk voordeel!
Dit voorbeeld toont ook dat deze acties alleen zinvol zijn bij databases met een gezamenlijk ruimtebeslag van 1 TB of hoger. Bij een database van 5 GB is de besparing 1/1000 van het bovenstaande bedrag, dus zo'n 500 Euro. Het is dan niet meer economisch rendabel om over ILM na te denken. Ook de besparing op de doorlooptijd van de backup is voor een dergelijke database meestal niet relevant. Het op tijd schonen of archiveren blijft natuurlijk altijd rendabel.

In sommige systemen worden, naast de bekende datum-, numerieke en alfanumerieke gegevens ook grotere kolomtypes of objecten gebruikt: Character large object (CLOB) en Binary large object (BLOB). Bijvoorbeeld:
- CLOB: XML-documenten, lange teksten zonder opmaak;
- BLOB: Pasfoto van medewerker, Word- of Excel-documenten, gescande documenten.

Deze LOB's kunnen in de database worden bewaard (als BLOB of CLOB) maar ook als bestand in een UNIX file system waarbij in de database alleen de directory- en file-naam worden opgeslagen en de applicatie deze bestanden buiten de database om van schijf leest. In het laatste geval kunnen deze bestanden ook weer op NLS staan. Maar ook als de bestanden als LOB-kolommen in de database zelf staan, is het (in Oracle) mogelijk (en goed gebruik) om deze kolommen in een eigen tablespace te plaatsen. Deze tablespace kan vervolgens weer op NLS worden geplaatst als dit voor de performance van het systeem acceptabel is.

Partitioneren van grote tabellen

Zeer grote tabellen met transactiegegevens, zoals orders, facturen, betalingen of Call Detail Records (CDR's) in een telecom-omgeving, worden vaak gepartitioneerd opgeslagen. De tabel bevat dan een partitie voor elke maand met de gegevens voor de betreffende maand. Partities kunnen in Oracle aan een tablespace worden gekoppeld en deze tablespaces worden weer (via de datafiles) op schijf, dus op OLS of NLS geplaatst. Voor de meest actuele maanden (zeg de meest recente drie maanden) kan men kiezen voor opslag op OLS, omdat deze gegevens veel wijzigen. Dit zijn voornamelijk toevoegingen, maar fysiek in de database zijn dit wijzigingen op de datablocks in de tabel. De oudere maanden kunnen telkens doorgeschoven worden naar NLS waar ze dan nog bijvoorbeeld 12 of 24 maanden worden bewaard. Daarna, tot de wettelijke of functioneel wenselijke bewaartermijn is verstreken, kunnen deze worden gearchiveerd op tape of CD/DVD.
Het verplaatsen van partities naar een tablespace op een ander medium kan in Oracle met het statement 'Alter table CDR move partition CDR_2006_12 tablespace TS_NLS_2006_12'. Dit statement zal fysiek de partitie naar de andere tablespace kopiëren en kost dus erg veel tijd. Het is niet acceptabel om elke maand alle partities een tablespace verder naar achteren te verplaatsen. Een goede procedure en duidelijke naamgeving zijn daarom zeer gewenst. Deze extra beheerwerkzaamheden kosten ook tijd en dus geld. De kosten hiervan moeten van de besparing worden afgetrokken.

Backup naar tape of disk

De backup (zie [2]) van een zeer grote database wordt vaak direct naar tape geschreven. De kosten van de extra storage die nodig is om de backup eerst naar disk te schrijven en dan van disk naar tape zijn natuurlijk erg hoog. Maar er zijn diverse redenen om toch de backup eerst naar disk te schrijven: de

snelheid van het schrijven naar disk is hoger, dus de doorlooptijd van de backup en een eventuele restore is lager; backup naar schijf is vaak eenvoudiger; Oracle adviseert deze werkwijze. Hierbij worden de backup bestanden natuurlijk op NLS geschreven. Nadat de database backup klaar is kan een tweede procedure gestart worden om deze backup bestanden weer naar een eigen tape pool te schrijven. Dit als extra beveiliging, voor het bewaren van een kopie buiten het rekencentrum of voor een terugval naar een versie van de database van een of twee weken geleden.

