Netezza Performance Server handelt query zeer snel af

Revolutionaire oplossing met lage TCO

Michiel Brunt en Frank Habers

De laatste jaren is duidelijk een trend ingezet dat steeds meer gegevens worden vastgelegd in datawarehouses, dat het aantal (externe) afnemers van informatie stijgt en bovendien meer (gedetailleerde) historie vereist is voor betrouwbare analyses. Ondanks de enorme hoeveelheid data verwachten alle gebruikers toch een 'google-respons' van hun datawarehouse. Om aan deze verwachting te voldoen biedt leverancier Netezza met de Netezza Performance Server (NPS) een nieuwe revolutionaire oplossing.

Een succesvol datawarehouse moet in ieder geval voldoen aan twee belangrijke randvoorwaarden: eenvoud en snelheid. Voor de voorwaarde 'eenvoud' zijn de afgelopen jaren allerlei tools, technieken en methodes ontwikkeld, maar de voorwaarde 'snelheid' blijkt nog steeds voor veel grote datawarehouses een struikelblok te zijn. Hiervoor is een aantal oorzaken aan te wijzen.
De eerste oorzaak is het feit dat steeds meer gegevens worden vastgelegd in het datawarehouse (als gevolg van bijvoorbeeld RFID, clickstream en e-business) en dat heeft logischerwijs nadelige gevolgen voor de query performance.
Voorheen was bijvoorbeeld alleen bekend welk product een klant kocht, met de beschikbaarheid van clickstream gegevens wordt ook duidelijk via welke weg (op de website) de klant het product heeft gekocht. De *grain*, het laagste niveau van transacties, wordt daarmee gedetailleerder. Het is een misvatting dat deze gegevens op voorhand kunnen worden geaggregeerd, daarmee gaat namelijk belangrijke informatie verloren (zie het kader over detaildata).

