Martin Kersten ziet nog genoeg uitdagingen voor database-leveranciers

"Het is droevig gesteld met zelfbeheer"

Teus Molenaar

De huidige commerciële database-leveranciers halen alles uit de kast om de prestaties van hun producten te verbeteren, maar verder dan suboptimalisatie zullen ze nooit komen, zo meent prof. dr. Martin Kersten. Twee steekwoorden: zelfbeheer en indexloze database.

Een gesprek met Kersten, hoofd van de afdeling Information Systems van het CWI, over de stand van zaken in 'database-land' mondt al snel uit in een hoorcollege. Enthousiast vertelt hij over de ontwikkelingen, zoekend naar structuur, want er gebeurt zoveel tegelijkertijd. Op het white board schetst hij structuren van databases, de problemen waar ze tegenaan lopen, mogelijke oplossingen, en gaat hij in op die geheel andere aanpak van de 'eigen' open source database MonetDB (zie kader).
Hij pakt meteen dit onderwerp aan; open source databases. "Het is duidelijk dat open source niet meer is weg te denken. Dit wordt gestaafd door de acquisities die plaatsvinden op dit vlak. Denk aan Oracle die BerkeleyDB, met de overname van Sleepycat, heeft binnengehaald. Het gaat dan wel om een embedded database, maar ze halen er wel technologie mee binnen. En MySQL ligt natuurlijk altijd al onder vuur. Oracle heeft eerder ook gepoogd dit bedrijf over te nemen. MySQL heeft toen terecht nee gezegd, want het is een heel ander business-model."
De technische kwaliteit van de open source databases kwalificeert Kersten als 'over het algemeen goed; zeker competitief met de mainstream-oplossingen'. Wel ziet hij de schaalbaarheid als een punt van zorg bij de open source databases.

Zelfbeheer

Een tweede punt van zorg noemt de professor het gebrek aan het zelfbeheer van de database-systemen. "Maar dat geldt evenzeer voor de commerciële varianten", onderstreept Kersten. "Het is gewoonweg onmogelijk om een grote database te bouwen als je maar één dag SQL-kennis hebt opgedaan. De out-of-the-box ervaring is bedroevend.
Self management is eigenlijk het bepalen van de indices, de ondersteunende toegangsniveaus, binnen een database. De indices worden vooral bepaald door de workload, dus daar moet je inzicht in hebben. Vroeger keek de DBA daar naar, koppelde dat aan het belang van een query set en kon in al zijn wijsheid beslissen of het nuttig was om bijvoorbeeld een extra index te maken voor een zekere query set om het geheel sneller te laten verlopen. Dát stuk van het beheer is wel redelijk door de software zelf overgenomen. Maar dat staat uiteindelijk nog heel ver af van wat er in de praktijk moet gebeuren. "Dit onderwerp blijft de komende jaren nog een van de grootste uitdagingen."
Hij komt met een simpel voorbeeld. "Vorig jaar was ik zes weken bij Microsoft Research op bezoek om goed bekend te geraken met SQL Server 2007. De eerste experimenten resulteerden in een desastreuze performance. Ik had even vergeten dat, als je een database opbouwt met behulp van alleen SQL update statements, er natuurlijk een log wordt aangemaakt van die transactie. Die log groeide uit de klauwen, binnen een uur was de harde schijf vol. Die log wordt aangemaakt om eventueel te kunnen terugkeren naar een oorspronkelijke situatie of bij een crash de database te herstellen. In mijn geval was die oorspronkelijke situatie een lege database, dus hoefte er helemaal geen log te worden aangemaakt. Het was een fout aan mijn kant, want ik had de log moeten uitzetten. Het was een fout aan de database-kant, want die had moeten nagaan wat hij écht had moeten doen om terug te gaan naar de initiële toestand of voor crash-herstel, namelijk het bewaren van de 1000 SQL statements in plaats van de 'before/after image record' in een snel groeiende table van een paar miljoen records."
Een ander voorbeeld: een experiment met TPC-H benchmark om de schaalbaarheid te meten van MonetDB, PostgreSQL en MySQL. Het bleek dat MonetDB tien minuten nodig had om de database te laden – veel te lang. Maar het bleek dat PostgreSQL er driehonderd minuten over deed, en MySQL zat er ergens tussenin. "Na een load moet je in PostgreSQL een zogenoemde statistics call doen die de database helemaal doorgaat om een goed query-plan te kunnen maken. Dat was de medewerker vergeten te doen. Dat hebben we opgelost en vervolgens zijn we de query's

Foto: Harry Otto.

