Profiling, het bepalen van de flessenhalzen in applicaties, is veelomvattend. Voor multiplatform analyses kwamen we volgens verwachting niet alleen uit bij development-ondersteuners zoals CompuWare en Mercury maar ook bij een echte beheer-vendor zoals de HP OpenView groep, een leverancier die ook nog mede aan de wieg gestaan heeft van ARM (Application Response Measurement), een open standaard die veel met profiling te maken heeft.

HP en de sterke ARM van OpenView

Multi-platform profiling (3)

Met de breakdown-analyses en elementmetingen tijdens productie zal het dus wel goedzitten, maar is het ook voldoende om tijdens ontwikkeling alle details te meten? Hoe verhoudt ARM zich nu tot OpenView en tot populaire middleware zoals WebSphere?

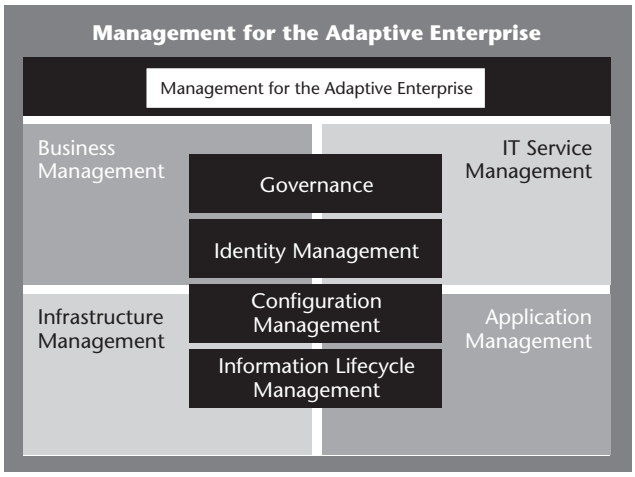
PROFILING De kunst van het vinden van flessenhalzen is niet alleen te bepalen welke binaire actie er te lang duurt, maar ook de oorzaak daarvan te helpen vinden. Een responstijd van pakweg zes seconden waarvan vier seconden in de 2-tier applicatieserver kan immers met slechte codering te maken hebben, maar evengoed met onjuiste server-sizing of een te krappe inter-service connectivity. Vandaar dat we in het eerste deel uit deze serie (Java Magazine nr. 2/2005) de wensen vanuit beide dimensies formuleerden.

De 'flessenhals'-dimensie vertaalt zich in een *breakdown-analyse*. De totaalresponstijd van onze applicatie moet opsplitsbaar zijn in alle tiers, inclusief het netwerk, en zo mogelijk ook nog naar de fysieke server binnen een tier en de extern aangeroepen service daarop. Nog mooier is: naar een individueel gedeployed stuk C# of Java-code binnen die (web-) service. Het is ook nuttig wanneer die breakdown-analyse met allerlei simulaties kan werken: bijvoorbeeld een traag netwerk, of verdubbelde load, of een qua RAM krap gesizede server.

Om nu deze breakdown te kunnen vertalen naar een oorzaak, zijnde applicatiecode of middleware/hardware, zijn *element measurements* nodig. Dat zijn de performance-metingen per server en per middleware-proces binnen die server, zoals bijvoorbeeld de Windows Performance Monitor. We praten vanzelfsprekend over productiemetingen, profiling gaat immers uiteindelijk om het oplossen van productie-flessenhalzen. Essentieel is dan wel dat deze measurements koppelbaar moeten zijn met een breakdown-analyse binnen de productie; daarvoor is het voldoende als de analyse tot op het niveau van een service gaat.

Voor beide soorten metingen is het tot slot vrij belangrijk om historisch vergelijkingsmateriaal te hebben. Dus zaken zoals baselines en *trend-analyses*. Alleen met dit soort gegevens is de impact van metingen voldoende diep, en kan het uiteindelijke medicijn bepaald worden om een flessenhals te verhelpen.

OPENVIEW We noemden HP al een echte beheervendor, en de OpenView lijn is ook gelijk het belangrijkste stuk software van deze leverancier. HP legt zoals bekend de nadruk op hardware en ICT-dienstverlening. Maar juist omdat OpenView 'zo dicht bij de hardware kan opereren' en ook mede dient om onze computersystemen optimaal te gebruiken heeft deze productgroep alle HP-reorganisaties overleefd, en neemt hij zelfs aan belang toe. Hetgeen overigens geen HP-eenkennigheid inhoudt: de naam 'Open' wordt waargemaakt, en OpenView beheert vele merken servers en RISC-werkpaarden en operating systems vanaf Windows en Linux tot aan AIX en OpenVMS. Het zit in de markt van zware beheertool-families, met als belangrijkste concurrenten CA Unicenter en IBM/Tivoli. Omdat profiling onze insteek is zullen de OpenView-lijn, met onder meer



Figuur 1. OpenView kwadrantschema.

concepten zoals business service beheer, automatische serverprovisioning en identiteitsbeheer, niet uitgebreid bespreken. Figuur 1 geeft de totale productfamilie weer. Twee groepen zullen nader worden toegelicht.

a) Infrastructure management. Hierin zitten zowel de 'element measurements' als de breakdown-analyse. De tools OpenView Performance Manager, Internet Services en Transaction Analyzer zijn voor dit verhaal het belangrijkst. Maar in deze groep zit onder meer ook diepgaand netwerkbeheer (Network Node Manager) en generiek serverbeheer (Operations).

b) Application management. Dit is in zekere zin een laag bovenop infrastructuurbeheer; het gaat om dezelfde tools maar nu meer toegespitst op de specifieke applicaties zoals J2EE of .NET, daarbinnen zelfs weer toegespitst op alle aparte brokken middleware, van de Sybase-database tot BEA Weblogic of SAP.

