Oracle Forms web deployment (2)

Performance en schaalbaarheid

In het voorgaande artikel over kwaliteitszorg van een Oracle Forms omgeving (zei Optimize nr. 3/2005) is een aantal tips gepresenteerd waarmee de gebruiksvriendelijkheid van de interface eenvoudig verbeterd kan worden. In dit vervolgartikel zal worden ingegaan op performance-gerelateerde aspecten van een webgebaseerde Oracle Forms omgeving, op het vooraf identificeren van eventuele configuratie- en schaalbaarheidsproblemen van de applicatieserver, maar ook op de werkelijke 'dikte' van de client. In de praktijk blijkt dat het gelijktijdig gebruik van meerdere Oracle Forms applicaties behoorlijke capaciteitseisen stelt aan de werkplek.

De performance en schaalbaarheid van een Forms-omgeving heeft grote invloed op de gebruikersacceptatie ervan. NET als de gebruiksvriendelijkheid van een Forms-applicatie is het geen overbodige luxe, voordat de applicatie in productie wordt genomen, hieraan de nodige aandacht te besteden. Een activiteit die valt onder de noemer kwaliteitszorg. Een belangrijk aspect van kwaliteitszorg is de mate waarin voldaan wordt aan de eisen en verwachtingen van de eindgebruikers. Voor Oracle Forms zijn de eisen en verwachtingen vaak direct gerelateerd aan ervaringen met een voorgaande versie. Het vaststellen van verschillen in de interface, zoals dat in een vorig artikel is gedaan, is eenvoudig vergeleken met het vaststellen van de performance en schaalbaarheid. De belangrijkste redenen hiervoor zijn dat het meetbaar maken van de performance en schaalbaarheid niet eenvoudig is en dat performance- en schaalbaarheidsverschillen zich meestal pas manifesteren wanneer een Oracle Forms-applicatie op grotere schaal wordt gebruikt.

Meten

Performance en schaalbaarheid zijn nauw aan elkaar gerelateerd. De schaalbaarheid van een Oracle Forms-omgeving geeft aan in welke mate het aantal eindgebruikers kan toenemen binnen de huidige configuratie zonder noemenswaardige degradatie van de performance. Kortom, je meet de performance om de schaalbaarheid van een Forms-omgeving te bepalen. Maar hoe meet je de performance van een Forms-omgeving zodat je van tevoren weet hoeveel hardware nodig is en hoeveel eindgebruikers de Forms-omgeving maximaal aankan? Bij WebForms zijn de interface en de forms engine uit elkaar getrokken. De interface draait op de werkplek binnen een browser en de forms engine op de applicatieserver. Tussen de interface en de forms engine worden gegevens uitgewisseld op basis van het bekende HTTP-protocol.
Het HTTP-protocol is gebaseerd op het vraag-en-antwoord-model. De client vraagt 'iets' aan de forms engine en de engine geeft altijd antwoord. Door met een tool de tijd te meten tussen het stellen van een vraag en het ontvangen van een antwoord kan inzicht worden verkregen in de reactiesnelheid van

Netwerkcomputer

De Network Computing Architecture, NCA, was een klein decennia geleden de verwezenlijking van Larry's visie op de toekomst. Het gedachtegoed van NCA was gestoeld op het volgende uitgangspunt: het realiseren van een taalonafhankelijk, gedistribueerd applicatiemodel waardoor de desktop-computer vervangen kan worden door een goedkopere terminal, de netwerkcomputer. Dat is bedoeld voor het verbeteren van de interoperabiliteit tussen verschillende softwarecomponenten, het vereenvoudigen van het beheer en het drukken van de kosten. Als onderliggende technologie was een hoofdrol weggelegd voor CORBA in combinatie met IDL waarbij Java nog een ondergeschikte rol speelde.
In de loop der tijd is het iets anders gelopen. De taalonafhankelijkheid heeft plaats gemaakt voor platformonafhankelijkheid en de onderliggende techniek is nu volledig gebaseerd op Java. Daarbij heeft de netwerkcomputer het levenslicht niet gezien en is afstand genomen van Microsoft-technologie. Ironisch genoeg kan je stellen dat het Microsoft .NET-platform meer eigenschappen van het oorspronkelijke NCA in zich draagt dan het Java-platform van vandaag.