Read-only tablespaces

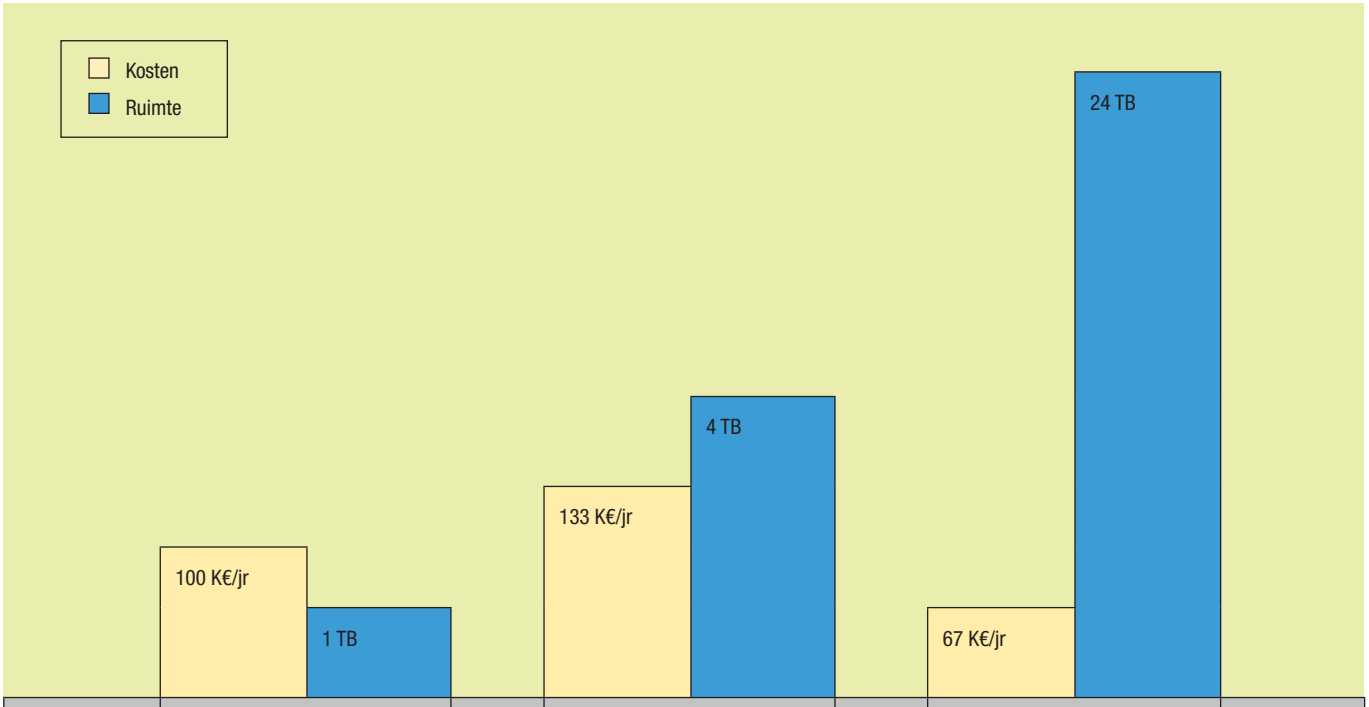
Bij gebruik van partities met oude gegevens is er nog een andere reden om de backup eerst naar disk te schrijven. Als een tablespace alleen partities bevat met gegevens die niet meer kunnen of mogen wijzigen, dan kan de tablespace op READ-ONLY gezet worden. Bij een backup naar disk is het dan mogelijk om eenmalig van deze tablespace een backup naar disk te maken. Bij de reguliere (dagelijkse of wekelijkse) full backup van de database hoeft van deze tablespaces niet meer een backup gemaakt te worden.
Zou de backup naar tape gemaakt worden dan is het niet praktisch om de read-only tablespaces over te slaan. Een tape is immers een sequentieel medium. Hierop staan de backups van de datafiles na elkaar. Zou een tape vrij gegeven worden (hergebruik of overschrijven van de tape na drie generaties backup of na drie weken), dan wordt ook de kopie van de read-only datafiles vrijgegeven en overschreven, dus moet deze opnieuw op tape worden gezet. Het vrijgeven van een deel van de tape is niet mogelijk of praktisch.
Een alternatief is nog om de read-only tablespaces op een separate set tapes te bewaren. Maar dat maakt de backup-procedure ingewikkelder en bewerkelijker en het laatste wat je als DBA wilt is een onbetrouwbare backup-procedure (zie ook [2]).

