Explosief groeiende databases vereisen performanceverbetering

'Solid state disks' herontdekt als databaseversnellers

Gert Brouwer

Inmiddels vrijwel volledig uit het mainframe verdwenen, worden 'solid state disks' (SSD's) steeds vaker toegepast in de zogenaamde open systemen, getuige onder meer enkele recent verschenen rapporten van IDC. Geen wonder: SSD's kunnen enorme performancevoordelen bieden. En de nadelen lijken betrekkelijk.

In de laatste decennia van de vorige eeuw deed de *solid state disk* (SSD), een volledig elektronische schijf zonder alle mechanische vertragingen, zijn intrede in de (IBM-)mainframemarkt. Vertragende factoren als 'seek'-tijden en 'latency' zijn tot nul gereduceerd. De snelheden van deze disks zijn wat dat betreft gelijk aan die van intern RAM-geheugen; alleen de externe koppeling zorgt voor vertraging ten opzichte van intern geheugen, en die vertraging is minimaal. Plug-compatible leveranciers, zoals Memorex, StorageTek, Hitachi en EMC deden in de jaren tachtig goede zaken met hun SSD-toepassingen.

De toenmalige SSD-systemen werden voornamelijk ingezet als *paging device*, als alternatief voor de kostbare en beperkte geheugenuitbreidingen van het eigenlijke mainframe. Op deze manier kon menig gebruiker zijn dure mainframe-uitbreiding uitstellen. Bovendien waren de SSD-systemen nauwelijks aan veroudering onderhevig, zodat zij meerdere generaties mainframes meegingen.

Een beter behoud van investering kon men zich niet indenken. Bijkomende, zeker niet onbelangrijke factor, was dat een mainframe-uitbreiding meestal resulteerde in een gepeperde rekening van zowel de leverancier van het besturingssysteem als die van de applicatiesoftware. Saillant detail: de SSD van EMC groeide uit tot het later zo succesvolle Symmetrix Disk-systeem.

WAAROM SSD?

In iedere vendor-presentatie wordt het uitvoerig getoond, vaak aan de hand van rapporten van Gartner en IDC: de datagroei is explosief. Een gegevensaanwas van 75 tot 100 procent per jaar is voor de meeste organisaties geen uitzondering. Ook aan databases gaat deze groei natuurlijk niet voorbij. Een verdubbeling van de database-omvang is eerder regel dan uitzondering.

Deze ontwikkeling leidt vroeg of laat tot performanceproblemen. De meeste grote serverleveranciers weten wel raad met de problemen van de klant. Zij verkopen graag een aantal extra processoren, extra geheugen of liefst nog een groter systeem. In enkele gevallen levert dit het gewenste resultaat op. In een groot aantal gevallen draagt het echter niet of nauwelijks bij aan verbetering van de performance van de database, omdat de flessenhals aan de I/O-kant zit.

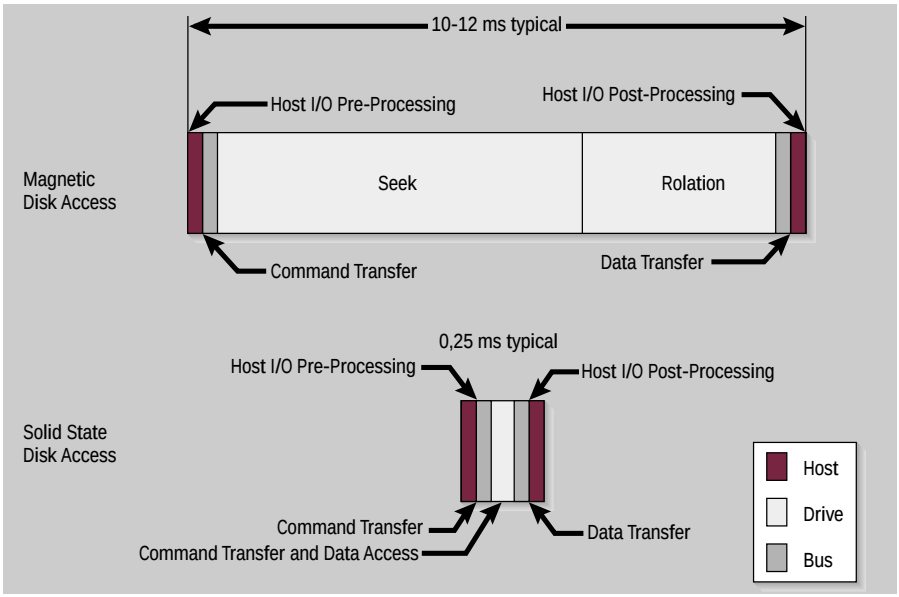
De invloed van I/O-bottlenecks op de

Wat is een solid state disk?