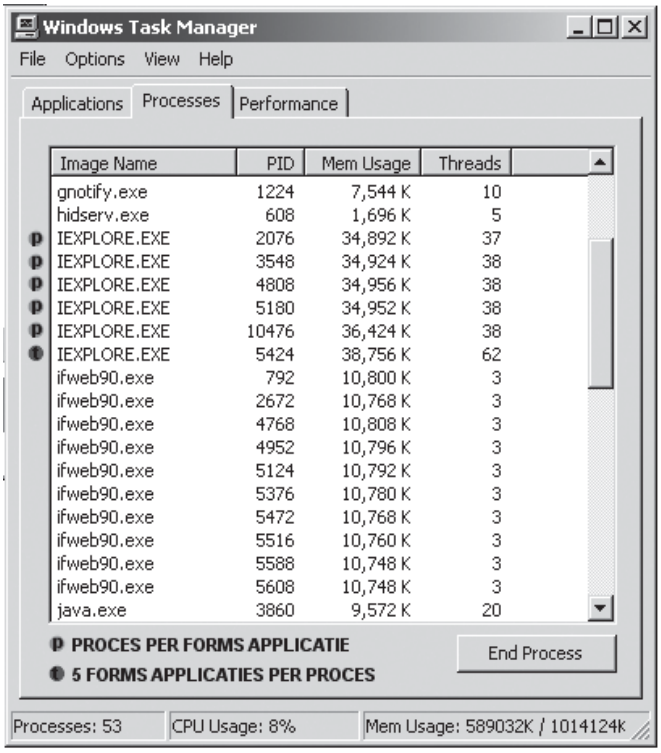
de forms engine. De reactiesnelheid, of responstijd, is vervolgens een goede indicatie voor de performance van de Forms applicatie. Wanneer de forms engine en/of de database overbelast raken dan zal de reactiesnelheid afnemen en dus de performance.
Een slechte performance van de forms engine betekent automatisch dat de eindgebruiker een slechte performance ervaart. Maar een goede performance van de forms engine betekent overigens nog niet dat de eindgebruiker ook een goede performance ervaart. Het kan natuurlijk gebeuren dat de werkplek zelf overbelast is en dat is met name voor Oracle Forms niet ondenkbaar.

Performance van de werkplek

Op het internet is nog een nieuwsitem te vinden van een aantal jaar geleden waarin een vooraankondiging werd gedaan over de komst van Oracle WebForms als opvolger van Forms 4.5. In deze aankondiging werd een tipje van de sluier opgelicht over het nieuwe applicatiemodel van Oracle Forms. WebForms zal uit twee componenten gaan bestaan: een client- en een servercomponent. De client zal geïmplementeerd worden als Java Applet en de servercomponent als een NCA Cartridge (zie kader). WebForms zal centraal worden beheerd en aangeboden. Verder kenmerkt WebForms zich door een nette 'thin client' implementatie, waarbij gesteld wordt dat de client ongeveer 100K geheugen in beslag zal nemen.
In de context van dit artikel is het verschil in geheugengebruik van de client wel het meest opzienbarend. De praktijk leert dat een gemiddelde Forms applicatie al snel 30 megabyte nodig heeft aan geheugen. Dat is 300 keer (!) zoveel geheugen dan aangekondigd. Zonder in een discussie te belanden over wat de definitie van een 'thin client' is, voelt iedereen wel aan dat 30Mb geen thin client is. Zeker niet als er meerdere Forms applicaties tegelijkertijd draaien. Bij performance-problemen met een WebForms applicatie zijn we geneigd de oorzaak te leggen bij het netwerk, de applicatieserver of de database. Meestal terecht maar in sommige gevallen kan het beschikbare geheugen op de werkplek de oorzaak zijn van performance-problemen. Er zijn organisaties die zich genoodzaakt zagen alle werkplekken te voorzien van extra geheugen om meerdere Forms-applicaties goed te kunnen gebruiken.

