Keuzen maken voor de hardware - performance-analyses en -tests

Goede performance is geen Utopia (3)

Toon Loonen

Ook van een overbelast LAN of een langzaam modem zal de gebruiker de gevolgen beslist merken...

In de opzet van zijn systeem krijgt de dba, naast de applicatie-architect, te maken met een groot aantal verschillende aspecten. Houdt hij met deze punten rekening, dan ligt een goede performance binnen bereik. Dit afsluitende deel van een drieluik onderzoekt de mogelijkheden daarvoor in de hardware. Verder wordt gekeken naar het belang van het analyseren en testen van de performance.

Naast alle maatregelen die men moet nemen om het systeem zo efficiënt mogelijk te schrijven -in vele aspecten, zoals performance, maar ook zaken als onderhoudbaarheid en testbaarheid- zijn er nog de eisen die aan de hardware gesteld mogen worden.
Monitor geregeld de computer waarop de databaseserver draait.

- Probeer bottlenecks in het gebruik van cpu, memory, I/O of netwerkbelasting op te lossen voordat ze een probleem geworden zijn.
- Let hierbij ook op de verdeling van de I/O over de verschillende schijven; mogelijk is één schijf zeer zwaar belast en de andere schijven nauwelijks.
- Bij een computer met meer cpu's zij gelet op de verdeling van de capaciteit over deze processoren. Zou de databaseserver maar 1 cpu gebruiken en draait er verder niet veel op deze machine, dan zijn mogelijk de drie overige cpu's werkeloos. Richtlijn voor cpu-belasting: deze mag niet hoger zijn dan 75% op de drukke tijd van een drukke dag.
- Let hierbij ook op wat overigens aan toepassingen op dezelfde server draait. Zijn daarbij toepassingen die (overdag) incidenteel een zeer hoge belasting geven, dan zullen de gebruikers van de andere systemen op dat moment een slechte performance ervaren. Hetzelfde kan gebeuren wanneer -incidenteel of permanent- het netwerk zeer zwaar wordt belast.

Naast de (database)server kan het netwerk de performance beïnvloeden. Een overbelast LAN of een communicatielijn via een langzaam modem of via Internet tussen client en server zal beslist nadelig zijn voor de door de gebruiker ervaren performance van het systeem.

MEMORY TOEKENNEN AAN SERVER

Veel winst valt ook te behalen door de computer waarop de databaseserver staat, uit te rusten met veel main memory en dit ook

Basistips voor performance (3)

Er is waarschijnlijk geen onderwerp waarover door dba's zoveel wordt gepraat en niettemin zo weinig is gepubliceerd als over performance in een (r)dbms. In een serie van drie artikelen beschrijft Toon Loonen de belangrijkste punten waarmee men in het database-ontwerp rekening moet houden om tenminste een basis voor een goede performance te leggen. Hij beperkt zich tot de 'universele' *vendor*-onafhankelijke aspecten en tips.
In deel 1, in DB/M 1, stonden de mogelijkheden binnen het rdbms centraal en in het vorig nummer werd aandacht geschonken aan de rol van de applicatie. De reeks sluit nu af door onder meer in te zoomen op de hardware. Vervolgens vat de auteur een aantal conclusies samen en doet hij enkele aanbevelingen.

aan de databaseserver toe te wijzen. Dit stelt de server in staat veel gebruikte gegevens in memory te houden, zodat hiervoor geen fysieke I/O meer nodig is.

Bekijk altijd de hoeveelheid geheugen van de computer en hoeveel hiervan werkelijk gebruikt wordt. Als er alleen een databaseserver op draait, wijs dan al het niet voor het besturingssysteem benodigde geheugen toe aan de databaseserver.

Is fysieke I/O een flessenhals in het systeem, bekijk dan of het plaatsen van extra memory, en het vervolgens toewijzen daarvan aan de databaseserver, mogelijk is om dit knelpunt op te lossen.

Tabellen in memory

Er zijn soms -per product verschillende- mogelijkheden om kleine, veel gebruikte tabellen of (het hoogste niveau van) indexen 'vast' in memory te plaatsen. Dit geeft waarschijnlijk weinig extra performanceverbetering, omdat het dbms toch al de meest gebruikte pagina's in memory houdt; de pages van deze tabellen waarschijnlijk ook.

In-memory databases

Het verst gaan in dit opzicht de *in-memory databases* [zie ook 9 en 10]. Hierbij zitten alle gegevens permanent in main memory. Dit voorkomt niet alleen de I/O, maar ook de overhead in de server die kijkt of de gegevens al in memory staan. Enkele leveranciers van deze producten claimen een performance die tienmaal beter is dan een traditioneel rdbms, waarbij de gegevens toch al geheel in memory geladen zijn. Dit soort rdbms is het meest interessant voor niet te grote databases die wel zeer intensief gebruikt worden.