Een solid state disk (SSD) is in feite niets meer dan 'georganiseerd chipgeheugen', dat zich aan de cpu manifesteert als een normale disk. Er vinden geen mechanische bewegingen plaats van draaiende schijven en bewegende koppen. Enkele microseconden na het verzoek van de host worden de gewenste data reeds aangeleverd. Een SSD is minimaal driehonderd maal sneller dan de snelste mechanische disk (zie figuur 1). Een Unix-filesysteem kan op een SSD worden aangemaakt zonder dat het besturingssysteem enig verschil opmerkt.

Data in een chipgeheugen hebben echter een beperkte levensduur; zodra de voedingsspanning wegvalt, verdwijnen de gegevens. SSD-leveranciers hebben dit opgelost door hun systemen te voorzien van een 'battery-backup' en een conventionele schijf. Zodra de spanning wegvalt, schakelt de SSD over op zijn interne batterij en worden de gegevens weggeschreven van SSD naar conventionele schijf. Komt de netspanning weer terug, dan worden de data teruggezet op de SSD.

Omdat een SSD is opgebouwd uit RAM-geheugen en de apparatuur nog niet in grote aantallen wordt geproduceerd, zijn SSD's aanzienlijk duurder dan de conventionele schijven. De inzet ervan kent dus een economische grens. Het is daarom van belang dat de systeemarchitect of -integrator bepaalt wat op de SSD thuishoort.



FIGUUR 1: ACCESS-TIJDEN SSD TEN OPZICHTE VAN CONVENTIONELE DISK.

databaseperformance is drieërlei; de mate waarin deze aspecten belangrijk zijn, is uiteraard afhankelijk van het type database en de applicatie:

- transactielogs;
- temporary database storage;
- tabellen en indexen.

Transactielogs

Transactielogs zijn over het algemeen het meest gevoelige gebied voor contentie in een OLTP-systeem. Alle modificaties, inserts, updates en deletes lopen door de transactielog van de database. Deze wordt hiermee flessenhals nummer één in de databaseperformance. De snelheid waarmee een transactie naar de transactielog wordt weggeschreven is de snelheid waarmee de gegevens zijn te verwerken. De snelheid waarmee de log wordt weggeschreven, wordt begrensd door het device waarop het systeem die wegschrijft.

Temporay database storage

Temporary database storage -zoals gebruikt bij specifieke sorts in een SQL-query, waarbij 'order by' of 'group by' als klasse wordt gespecificeerd- is een ander gebied waarin potentieel I/O-contentie optreedt. Temporary storage kan ook worden gebruikt door applicaties en opgeslagen procedures om tussentijdse query's, nodig voor complexe processen, op te slaan. De gealloceerde ruimte die het dbms als temporary storage bestempelt, wordt tegelijkertijd gebruikt door vele -zo niet alle- databaseprocessen. Hiermee kan deze temporary storage een beperkende factor worden voor de applicatie/dbms-throughput.

Men probeert dit vaak op te lossen door de temporary storage op een gebufferd filesysteem te plaatsen. Gedachte hierachter is dat de schrijfoperaties in de temporary storage de snelheid van de cache van het filesysteem kunnen benutten; de daarop volgende leesoperaties kunnen gebruik maken van de data in de cache en hoeven niet fysiek naar disk.