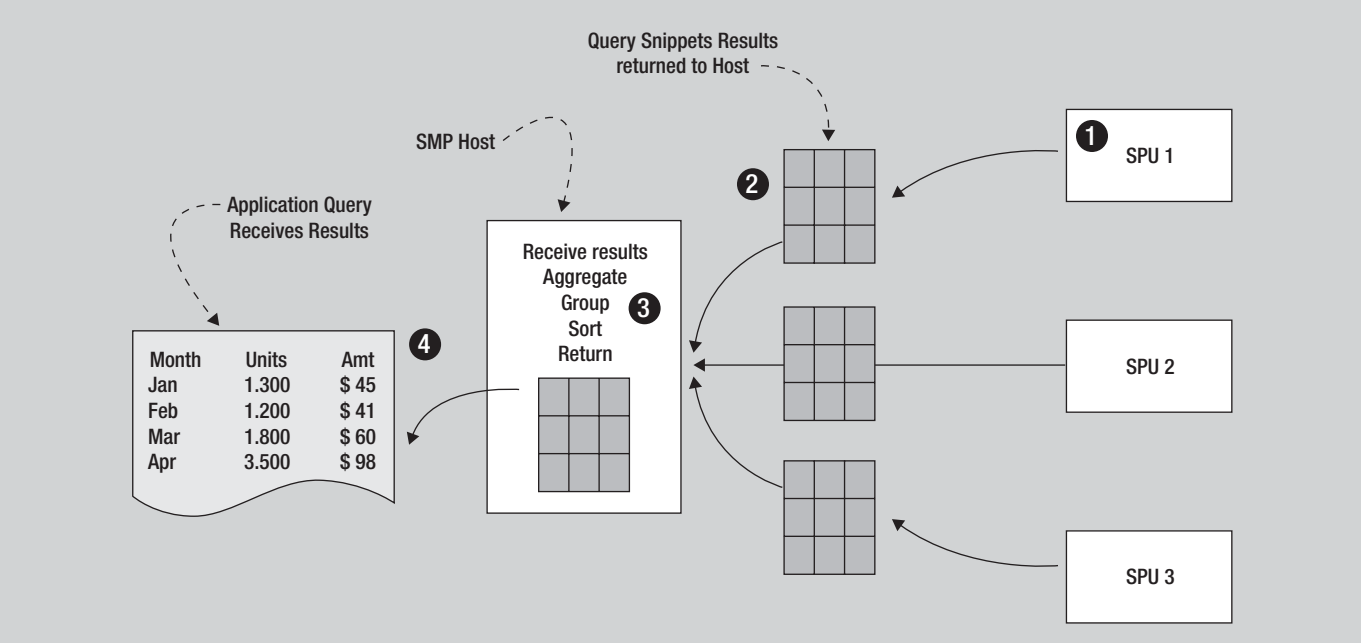
Historie

Een andere oorzaak van de groei van het datawarehouse is de behoefte aan steeds meer historische informatie. Enerzijds is er behoefte aan meer jaren aan historie van transacties en andere gebeurtenissen (bijvoorbeeld opgelegd vanuit wet- en regelgeving), anderzijds bestaat de wens om te beschikken over historie van relatief statische gegevens (dimensies). Voor een marketeer is bijvoorbeeld het oude adres van een verhuisde klant ook van belang en deze mag daarom niet worden verwijderd uit het datawarehouse.
De derde oorzaak dat performance een steeds grotere uitdaging wordt, is het steeds intensiever gebruik van informatie. Enerzijds zijn er meer interne gebruikers die frequenter het datawarehouse raadplegen, anderzijds krijgen ook steeds meer externe gebruikers (klanten, leveranciers) toegang tot het datawarehouse. Bovendien ontstaan er steeds meer (ook operationele) applicaties die gegevens opvragen uit het datawarehouse.
Door deze ontwikkelingen is het behalen van een goede query performance geen eenvoudige opgave. Daarbij speelt bovendien mee dat de performance-eisen van een gebruiker juist toenemen. Een goede performance is voor veel toepassingen het onderscheid tussen succes en falen. Leverancier Netezza heeft dit onderkend en heeft een revolutionaire nieuwe high performance-oplossing ontwikkeld. Om de kracht van deze oplossing duidelijk te maken, gaan we eerst in op de (bottleneck van) traditionele oplossingen.

Het traditionele probleem

Veel traditionele datawarehouse-oplossingen bestaan uit een RDBMS van de big three (Oracle, DB2, SQL Server), dat op een of meerdere servers is geïnstalleerd in combinatie met een verzameling van schijven. Zoals bij elke computertoepassing zijn ook bij de query performance drie hardware-componenten van belang:
- de schijven waarop gegevens worden opgeslagen;
- de processor (CPU) die bewerkingen en berekeningen uitvoert;
- het geheugen dat gegevens aan de processor aanlevert.

Elke component heeft een bepaalde rol bij het uitvoeren van een query. Op het moment dat het RDBMS een query ontvangt, zal het executieplan worden bepaald en wordt de opdracht gegeven om de vereiste blokken van gegevens van schijf af te halen en te laden in het geheugen. Vervolgens selecteert de CPU uit deze blokken (die uit meerdere rijen bestaan) alleen de rijen en kolommen die nodig zijn voor het eindresultaat. Ten slotte voert de CPU de berekeningen uit, worden de gegevens op basis van de geselecteerde gegevens in het geheugen geaggregeerd en wordt het eindresultaat gepresenteerd.
De bottleneck in dit proces is (bijna) altijd de relatief langzame schijf en het verplaatsen van de gegevens van schijf naar



Afbeelding 1: Stappen bij uitvoeren van een query.

geheugen, ofwel de I/O. Dit wordt veroorzaakt door de relatief lage toegangssnelheid van schijf (het zoeken van de juiste plaats om te gaan lezen) en de relatief beperkte bandbreedte van de I/O-bus naar het geheugen.