Prof. dr. Martin Kersten: "Duidelijk is dat open source niet meer is weg te denken."

gaan draaien. Ook na optimalisatie bleek dat er een gigantisch verschil is in performance bij drie verschillende open source database-systemen. MySQL en PostgreSQL konden bepaalde query's niet draaien, omdat ze langer dan een uur zouden duren. En dan heb je het over een database van 1 GB, dat past tegenwoordig gewoon in het geheugen. Die grote variëteit in performance zul je overigens ook bij de commerciële systemen aantreffen."
De conclusie luidt: zonder diepgaande kennis van SQL en database-technologie is het niet mogelijk een database behoorlijk te laten presteren. Kersten meldt dat de database-leveranciers wel druk doende zijn dit probleem aan te pakken. Er wordt onder andere een nieuwe benchmark ontwikkeld (TPC-DS) voor de fijnregeling van de kernels. "Maar onderschat niet hoeveel werk het kost om de kernel aan te pakken. Voor SQL Server zit al een paar honderd man alleen aan de kern te werken. Wij doen het trouwens met negen mensen."

Kolommen

Een korte termijn probleem is het oplossen van de achterhaalde manier waarop bestaande databases met geheugen en CPU omgaan. De nummer 1 in de TPC-H 100 GB benchmark-lijst gebruikt een database-constellatie met vier multi-core CPU's, 128 GB RAM-geheugen en 100 schijven van 36 GB. "Daar draaien we een database op van 100 GB. Zoals je ziet, zijn de memory databases ongemerkt al onder ons", grapt Kersten.

De kern van het probleem is dat weliswaar de hardware zoveel sneller is geworden, maar dat de database software als gevolg van zijn structuur niet optimaal gebruik kan maken van die snelheidswinst. Ze maken heel slecht gebruik van hun memory-hiërarchie. Uit testen bleek dat ze maar tien procent van de CPU gebruik maken. Dat betekent dat de CPU stond te wachten op het geheugen om rekentaken toegespeeld te krijgen.
"Dat komt door de structuur in rijen. De database doorzoekt alle rijen om een gebiedje van 64 tot 128 bit naar CPU-cache te schrijven. Maar van de 64 bit, zijn er vaak maar 4 nodig. Er is dus iedere keer onnodig verkeer van 60 bit. Er gebeurt dus wel heel veel, en misschien ook wel snel, maar het uiteindelijke effect is dat toch maar heel weinig nuttige informatie in de CPU terecht komt. Er is nog wel wat winst te halen met *cache finetuning*, maar dat zal niet voldoende blijken. Ze kunnen niet zomaar de fysieke database-structuur helemaal omgooien."
De inherente traagheid van relationele databases is op te lossen door de database niet in rijen, maar in kolommen in te delen. Deze aanpak staat volgens de professor op doorbreken in de open source wereld en bij de niche-spelers. Mike Stonebreaker, uitvinder van de Ingres en PostgreSQL database, heeft de omslag gemaakt en heeft twee jaar geleden met zijn C-store de kolom-gebaseerde aanpak van Kersten c.s. omarmd. "Schijfruimte is goedkoop", vertelt Kersten. "Dus je kunt een database gewoon repliceren en kolommen op verschillende manieren sorteren. Als

Foto: Harry Otto.

"Je kunt wel voorspellen wat er gebeurt met acht CPU's: die staan met z'n achten te wachten."

je goed bijhoudt hoe die sortering is, dan kun je query's heel snel afhandelen, want je hoeft niet meer te zoeken."
De grote database-spelers kunnen niet van de ene op de andere dag de kolomgebaseerde aanpak overnemen, omdat ze met een enorme legacy zitten. "Ze moeten een migratiepad ontwikkelen. Daarom zijn ze zo geïnteresseerd in de open source databases."

Compressies

De grote vraag is in hoeverre multi-core CPU's iets te betekenen hebben voor databases. "Als je nagaat dat er met één CPU al een probleem is, dan kun je wel voorspellen wat er gebeurt met acht CPU's: die staan dan met z'n achten te wachten."

Een database heeft voor verwerking te maken met disk, geheugen, en CPU met cache. Het transport tussen die onderdelen dient te worden versneld. "Nu weten we dat sommige waarden in een database heel vaak voorkomen en sommige nauwelijks. Met dat gegeven kun je een tabel al gaan comprimeren. Sommige leveranciers doen dat ook. Die comprimeren de database files op disk. Bij een query wordt het stukje dan gecomprimeerd uit het geheugen gehaald, vervolgens gaat het naar de CPU die het expandeert, en deze zet het terug in de page pool waarna het klaar staat voor gebruik. Het is veel handiger om die kleine stukjes in de cache te houden en daar te verwerken. Dit heet *light weight compressie*. Je bent niet geïnteresseerd in maximale indikking van alle data, maar alleen van die stukjes die je voor query's nodig hebt. Met dit soort grapjes kun je de bandbreedte tussen disk en CPU verdrievoudigen. Dit kun je ook in de traditionele database doen."