Thread en proces

Er is echter een alternatief om het geheugengebruik van meerdere Forms-toepassingen zoveel mogelijk te beperken. Het alternatief is gebaseerd op het verschil tussen een thread en een proces. Wanneer een applicatie, zoals een browser, wordt gestart dan wordt er een proces gestart waaraan een deel van het geheugen wordt toegekend. Met behulp van het tool taakbeheer (Windows) kan op ieder moment inzicht worden verkregen in de draaiende processen en het bijbehorend geheugengebruik. Het meerdere keren opstarten van eenzelfde toepassing resulteert in evenzoveel processen en dito geheugengebruik. Processen zijn volledig autonoom; er kan niks worden gedeeld. Deze eigenschap heeft tot gevolg dat voor iedere Forms-applicatie een nieuwe Java-omgeving (JRE) geladen wordt, en het is juist de Java omgeving die veel geheugen consumeert, met name de nieuwste Java-versie 5.0. In tegenstelling tot een proces kan een thread wel het geheugen delen. Een thread behoort altijd tot een proces. Binnen een proces kunnen meerdere threads tegelijkertijd draaien waardoor bij de eindgebruiker de indruk gewekt kan worden dat er meerdere applicaties draaien.
Wanneer een browser opgestart wordt vanuit het startmenu wordt er altijd een nieuw proces gestart. Wanneer vanuit de browser een nieuw venster wordt geopend (file->new->window) dan wordt er niet een nieuw proces gestart maar een nieuwe thread binnen het bestaande proces. Dit kan met het tool taakbeheer eenvoudig worden vastgesteld. Het op deze manier opstarten van meerdere Forms-applicaties heeft tot gevolg dat er geen nieuwe Java-omgeving wordt geladen maar dat de initieel geladen Java omgeving wordt gedeeld door de verschillende Forms-applicaties en dat scheelt aanzienlijk in het geheugengebruik.



Afbeelding 1. Tien Forms-applicaties: vijf als thread en vijf als proces.

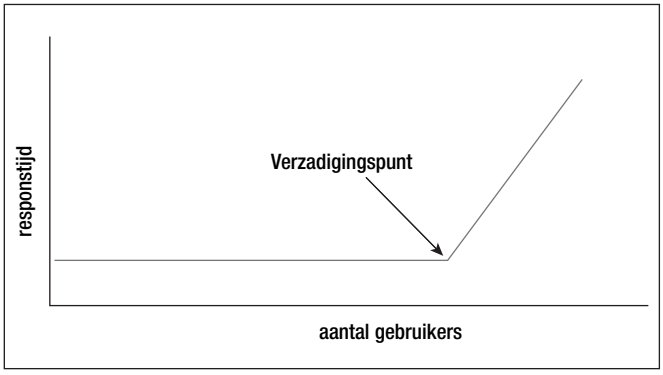
Startpagina

In afbeelding 1 zijn tien Forms-applicaties gestart (ifweb90.exe), waarbij de eerste vijf processen (IEXPLORE.EXE) overeenko-

men met vijf Forms-applicaties en het laatste proces ook met vijf applicaties, maar dan middels vijf threads. Het opstarten van een Forms-applicatie als een thread kan ook programmatisch door een 'startpagina' te ontwikkelen en met behulp van JavaScript de desbetreffende Forms-applicaties te starten. Nog mooier is het om de startpagina zelf met Oracle Forms te realiseren door de verschillende Forms-applicaties op te starten met de functie web.show_document. Het opstarten van een Forms-applicatie als thread heeft ook een aantal consequenties. Ten eerste dient iedere Forms-toepassing dezelfde Java-versie te gebruiken. In de meeste gevallen betekent dit dat voor iedere toepassing dezelfde JInitiator plugin gebruikt dient te worden. Ten tweede kunnen gebruikersfuncties in de Forms-applicatie alleen na elkaar worden uitgevoerd. Een (lange) operatie in één applicatie betekent dat de andere applicaties tijdelijk niet zullen reageren. Ten derde betekent een applicatiefout in één van de browservensters abrupte beëindiging van alle browservensters en dus alle Forms-applicaties.