Het probleem bij traditionele oplossingen is dat juist bij deze bottleneck (de Input/Output) een traditionele database zeer grof te werk gaat. In plaats dat alleen de gevraagde rijen en kolommen van een query worden geladen in het geheugen, worden gehele blokken van gegevens geladen in het geheugen. Een blok bevat al snel 100 rijen (afhankelijk van de blokgrootte en de breedte van een rij) en daarmee worden ook veel irrelevante rijen geladen in het geheugen. Bovendien is veelal maar een beperkt aantal kolommen van een gehele rij relevant, dus worden ook overbodige kolommen (dus gegevens) geladen in het geheugen.

'The big three' hebben in de loop van de jaren allerlei extra functionaliteiten aan hun databases toegevoegd om de performance te verbeteren. Naast de gebruikelijke btree-indexen zijn er bijvoorbeeld bitmap-indexen die tot een betere performance leiden bij selecties op kolommen met een beperkt aantal verschillende waarden. Daarnaast is functionaliteit voor automatische aggregatietabellen ontwikkeld (materialized views in Oracle, summary tables in DB2, indexed views in SQL Server), waarbij gegevens bijvoorbeeld geaggregeerd per maand worden opgeslagen en de query optimizer van het RDBMS de query van de gebruiker herschrijft om, indien mogelijk, automatisch gebruik te maken van deze aggregatietabel (zonder dat de gebruiker of toepassing kennis hoeft te hebben van het bestaan van deze aggregatietabel). Verder zijn functionaliteiten als partitionering, parallel processing, caching en compressie toegevoegd. Echter, al deze functies lossen het probleem niet bij de wortel op, namelijk door structureel I/O te minimaliseren door enkel en alleen de vereiste data van schijf te selecteren. Een bijkomend nadeel is dat het inrichten en onderhouden van deze functies een verhoging van de beheerinspanning (en dus kosten) tot gevolg hebben.

Onder de motorkap van Netezza

Leverancier Netezza (opgericht in 2001) heeft met een frisse blik de kern van de performance-problematiek geanalyseerd en heeft een revolutionaire oplossing ontwikkeld: de Netezza Performance Server (NPS). Dit is een specifieke totaaloplossing voor high performance en is uitermate geschikt voor datawarehouses. NPS is niet geschikt (en gemaakt) voor OLTP-toepassingen. NPS is een combinatie van een server, een besturingssysteem, een RDBMS en schijven. De NPS bestaat uit de volgende twee componenten:

- een centrale unit, de host (SMP Linux), die query's ontvangt, distribueert en deelresultaten verzamelt en integreert;
- tientallen tot honderden Snippet Processing Units (SPU's), die onafhankelijk van elkaar de deel-query's uitvoeren en het resultaat opleveren aan de host.

Elke SPU bevat een standaard CPU, standaard geheugen, een standaard harde schijf voor de opslag van data en een (zeer belangrijke) *programmable hardware filter*. De SPU's zijn autonoom, ze voeren zelfstandig de query uit die door de host wordt gedistribueerd over de SPU's.

Bij het laden van data in Netezza zal iedere SPU een deel van de data ontvangen. De distributiewijze van data wordt per tabel ingesteld, vaak op basis van de primary key. Op deze manier krijgt iedere SPU evenveel data en zal dus ook bij het uitvoeren van een query iedere SPU bij benadering een evenredig aantal rijen moeten verwerken.