Naast de geheugeninstellingen kennen de dbms-producten vele andere parameters die de performance kunnen beïnvloeden, hetzij positief of negatief. Deze parameters zijn zo leverancierafhankelijk, dat ze buiten het bestek van dit artikel vallen. Een cursus of handboek 'systeemadministratie' of 'performance en tuning' van de betrokken vendor en enige ervaring met het product zal de dba in staat stellen het systeem verder te tunen.

PERFORMANCE ANALYSEREN EN TESTEN

Ondanks al het, veelal preventieve, werk dat hierboven beschreven is, zullen zich hier en daar performanceproblemen voordoen. De programmeur zal deze vaak niet constateren tijdens zijn unit/programma-test, omdat hij meestal met heel weinig gegevens in de database test.

Specifieke performancetesten op een database van productiegrootte zijn nodig

Om performanceproblemen in de productie-omgeving uit te sluiten, zullen dan ook specifieke performancetesten op een database met een inhoud op productiegrootte nodig zijn. De tester die hiervoor de productiedatabase kan en mag gebruiken, bevindt zich in ideale omstandigheden. Soms mag het niet (onder meer wegens privacy-restricties), maar kan wel een kopie van de productiedatabase worden gebruikt waarin de meest gevoelige gegevens (medewerkers, klanten) anoniem gemaakt zijn door naam en adres te wijzigen in bijvoorbeeld "naam_<id>", "straat_<id>".

Misschien valt aan de stap die het meeste tijd kost niets te verbeteren, terwijl vooruitgang op de andere punten het systeem toch sneller maakt

Voor een nieuw systeem is geen productiedatabase aanwezig. Maar als ook een conversie vanuit een bestaand systeem gebouwd moet worden, zal deze, getest op een kopie van de productiedatabase, ook een goede omgeving opleveren voor het uitvoeren van een performancetest.

Valt ook deze optie af, dan zal men een testdatabase moeten opbouwen door tabellen te vullen met statements, zoals:

```
teller = 0
WHILE teller < 10000
begin
  teller = teller + 1
  telchar = <characterversie van de teller>
  INSERT into klant (nummer, naam, straat,
      huisnr, etc.)
    values (teller, "naam"+telchar,
      "straat"+telchar, teller modulo 100, etc.)
  INSERT into order
    values. . . .
end
```

Het is lastig hierbij een testdatabase op te zetten met een verdeling van de gegevens die enigszins overeenkomt met de werkelijkheid - als die al bekend is op dat moment. Verder zal de database zo gevuld moeten, dat aan de validatie-eisen wordt voldaan; dit om andere fouten in de programmatuur te voorkomen. Erg veel werk al met al, maar onvermijdelijk voor een groot systeem met hoge performance-eisen.

PERFORMANCE-ANALYSES

Als tijdens testen op bovenstaande database, of in productie, toch nog performanceproblemen geconstateerd worden, zullen deze geanalyseerd moeten worden. Tijdens het herhalen van de test zijn een aantal zaken te meten.

Als een programma of een stored procedure een groot aantal stappen bevat, bouw dan *displays* in tussen elke stap, of tussen de belangrijkste stappen. De display (met stap-id, tijd en sleutelwaarden) kan in plaats van naar het scherm eventueel ook naar een tijdelijke tabel, die later geanalyseerd wordt. Deze analyse geeft de

Extra testtools

Naast de mogelijkheden van het rdbms voor performance-analyse kunnen producten van andere leveranciers nuttig zijn. Zo biedt Performance Studio van Rational de mogelijkheid de SQL-statements te traceren die van de client naar de server gaan, het antwoord (resultset), returnstatus, begintijd en eindtijd te bekijken en meer. Ook kan met dit product de belasting van een groot aantal (1000) gebruikers op de server gesimuleerd worden door eenmaal een serie van enkele transacties uit te voeren en dit via het gewenste aantal sessies (bijvoorbeeld 1000) tegelijkertijd na te spelen.
Het is belangrijk een dergelijke test in een vroeg stadium van het project uit te voeren met enkele voor dit doel geschreven transacties, om te controleren of de betreffende architectuur en de geplande hardware het (maximaal) verwachte aantal transacties inderdaad kunnen verwerken. Mocht dat niet het geval zijn, dan is het nog niet te laat om maatregelen te nemen.

stap die de meeste tijd kost -relatief ten opzichte van wat verwacht mag worden!- en dus het grootste probleem veroorzaakt. Bedenk dat het kan gebeuren dat aan de stap die het meeste tijd kost niets te verbeteren valt, terwijl het geheel toch beter kan omdat dat andere stappen wel sneller zijn te zetten.