Deze truc loopt fout als men te maken heeft met zeer drukke systemen waarbij de leesoperaties niet uit de cache kunnen plaatsvinden. Leesoperaties in een filesysteem worden vaak synchroon uitgevoerd, terwijl de meeste databasesystemen zich asynchroon toegang verschaffen tot de devices. Toevoegen van databasecache om het probleem van de tijdelijke 'hotspots' op te lossen, werkt alleen als de cache voor de temporary storage kan worden afgescheiden van de rest van de cache,

Voordeel is de snelheid van implementatie. Nadelen zijn de zoektocht naar de juiste 'hotspots' en de hoge aanschafprijs

bedoeld voor de databasebuffering. Nadeel hiervan is dat nu waardevolle cache aan de primaire database-objecten wordt onttrokken.

Tabellen en indexen

Tabellen en indexen zijn de gebieden binnen de database die een applicatie over het algemeen het meest benadert. Een van de eerste acties om een database te tunen is het plaatsen van deze hotspots op een aparte disk of zelfs op een aparte controller. Deze actie balanceert de I/O-load, maar nodigt uit tot inefficiënt diskgebruik. In zeer grote databasesystemen -bijvoorbeeld een decision support system (DSS)- kunnen deze tabellen en indexen echter zo groot worden, dat een nieuwe I/O-bottleneck ontstaat. Dit kan weer opgelost worden door striping toe te passen. Op sommige systemen heeft het toevoegen van extra geheugen een averechts effect op de algehele performance, omdat een interrupt nodig is voor het aanspreken van dit extra geheugen.

Voor de meeste databases geldt tegenwoordig dat de cache slechts een fractie van de grootte van de database bedraagt.

I/O-BANDBREEDTE

Als disk-I/O de flessenhals vormt om te komen tot een goede systeemperformance is vergroting van de bandbreedte de enige oplossing. Dit kan door het scheiden van de hotspots en/of gebruik te maken van filesystemen voorzien van cache en het invoeren van striping.

Een andere methode is gebruik te maken van apparatuur met een zeer snelle I/O, zoals de solid state disk. Hierop worden dan de hotspots geplaatst. Een SSD geeft de dba meer zeggenschap over de plaats van dit geheugen, dat daardoor in omvang vaak kleiner kan worden. Met veelal als gevolg dat de kosten per MB omlaag gaan. Maar zelfs bij gelijkblijvende kosten per MB komt het tot aanzienlijke besparingen.

Bovendien is de SSD 'mee te nemen' naar een volgend systeem -zoals dit ook in de mainframetijd werd gedaan- of daar in te zetten waar een probleem ontstaat.

Systeemgeheugen kan een organisatie zelden meenemen naar de volgende generatie, zelfs als zij de leverancier trouw blijft. Toevoeging van een SSD voor de transaction logs of als temporary storage is zeer snel uit te voeren met weinig of geen downtime en met een minimale interventie van dba en systems administrator.

ORACLE	ROLLBACK SEGMENTS	REDO LOGS	TEMPORARY SEGMENTS	
Sybase	Tempdb	Transaction logs	Vaak gebruikte tabellen	Indexen
Informix	RootDBS-tabel	PHYSDBS	LOGSDBS	TMPDBS1
Progress	AI	BI	Temporary tables	
SQL	Vaak gebruikte tabellen	Transactielogs	Temporary tables	Indexen
SAP	PSAPBTAB tables	PSAPCLU tables	PSAPSTAB tables	M tables

Tabel 1: Potentiële 'hotspots'.

TOEPASSING VAN SSD

In een op een 'flatfile-structuur' gebaseerde applicatie is het tamelijk eenvoudig de 'hot files' te identificeren. Plaatsing van deze componenten op een SSD zal tot onmiddellijk resultaat leiden. Inzet van een database maakt de zaak in één keer flink complexer. Nodig zijn gedegen kennis van de betreffende database en zijn lees- en schrijfgedrag voordat ook maar iets gedaan kan worden. Vaak biedt de applicatie zelf geen enkele ruimte tot verbetering vanwege allerlei interne 'locks' of omdat men moet wachten op de voltooiing van andere processen. Databases voeren veelal hun eigen slimme optimalisatie uit.

De organisatie moet zeker weten dat de performanceproblemen I/O-gerelateerd zijn

Oracle bijvoorbeeld buffert alle lees- en schrijfoperaties in de 'System Global Area'; gegevens worden alleen naar disk geschreven op het moment dat dit nodig is. Het besturingssysteem doet zijn eigen buffering. Dit betekent dat twee sets buffers worden gescand voordat de fysieke disk eraan te pas komt. Deze dubbelbufferstrategie werkt voor de meeste applicaties uitstekend.