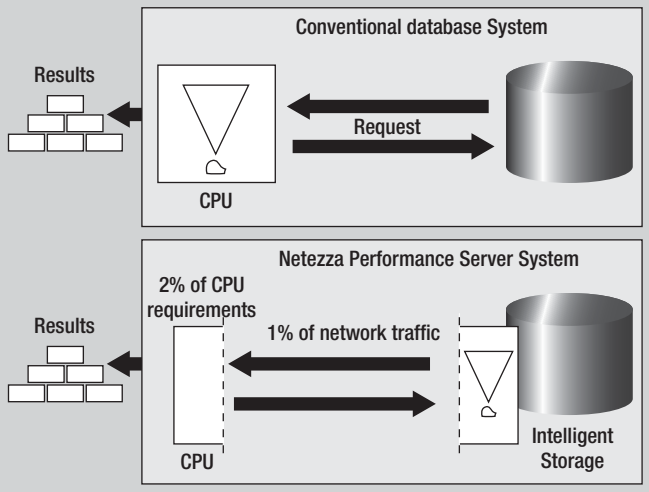
Afbeelding 1 toont de stappen bij de uitvoering van een query. Nadat de query is ontvangen door de Netezza host, krijgen alle SPU's de opdracht om de benodigde gegevens op te leveren (stap 1), ofwel dat relevante deel van de gegevens dat op de harde

De kracht van het programmeerbare hardware filter

Doordat het hardware filter (FPGA, Field Programmable Disk Array) programmeerbaar is, weet deze component op het moment dat de data gelezen worden van schijf of deze relevant zijn voor de query. Op die manier worden alleen de relevante kolommen en rijen doorgegeven aan het geheugen. Dit in tegenstelling tot een traditionele oplossing waarbij de relevante blokken(!) van gegevens worden geladen in het geheugen, waarna pas wordt bepaald welke rijen en kolommen binnen het blok relevant zijn.

Afbeelding 2 visualiseert dit verschil. Netezza noemt haar oplossing *Intelligent Query Streaming* en gebruikt hiervoor de zinsnede "bringing the query to the data".

De volgende vergelijking met de 'mensenwereld' maakt het verschil duidelijk tussen Netezza en een traditioneel RDBMS. In een voetbalstadion is het niet meer dan logisch dat bij de toegangspoort wordt gecontroleerd of een supporter een geldig ticket heeft en al dan niet het stadion mag betreden. Echter, bij een traditioneel RDBMS wordt per touringbus met voetbalsupporters (een blok van gegevens) bepaald of in ieder geval één supporter een geldig ticket heeft (een relevant record voor de query). Vervolgens worden alle supporters uit deze touringbus toegelaten tot het voetbalstadion, waarbij in het stadion dan wordt gecontroleerd welke supporters een geldig ticket hebben. De supporters zonder geldig ticket moeten alsnog het stadion verlaten. Het hoeft geen betoog dat deze werkwijze inefficiënt is. Het programmeerbare hardware filter voorkomt deze inefficiëntie.



Afbeelding 2: De traditionele oplossing versus de oplossing van Netezza.

schijf van de desbetreffende SPU staat. Iedere SPU (dit kunnen er honderden zijn) doorzoekt zijn schijf. Het programmeerbare hardware filter is zo ingesteld dat vanaf schijf (!) alleen de benodigde gegevens (rijen en kolommen) worden doorgelaten naar het geheugen van de SPU. Dit in tegenstelling tot een traditioneel RDBMS, waarbij een compleet blok wordt geladen in het geheugen (zie ook kader 'De kracht van het programmeerbare hardware filter').

Nadat de relevante gegevens zijn geselecteerd, voert de processor op de SPU de benodigde berekeningen uit en aggregeert de gegevens. De tussenresultaten van alle SPU's worden aangeboden aan de centrale host (stap 2). De CPU van de host voert alleen het afrondende werk uit, namelijk het integreren en aggregeren van de tussenresultaten van de SPU's en het uitvoeren van de totaalberekeningen (nummer 3). Gemiddeld vergt deze stap op de centrale host maar 2 procent van de totale query en is dus niet bepalend voor de query performance.