De betreffende stap, mogelijk een ingewikkeld select statement, moet verder geanalyseerd worden. Het rdbms heeft mogelijkheden voor het tonen van:

- de wijze waarop de optimizer de query heeft uitgewerkt;
- doorlooptijd en cpu-tijd;
- de I/O (fysiek naar schijf of logisch naar pages in memory) voor elke tussenstap.

Bekijk hoe reëel deze keuzen en waarden zijn ten opzichte van wat de dba ziet als meest ideale oplossing. Mogelijk kiest de optimizer een verkeerd pad en moet je de query wat anders schrijven of splitsen om het probleem op te lossen. Mogelijk wordt een tabelscan

Houd een goede prioriteit voor ogen (...) hardware is soms goedkoper dan de tijd van een dba

uitgevoerd en kan een extra index soelaas bieden. Bij veel fysieke I/O helpt mogelijk het toewijzen van meer geheugen aan de databaseserver. Verdere uitwerking van dit onderwerp is te productafhankelijk. Hiervoor is een cursus of handboek van de leverancier dan ook noodzakelijk.

Houd bij het oplossen van de performanceproblemen een goede prioriteit voor ogen. Het heeft geen enkele zin zeer incidenteel gebruikte functies (jaarwerk) te versnellen van 60 naar bijvoorbeeld 45 minuten; het heeft zelfs weinig zin als dit naar 20 seconden zou kunnen. Begin dus bij de grote problemen in de veel gebruikte functies, en pak daarna de grote problemen aan in weinig gebruikte functies en de kleine problemen in veel gebruikte functies.

Kleine (performance)problemen in weinig gebruikte functies zijn waarschijnlijk de moeite van het oplossen niet waard. Als daarna nog steeds een algemeen performanceprobleem bestaat, kan het efficiënter zijn wat extra hardware (snellere processor, meer memory) te kopen dan nog meer tijd en energie in deze problemen te steken - hardware is bij deze laatste categorie problemen al gauw goedkoper dan de tijd van een dba... Vanzelfsprekend is dit ook afhankelijk van de situatie: draait het systeem op één locatie of bijvoorbeeld op duizend bankkantoren? In het laatste geval zou het bedrijf duizend stuks hardware moeten aanschaffen.

AANBEVELINGEN

In deze artikelreeks is een groot aantal aspecten beschreven waarmee de dba -en ook de applicatie-architect- van een systeem te maken krijgt in de opzet van zijn systeem. Door met deze zaken rekening te houden is een goede performance mogelijk, maar nog niet gegarandeerd. De beschreven aspecten zijn nog algemeen en niet toegespitst op een specifiek rdbms-product, dat weer zijn eigen mogelijkheden en beperkingen zal kennen. Een cursus of handboek voor tuning van het op het project gebruikte rdbms zal dan ook noodzakelijk blijven voor een verdere diepgang. De punten in dit artikel zullen een dba echter wel helpen bij de algemene opzet (architectuur) van het systeem en de basiskennis geven voor een leverancierspecifieke cursus of handboek.

Het is belangrijk performance te testen in een vroeg stadium van een project. Gebruik daarbij enkele specifiek voor dit doel geschreven transacties. Zo'n test brengt aan het licht of de gekozen architectuur en de geplande hardware het (maximaal) verwachte aantal transacties inderdaad aankunnen. ●

Literatuur

1. Loonen, *Gegevenskoppelingen in een 4GL/RDBMS omgeving.* Database Magazine 8/1996.
2. Loonen, *Gebruik van afgeleide gegevens in ontwerp en bouw.* DB/M 1/1997.
3. Loonen, *Modelleren van subtypes.* DB/M 1/1999.
4. Loonen, *Naamgeving van objecten in een RDBMS.* DB/M 4/1998.
5. Loonen, *Mutatierapportage, de tijdgeest van de database.* DB/M 3/1999.
6. Loonen, *Het schrijven van een nette WHERE clause.* DB/M 7/1997.
7. Loonen, *Gegevensmodel van een distributed data dictionary.* DB/M 4/1997.
8. Loonen, *Ontwikkelstraat, hergebruik door inzet van architectuur.* Software Release 7-8/2000.
9. Smit e.a., *In memory-database, einde van een religiestrijd?* DB/M 3/2000.
10. R.F. van der Lans, *Tijd was rijp voor veelbelovend TimesTen.* DB/M 6/2001.

Toon Loonen (toon.loonen@cgey.nl of toon.loonen@inter.nl.net) is als consultant werkzaam bij Cap Gemini Ernst & Young.