Archiveren en schonen van de database

Een significante besparing kan verder verkregen worden door goed te kijken naar archivering en schonen van oude gegevens. Hiermee kan de besparing nogmaals flink oplopen. Dit archiveren en schonen moet in het ontwerp van het systeem worden meegenomen om effectief uitgevoerd te kunnen worden. Daarbij moet ook beschreven worden hoe de gearchiveerde gegevens opnieuw kunnen worden bekeken. Zie verder [1].

Hot Standby systeem op NLS

Discutabel is of een Hot Standby systeem[1] ook op NLS geplaatst



Type	OLS	NLS	Offline / Tape / CD
Gebruik	- 3 maanden CDR's - Kleine tabellen - OS, programma's - Kleine OS bestanden - Parameterfiles, logs	- 12 maanden CDR's - BLOB's en CLOB's - Rapporten - Interface bestanden - Oracle archives - Backups op disk	- 60 maanden CDR's - Historische bestanden - Backup op tape

Afbeelding 1: Verhouding kosten en ruimte voor OLS, NLS en offline storage bij een kostenverhouding 30:10:1 en ruimteverhouding 1:4:24.

Solid state disks en in memory database

Het rijtje OLS, NLS en offline storage kan aan de voorkant nog aangevuld worden met het gewone computergeheugen. Naast de database buffers zijn er 2 mogelijkheden om een database geheel of gedeeltelijk hierin op te slaan: solid state disks (op basis van SDRAM, zie [5]); in memory database (bijvoorbeeld Oracle Times Ten, zie [3] en [4]). Bij het eerste worden de datafiles van een gewone relationele database opgeslagen op een unit die een gewone schijf of een SAN emuleert. Het kan dus met elk database-product worden toegepast. Bij het tweede alternatief staan de tabellen van de database gewoon in het programmageheugen van de database server.

Deze alternatieven zijn (mits goed toegepast) duidelijk veel sneller dan zelfs OLS, maar ook veel duurder. Voor een TB database is dit geen alternatief. Een in memory database komt het beste tot zijn recht bij een kleine maar zeer zwaar gebruikte database. Verder kan Oracle Times Ten gebruikt worden als een front-end voor een gewone Oracle database.
We kunnen dan, voortgaande op het voorbeeld in het kader 'Een voorbeeld': de zwaarst gebruikte kleinere tabellen (0,1 procent van de ruimte) in Oracle Times Ten plaatsen; de andere kleine tabellen en de recente partities/maanden van de grote tabellen op OLS plaatsen; de oudere partities/maanden van de grote tabellen op NLS plaatsen; de nog oudere maanden daarna archiveren op tape of CD.
Bij gebruik van een solid state disk kunnen de tabellen die het meest gebruikt of gewijzigd worden natuurlijk in een eigen tablespace geplaatst worden. Deze tablespace wordt dan op de solid state disk geplaatst. De andere tablespaces komen op OLS of NLS. Of dit erg veel voordeel oplevert is twijfelachtig, omdat het DBMS het overgrote deel van de zwaar gebruikte tabellen toch al in zijn eigen data cache zal houden.