De fenomenale query performance van Netezza is dus te verklaren doordat enerzijds alleen de benodigde data van schijf wordt gehaald en dat anderzijds query's volledig parallel worden uitgevoerd op de SPU's (Netezza noemt dit Assymetrical Massive Parallel Processing). Bovendien versterken beide voordelen elkaar: wat moet gebeuren vindt efficiënt plaats, en wat efficiënt plaatsvindt, wordt ook nog eens sterk verdeeld uitgevoerd. Bovendien biedt NPS nog andere functionaliteiten om de performance verder te verhogen:

- Zone-maps. Dit is een technologie die automatisch de natuurlijke volgorde van data detecteert en daardoor bij benadering weet waar de gegevens op de schijf staan. Een (gesimplificeerd) voorbeeld hiervan is dat orders, die dagelijks worden bijgeladen in het datawarehouse, op volgorde van orderdatum te vinden zijn op schijf;
- De SPU's kunnen onderling gegevens uitwisselen. Bijvoorbeeld een kleine join-tabel wordt dynamisch in zijn geheel over het interne Gigabit Ethernet-netwerk naar alle SPU's verstuurd;
- De prioriteiten van query's kunnen worden bepaald op basis van verschillende criteria. Een query van een seconde kan op basis van het criterium doorlooptijd voorrang krijgen op een query die vijf seconden duurt. Ook kan voor bepaalde groepen gebruikers een hogere prioriteit worden ingesteld.

Performance-cijfers

NPS is vanuit technologisch oogpunt buitengewoon interessant, maar de belangrijkste vraag is vanzelfsprekend: hoe snel is NPS? Om dit te bepalen hebben we een performance-test uitgevoerd op een NPS met 26 SPU's. De testcase zag er als volgt uit:

- Een sterschema met een feitentabel met 400 miljoen rijen en 10 dimensietabellen, waarvan de grootste dimensietabel 1,4 miljoen rijen bevat;
- Er zijn zes verschillende query's uitgevoerd (traditionele datawarehouse query's: select.. from.. where.. group by);
- Per query zijn gemiddeld zes dimensietabellen gekoppeld aan de feitentabel;

- Er is geen specifieke tuning uitgevoerd, de database en tabellen zijn aangemaakt en gevuld. Er zijn dus geen indexen aangemaakt (dit is helemaal niet mogelijk).

Het resultaat: Alle query's leverden binnen 1,2 tot 2,5 seconden het resultaat op. Dit maakt duidelijk dat de performance van NPS fenomenaal is. Indien deze performance niet volstaat kan NPS uitgebreid worden door het aantal SPU's te vergroten. De performance van NPS is lineair schaalbaar, een verdubbeling van het aantal SPU's leidt tot een halvering van de doorlooptijd van een query.
De laadsnelheid van gegevens naar de database is ook hoog. Ter indicatie, op het de NPS met 26 SPU's is een laadtest uitgevoerd, waarbij een tabel van 2 miljoen records allereerst is gekoppeld aan een tabel van 20 miljoen records en vervolgens 20 miljoen records (ruim 2 Gigabyte aan data) zijn geladen naar de database. Dit proces duurde 100 seconden, dat is eveneens een goede prestatie.

Minimale beheerinspanning

Netezza onderscheidt zich ook van traditionele databases doordat de beheerinspanning minimaal is. Iedere DBA en datawarehouse-ontwerper weet dat performance-optimalisatie traditioneel veel inspanning vergt (en blijft vergen). Dit wordt veroorzaakt door activiteiten als het maken van aggregatietabellen, bepalen en instellen van indexen, beheren van datafiles, inrichting van table spaces, instellen van logfiles, instellen van database-parameters (en het meten van de effecten), inrichten van replicatie, keuzes rondom RAID- instellingen, instellingen op OS-kernel niveau, etcetera.

Waarom detaildata vereist?