Vlinderdas

De uiteenzetting over streaming data en databases levert een mooie tekening op in de vorm van een vlinderdas. Links de RFID-tags met hun gegevens die worden verzameld en vervoerd naar een datawarehouse voor verdere verwerking. Het rechterdeel van de vlinderstrik staat voor applicaties die gebruik maken van bestanden in het datawarehouse. Een ander model is dat waarbij een reeks sensoren (bijvoorbeeld temperatuurmeters) via WiFi data naar het datawarehouse stuurt.
"Draadloze communicatie is vrij duur, dus is het de kunst om de datastromen zo klein mogelijk te houden. Het is mogelijk om terwijl de data voorbij vliegen – bijvoorbeeld bij een ticker-tape van aandelenkoersen – een query uit te voeren om na te gaan of bij-

Cracking query's en databases

Om query's snel te kunnen afhandelen, is het nodig om indexen te bouwen. Maar niet op elke kolom, want dat wordt erg duur in het onderhoud. De grote vraag waar elke database-beheerder zich voor gesteld ziet, is: welke index?
"Wij lossen zijn probleem op met database cracking. We zeggen gewoon: vergeet de indices. Bouw ze gewoon niet meer. Je hebt een tabel, en dat is niet meer dan een stapel. Elke nieuwe tuple zet je er gewoon bij. Nu komt de eerste query; een gebruiker heeft een vraag. Maar ik heb geen index. Het enige wat ik kan doen, is door de hele stapel heen lopen en eruit halen wat past bij de vraag. Stel, we hebben eigenschap A en ik ben geïnteresseerd in alles waarbij A<8 en A>5. Ik loop door de database heen, maar terwijl ik dat doe, ga ik hem fysiek anders ordenen op zo'n manier dat alles waarbij A een waarde heeft tussen de 8 en de 5 bij elkaar komt te staan. Die query onthoud ik. Dat is een dure operatie. En dat doe ik steeds weer bij elke volgende query. Wat we in feite doen, is een index – een soort navigatiepad – opbouwen naar de query-antwoorden die in de praktijk het meest voorkomen. Vooral bij enorm grote databases komt het voor dat de gebruikers meestal alleen belangstelling hebben voor een klein gebiedje. Dit is een stuk effectiever dan steeds je indices bijhouden."
Het tweede is query cracking. "In de praktijk wordt vaak een richtlijn uitgevaardigd dat een gebruiker alleen een query mag uitvoeren die niet langer dan bijvoorbeeld tien minuten duurt. Maar hoe weet een gebruiker hoe lang een query gaat duren? Dat kan hij niet weten, en de database weet het ook niet. Misschien kunnen we dat oplossen door een query op te splitsen in twee query's. Je voert de eerste query uit en daar krijg je een volledig antwoord op. De tweede query kan door de gebruiker op een later tijdstip worden benut om ontbrekende antwoorden op te halen. Het voordeel is dat als een gebruiker een fout heeft gemaakt, hij dat meteen al merkt bij het eerste antwoord. Die tweede query wordt dan niet uitgevoerd; dat scheelt veel werk. De uitdaging is om dure query's zodanig op te delen dat je deze vorm van 'ondervraging' kan uitvoeren."

Onder de projectnaam Pinwheel heeft het CWI onlangs subsidie gekregen van NWO om deze twee 'cracking-uitdagingen' aan te gaan.

voorbeeld IBM of Philips *out of sync* zijn, want dan kun je er misschien iets leuks mee doen.
Streaming is dus sterk gekoppeld aan embedded database-technologie. Dat is geen probleem, want de meeste leveranciers hebben wel een embedded versie. Probleem is wel dat de query's nooit eindig zijn. Je kunt niet een query uitvoeren over een afgebakend stuk database, want met streaming komen er steeds nieuwe gegevens bij. Dan krijg je een structuur met een open ended query, waarbij gewerkt wordt met voorlopige maxima en minima. Als er een vraag komt, dan wordt gerekend met die tijdelijke grenzen, totdat er een nieuw maximum komt. Dan wordt die waarde automatisch aangepast. Afgelopen jaren is er in de research-wereld veel aan streaming gedaan. De vraag is of het echt doorbreekt. Wat je hier doet, is het sturen van boodschappen van de ene cycle naar een andere; van de ene database naar een andere. Je verstuurt boodschappen. Met Message streams van Oracle bijvoorbeeld. Als het aantal boodschappen beperkt is, dan is het voldoende. De vraag is of het in het bedrijfsleven gebruikt gaat worden, want er zijn daar niet zoveel hoogfrequente streaming-applicaties." Lange termijn onderzoek ligt voor Kersten op het gebied van database en query cracking (zie kader). Daar is zijn groep mee bezig. Maar ook op andere universiteiten staat dit onderwerp op het lijstje.