Performance van de applicatieserver

In de praktijk is gebleken dat het bepalen van de schaalbaarheid van een Forms-omgeving een specialistische aangelegenheid is. Het vaststellen van de schaalbaarheid op basis van een kleinschalige simulatie of op basis van bekende benchmarks is een onbetrouwbare methode. Hieraan liggen een aantal oorzaken ten grondslag. In de eerste plaats is het resourcegebruik (geheugen en rekencapaciteit) in relatie tot het aantal eindgebruikers niet lineair. 100 eindgebruikers gebruiken niet twee keer zoveel resources als 50 eindgebruikers. Ten tweede is het resourcegebruik afhankelijk van de complexiteit van de Forms-applicatie (open forms, aantal blocks/canvases en windows) en ten derde is de resourcebehoefte sterk afhankelijk van het 'eindgebruikersgedrag'. Hiermee wordt bedoeld de intensiteit en de wijze waarop de Forms-applicatie wordt gebruikt.
Kortom, geen Forms-omgeving is hetzelfde dus ook niet de schaalbaarheid.
De schaalbaarheid van een Forms-omgeving is te bepalen door middel van het meten van de responstijd van de Forms engine zoals eerder beschreven in dit artikel. Hiervoor zijn een aantal specialistische tools op de markt beschikbaar. Deze tools worden load- of stress-testtools genoemd. Oracle heeft zelf geen testtool in haar productportfolio maar wellicht na een volgende overname wel.
De schaalbaarheid van een Oracle Forms-omgeving wordt voornamelijk bepaald door het beschikbare (fysieke) geheugen en de rekencapaciteit van de applicatieserver. Kenmerkend voor een Oracle Forms-omgeving is dat de gemiddelde responstijd nagenoeg constant is bij een toenemend aantal eindgebruikers totdat een zeker omslagpunt wordt bereikt. Vanaf dit punt zal de gemiddelde responstijd sterk toenemen waardoor de performance significant degradeert. Dit omslagpunt wordt het verzadigingspunt (eng: saturation point) van de applicatieserver genoemd(zie afbeelding 2).



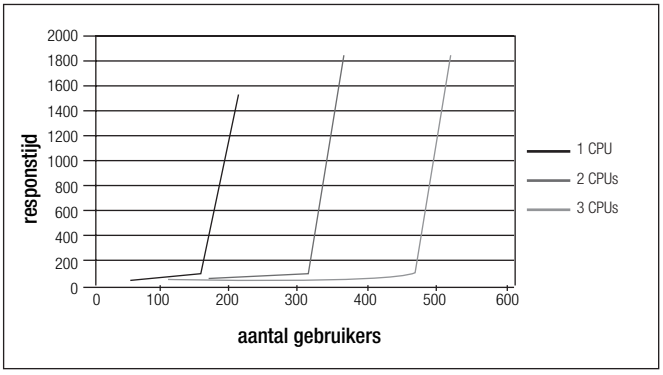
Afbeelding 2. Het verzadigingspunt van de applicatieserver.

Lineair verband

Met het verzadigingspunt is vastgesteld dat de maximale schaalbaarheid van de server is bereikt. Met het toevoegen van extra geheugen of rekencapaciteit kan het verzadigingspunt worden verlegd. Er zal vervolgens moeten worden bepaald wat de oorzaak is van het verzadigingspunt: geheugen of rekencapaciteit. De ervaring leert dat er wel een lineair verband bestaat tussen het geheugen/de rekencapaciteit en het verzadigingspunt. Verdubbel je het geheugen dan kunnen er twee keer zoveel eindgebruikers worden bediend. Als de Forms-omgeving bestaat uit een cluster van gelijkwaardige applicatieservers dan volstaat het bijvoorbeeld om voor het vaststellen van de schaalbaarheid slecht één applicatieserver in te zetten. Een manier om te bepalen wat de oorzaak is van het bereiken van het verzadigingspunt, geheugen of rekencapaciteit, is door een resource te beperken. Door bijvoorbeeld het aantal CPU's te beperken en de hoeveelheid geheugen te vergroten kan een CPU-begrensde test worden uitgevoerd. Het verzadigingspunt is dan met zekerheid het gevolg van de beschikbare rekencapaciteit. Een vergelijkbare test kan worden uitgevoerd voor het geheugen waarmee een geheugen-begrensde test kan worden uitgevoerd. Het resultaat zal grafisch gezien een vergelijkbaar beeld opleveren als afbeelding 3 en 4 laten zien:

Testtool

Het basisprincipe van een load- of stresstest is gebaseerd op simulatie. Door één of meerdere werkprocessen op te nemen en vervolgens meerdere keren af te spelen kan een zeer realistische simulatie van een productieomgeving worden uitgevoerd. Tijdens het uitvoeren van de simulatie worden verschillende statistieken verzameld waaronder de responstijden van de Forms-engine.
Op deze wijze kan nog voordat de nieuwe Forms-omgeving in productie wordt genomen gefundeerde uitspraken worden gedaan over de performance, schaalbaarheid maar ook de

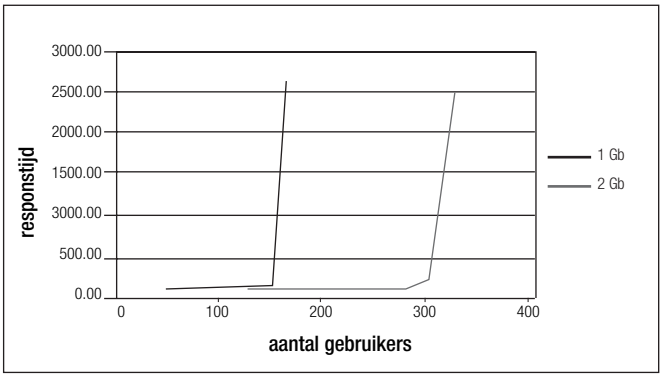


Afbeelding 3. Door het aantal CPU's te beperken kan een CPU-begrensde test worden uitgevoerd.

beschikbaarheid van de Forms-omgeving. Eventuele configuratieproblemen op besturingssysteem-, applicatieserver- en/of databaseniveau komen al tijdens een simulatie aan het licht. Dit zijn over het algemeen parameters zoals het maximum aantal filedescriptors (nofiles), het maximum aantal simultane webserverconnecties (MaxClients) of het maximum aantal databasesessies (SESSIONS, LICENSE_MAX_SESSIONS). Voor een zo realistisch mogelijke simulatie van de productieomgeving is wel een goede kennis van de aanwezige gebruikersprofielen een voorwaarde. Een helpdeskmedewerker zal een Forms-applicatie minder intensief gebruiken dan bijvoorbeeld een data entry-medewerker.
Wanneer bekend is welke eindgebruikersprofielen er zijn, hoeveel eindgebruikers onderdeel uitmaken van zo'n eindgebruikersprofiel en met welke frequentie de verschillende werkprocessen uitgevoerd worden, kan begonnen worden met het voorbereiden en uitvoeren van een succesvolle stresstest.

Pro-actief

Tot nu toe is het testtool beschreven in de context van een migratie om de performance en de schaalbaarheid van een forms omgeving te bepalen. Ook na een migratie heeft het gebruik van een testtool meerwaarde. Bijvoorbeeld om de impact van configuratiewijzigingen meetbaar te maken door de responstijden voor en na een wijziging te vergelijken. Een wijziging kan zijn het toepassen van een patchset of het migreren van een onderliggende database.
Maar ook productieklachten over de performance vanuit de eindgebruikersorganisatie is een goede reden om een simulatie uit te voeren en de klachten meetbaar te maken. In de meest ideale situatie wordt periodiek een simulatie uitgevoerd en de responstijden vergeleken met voorgaande simulaties. Op deze wijze kan pro-actief gereageerd worden op potentiële productieproblemen hetgeen uiteindelijk ten goede komt aan de kwaliteit en dus de gebruikersacceptatie van een Forms-omgeving.



Afbeelding 4. Een vergelijkbare test kan worden uitgevoerd voor het geheugen.

Ir. Jeroen van Schaijk is consultant bij IT-eye (e-mail: jeroen.van.schaijk@it-eye.nl).