Een rapport is vaak een aggregatie van gegevens in het datawarehouse (bijvoorbeeld bij een maandrapportage worden alle orderregels geaggregeerd naar maanden). De veel gemaakte fout is dat daaruit de conclusie kan worden getrokken dat de gegevens dus geaggregeerd opgeslagen kunnen worden in het datawarehouse. Vooraf is namelijk niet vast te stellen via welk pad (dus via welke dimensies) wel en niet (!) geaggregeerd gaat worden. Het vastleggen van bijvoorbeeld individuele orderregels lijkt niet relevant, omdat individuele orderregels niet worden geanalyseerd. Echter, een geaggregeerde vraag als "wat is de omzet van artikel X, dat op de laatste zaterdagochtend van de maand door klanten wordt gekocht die gemiddeld meer dan 100 euro per maand besteden", leidt ertoe dat gegevens op artikelniveau, klantniveau en datumniveau moeten worden vastgelegd. Om deze (dimensie)gegevens te kunnen koppelen aan omzet, is het vereist dat de omzetgegevens op het laagste niveau (orderregel) worden vastgelegd. Details zijn dus vereist in het datawarehouse en elke aggregatie leidt tot informatieverlies. Tevens zijn details noodzakelijk voor het verklaren van geaggregeerde cijfers. Bovendien is het laagste detailniveau een randvoorwaarde voor een uitbreidbaar (gegevensmodel van het) datawarehouse.

Bij NPS hoeven dit soort acties en afwegingen niet gemaakt te worden. Het is een kwestie van NPS aansluiten, de database aanmaken en het laden van de data (Netezza noemt dit 'load and go implementation'). De enige beheerinspanning zit in het eenmalig inrichten van de backup & recovery en het eenmalig bepalen van de distributie van de data over de SPU's. De NPS beschikt bovendien over automatische herstelfunctionaliteit. Mocht een SPU defect raken, dan wordt automatisch een reserve SPU geactiveerd. De data van de defecte SPU zijn nog beschikbaar op een van de andere SPU's. Deze data worden gekopieerd naar de geactiveerde reserve SPU. De gebruiker merkt niets van deze wisseling. Het vervangen van een SPU is niets anders dan de defecte SPU eruit halen en de nieuwe SPU plaatsen. De database blijft tussentijds gewoon beschikbaar.

Vanzelfsprekend rijst de vraag of er ook nog nadelen kleven aan NPS. We hebben maar één mogelijk nadeel kunnen constateren, namelijk een probleem op het gebied van ETL (Extractie, Transformatie en Laden). NPS laadt gegevens zeer snel zolang het inserts betreft, updates zijn binnen NPS traag. Netezza adviseert dan ook om updates te verwerken via (zeer snelle) deletes en inserts. Dat betekent wel dat het gebruikte ETL-tool daarvoor faciliteiten moet bieden. Gelukkig zijn leveranciers van ETL-tools druk bezig om optimaal aan te sluiten op de NPS. Netezza heeft inmiddels partnerships met onder andere Business Objects, Informatica en Sunopsis afgesloten.

Prijsstelling

Dat brengt ons op de prijsstelling van deze oplossing. NPS maakt gebruik van standaard hardwarecomponenten (CPU, geheugen, schijven) die terug te vinden zijn in een gemiddelde (kantoor)server. Door gebruik te maken van deze standaard-componenten kan Netezza de NPS zeer concurrerend prijzen. De prijs is daarbij logischerwijs afhankelijk van het aantal SPU's. Dat loopt uiteen van 28 SPU's met 1,5 tot 3 TB aan schijfruimte tot en met een model met 896 processoren en 100 TB aan schijfruimte. In combinatie met de minimale beheerinspanning leidt NPS tot een zeer lage TCO.

Conclusie

Netezza biedt met NPS een revolutionaire oplossing met onderscheidend vermogen. Doordat de oplossing van Netezza een combinatie van hardware, RDBMS en dataopslag is, is de verwachting niet dat de 'big three' de functionaliteit van Netezza kunnen opnemen in hun RDBMS. Netezza heeft dus een unieke propositie ten opzichte van deze, en overige, leveranciers en dat zal de kansen van Netezza vergroten om in de komende jaren een toonaangevende speler op het gebied van high performance solutions te worden.

Michiel Brunt en **Frank Habers** zijn respectievelijk architect en directeur bij Inergy Analytical Solutions.