Teus Molenaar is freelance journalist.

MonetDB

Onder aanvoering van Martin Kersten heeft het database team van het Centrum voor Wiskunde en Informatica (CWI) een database ontwikkeld die is gebaseerd op de technische mogelijkheden, en de gebruiksbehoefte, van de jaren negentig, terwijl de 'grote' database-leveranciers (IBM, Oracle, Sybase, Microsoft) hun producten hebben ontwikkeld in de jaren tachtig. Het grote verschil is dat de 'originele' databases alle tuples opslaat in rijen, terwijl MonetDB ze in kolommen opslaat, hetgeen enorme tijdwinst oplevert bij query's.
MonetDB, zoals de database heet, is een bijproduct van tien jaar onderzoek op de Amsterdamse universiteit naar database-architecturen. De database is vooral gericht op query-dominante toepassingen en grote databases, waarbij het gaat om tabellen met honderden kolommen en miljoenen records. MonetDB is al gebruikt voor een scala van toepassingen, zoals datamining, OLAP, geografische informatiesystemen, XML en multimedia retrieval. Het biedt de gebruiker een volwaardig alternatief voor bijvoorbeeld MySQL en is leidend in de ondersteuning van XML-databases. Het product kent onder andere automatisch onderhouden indices, query-optimalisatie gedurende de verwerking, en een modulaire software-architectuur. Omdat MonetDB is ontwikkeld met gemeenschapsgeld is het verkrijgbaar onder het Mozilla licentiemodel (open source).

Update

Bloor Research: Progress Apama marktleider in Event Processing

Progress Apama heeft volgens Bloor een aantal sterke voordelen boven concurrerende oplossingen: de volledigheid van het product, het prestatieniveau en de omvang en variatie van de bestaande klantenkring. Het Apama-platform van Progress is het eerste 'complex event processing'-systeem dat de gebruiker in staat stelt om vluchtige gegevensstromen te monitoren, analyseren en erop te reageren – en dat alles binnen enkele milliseconden, aldus het rapport. Philip Howard, Research Director bij Bloor en de auteur van het rapport schrijft: "Het Apama-product van Progress heeft verschillende grote voordelen ten opzichte van de meeste van zijn concurrenten: het aantal jaren dat het al bestaat, het grote klantenbestand en het feit dat het een wereldwijde speler is zijn slechts enkele voorbeelden. Geen enkele andere leverancier kan al deze kwaliteiten met het prestatieniveau en de brede inzetbaarheid van het Apama-product aanbieden. Met de integratie van de technologieën van Progress en Apama is er nu een werkelijk compleet event processing-platform op de markt gekomen." Progress Apama is het enige alomvattende event processingplatform op de markt. Naast een event processing engine, beschikt dit platform over een ontwikkelomgeving waarmee zowel zakelijke gebruikers als IT-afdelingen over een dashboard beschikken voor het monitoren van real-time gegevensstromen, en een 'event data store' voor het vastleggen, opnieuw afspelen en analyseren van gegevensstromen. De geavanceerde event processing-functionaliteit van het Apama-platform kan op een breed scala van zeer snelle gegevensstromen toegepast worden, waaronder financiële markt-gegevens, RFID-signalen, gegevensverkeer van telecommunicatie-netwerken en telemetrische gegevens van satellieten. Producten binnen het Progress Apama aanbod zijn onder meer Apama Algorithmic Trading Platform en Apama ESP Platform.
De architectuur en verschillende elementen van Progress Apama worden in het rapport gedetailleerd beschreven en beoordeeld. Het Apama-platform-rapport kan van www.progress.com/realtime/docs/analyst/bloor_apamareview_061910.pdf worden gedownload.

Rectificatie

In het artikel 'Hogere orde concepten en hun visualisatie' van Maurice Gittens in DB/M 7-2006 is helaas tijdens het redactieproces een onjuistheid geslopen. Onze lezer dr. A.J.A. van Griethuysen maakte ons erop attent dat er enkele keren *intentie* (= bedoeling) vermeld staat, waar dit de dimensie *intensie* (= begripsinhoud) moet zijn, in semantische samenhang met de dimensie *extensie*. De redactie dankt hem voor zijn signalering; onze verontschuldigingen voor deze begripsverwarring.