Maar in een aantal applicaties worden deze buffers volledig overgeslagen. De parallelserver bijvoorbeeld slaat de 'operating systems buffers' over om de consistentie van de data op verschillende momenten te kunnen verzekeren. De grootste I/O-activiteit onder Oracle vindt men in de rollback-segmenten, de redo logs en de temporary segmenten. Van de meeste databases is bekend welke segmenten een hotspot kunnen veroorzaken, dit is echter ook afhankelijk van de applicatie. Tabel 1 geeft hiervan een indruk.

PRAKTIJKVOORBEELDEN

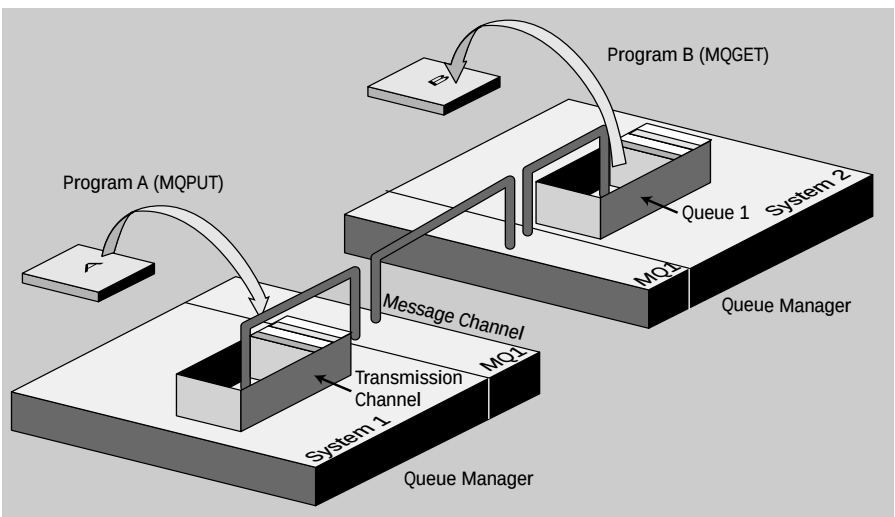
Een grote Nederlandse bank constateerde enkele jaren geleden dat een van zijn snelst groeiende applicaties dreigde vast te lopen vanwege de enorme groei. Deze onder Solaris draaiende applicatie, Asset Control genaamd, verzamelt de beursgegevens van onder andere Bloomberg, Reuters en Telekurs en controleert en verifieert de op zichzelf ongelijksoortige data, om die vervolgens op een uniforme manier aan te bieden aan een database (Sybase of Oracle).

Men vermoedde dat een I/O-flessenhals oorzaak was van de beperking in performance. De bank maakte reeds gebruik van de op dat moment snelste 'cache-centric' schijfeenheden, die gekenmerkt worden door relatief grote cache-buffers. Toch kon men op dat moment op bepaalde onderdelen niet meer dan 34 I/O's realiseren. Volgens de toenmalige groeiverwachtingen zou men binnen enkele jaren moeten doorgroeien naar enkele duizenden I/O's per seconde. Tuning van zowel applicatie als database boden onvoldoende soelaas.

Een aantal leveranciers werd uitgenodigd een oplossing te bieden. Dat resulteerde in uitgebreide benchmarktesten met twee leveranciers. Hierbij bleek dat zelfs een disk zonder cache in combinatie met RAM-logging de gewenste verbetering zou geven. Om het risico van RAM-logging te elimineren werd in de praktijk gekozen voor SSD. Uiteindelijk bleek 400 MB SSD voldoende om de hotspots van de applicatie, die gebruik maakte van enkele TB aan disk, te herbergen.

Inmiddels is het aantal I/O's per seconde gegroeid van 34 naar 300. Prijstechnisch was deze toepassing ook interessant.
In plaats van relatief dure cache-centric schijvensystemen kon de bank kiezen voor goedkope systemen zonder cache (minder dan de helft van de prijs!).

Lees verder op pagina 26.



Figuur 2: MQSeries: mainframedata migreert naar open systemen en omgekeerd.