Oracle Times Ten heeft nog enkele voordelen ten opzichte van een database op solid state disks. Omdat Oracle Times Ten weet dat de gehele tabel in het geheugen staat, is er geen software nodig om te controleren of een datablock in memory staat of nog van disk opgehaald moet worden. Er verdwijnt dus veel overhead. Dit en andere aspecten maakt dat de software eenvoudiger en dus veel sneller is. Oracle Times Ten claimt tien keer sneller te zijn dan een database waarbij de data al geheel in cache zijn opgeslagen. Terwijl bij een gewone database nooit voor een individuele query een zekere responstijd kan worden gegarandeerd, omdat de database op dat moment net erg druk kan zijn met andere vragen of met eigen beheer, zegt Oracle dat dit met een Oracle Times Ten database wel kan.

kan worden. In de meeste gevallen zal voor de standby systemen dezelfde verdeling tussen OLS en NLS worden gekozen als voor het primaire systeem. Alleen als het standby systeem, bij uitval van het primaire systeem, tijdelijk wordt gebruikt en er weer zo snel mogelijk wordt teruggeschakeld naar het primaire systeem en als bij deze tijdelijke situatie een verminderde performance acceptabel is, dan kan een Hot Standby systeem geheel op NLS worden geplaatst.

Conclusie

De hoeveelheid gegevens die in databases wordt opgeslagen groeit (verdubbelt) jaarlijks. Er worden meer gegevens in de databases opgeslagen. Er worden grotere objecten (BLOB's en CLOB's) in de database opgeslagen. De gegevens blijven vaak langer in de database staan.

Gegevens staan steeds vaker in meer databases, bijvoorbeeld door het introduceren van een datawarehouse. Daarnaast groeit het aantal zeer grote (meer dan 1 TB) databases nog harder.
De kosten van de opslag van deze gegevens dreigen de pan uit te rijzen. Het toepassen van de principes van ILM kan bij deze zeer grote databases tot een significante kostenbesparing leiden terwijl de performance en beschikbaarheid van de database hieronder niet significant achteruit gaat. Dit kan door:

- De meest gebruikte kleinere tabellen (minder dan 1 procent van de ruimte) in main memory te plaatsen via caching of een in memory database als front-end voor een gewoon RDBMS;
- De andere kleine en de middelgrote tabellen en de recente partities van de zeer grote tabellen op OLS te plaatsen;
- De tablespaces met LOB kolommen en tablespaces met oudere partities van deze zeer grote tabellen op NLS te plaatsen;
- De nog oudere partities daarna te archiveren op tape of CD.

Hierbij zijn er wel extra beheerkosten. Deze moeten van de besparing worden afgetrokken. Bij kleinere databases (enkele Gigabytes) is de besparing maximaal enkele honderden euro's. Het is dan niet meer economisch rendabel om de database te verdelen over OLS en NLS. Maar op tijd schonen of archiveren blijft natuurlijk altijd rendabel.

Noot

1. *Een hot standby systeem neemt de taken van een ander systeem 'direct' over bij uitval van dat andere systeem.*

Literatuur

1. *Loonen. Wat archiveren en op welk formaat? Database Magazine 2003/1.*
2. *Loonen. Backup procedures en continuïteit van database services. LAN Magazine 2007/9.*
3. *Van der Lans, R.F., Tijd was rijp voor veelbelovend TimesTen. Database Magazine 2001/6.*
4. *Oracle Times Ten database: www.oracle.com/database/timesten.html.*
5. *Solid state disks: http://en.wikipedia.org/wiki/Solid_state_disk.*
6. *ILM volgens IBM: www.redbooks.ibm.com/ en zoek op ILM.*
7. *ILM volgens HP: www.hp.com en zoek op ILM.*
8. *ILM volgens EMC: www.emc.com en zoek op ILM.*

Toon Loonen en Arwin van den Bos zijn werkzaam bij Capgemini.