OVTA: PRODUCTIE Transaction Analyzer is HP's belangrijkste tool voor profiling. Hij is gepositioneerd voor wat de naam aanduidt: breakdown-analyses van een enkele (gebruikers-)transactie tot aan alle individuele acties, en hun bijbehorende doorlooptijden. De architectuur is gebaseerd op een (Java-) console, één of meerdere 'measurement servers' en tot slot agents, die hier collectors en monitors heten. In feite zijn er twee producten onder één noemer. OVTA Java Diagnostics kent namelijk een eigen console en meet-techniek, en regelt via JMX vooral element-measurements die dan weer helpen om een totaalrapport per transactie te maken.

OVTA haalt zijn kracht vooral uit de diverse agents. We nemen als uitgangspunt de .NET/COM+ mogelijkheden; de breakdown-opties voor J2EE en CORBA zijn hier sterk mee te vergelijken. De .NET-collector draait op de measurement server en bevraagt via WMI elke Windows-applicatieserver die voor deze profiling relevant is, met uitzondering van de databasemachines. Via WMI worden met name element measurements opgevraagd. Ook zorgt de collector voor het dataverkeer met de monitors. Deze monitors zijn essentieel voor de breakdown analyses. Dat zijn er drie:

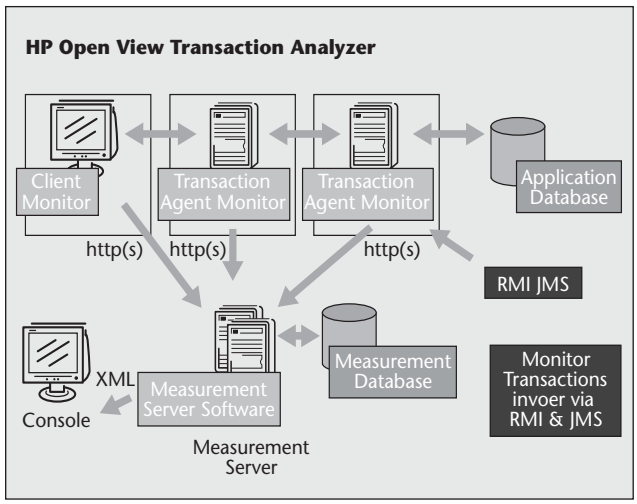
a) De client monitor. Deze meet de responstijd zoals een eindgebruiker hem ervaart, en dat is natuurlijk het startpunt van de breakdown analyse.

b) De 'webserver monitor', deze draait op de IIS machines welke de presentatielaag doen.

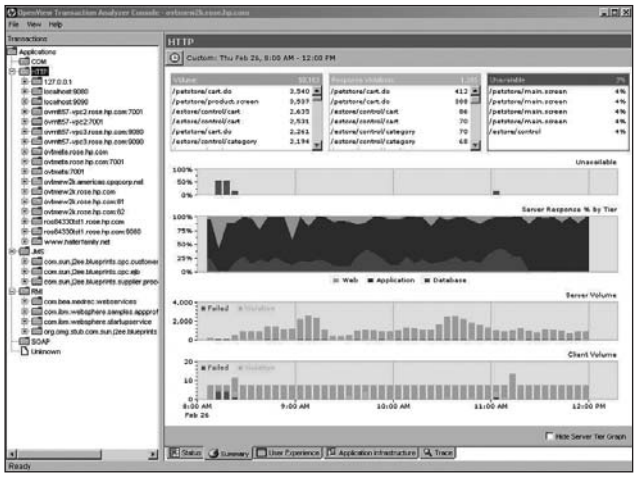
c) De .NET transaction monitor. Deze analyseert lokaal op elke applicatieserver het verkeer, zowel op de overall service als dieper per class en method. Hij toont ook desgewenst de dynamische SQL-calls, al kan hij alleen de totale database-responstijd meten.

Monitors zorgen echter niet alleen voor het 'besnuffelen' van alle calls op de server en hun responstijden, ook doen zij een stuk element measurement. Dat gaat dan met name om dezelfde transacties die ze toch al moeten bewaken. Wanneer bijvoorbeeld de doorlooptijd van transactie ModifyPetAttributes vanaf werkstation NLROT301 gevolgd moet worden, dan meet OVTA ook het totale aantal calls op die transactie per minuut. Hiermee wordt een redelijke indruk gekregen van het sessie-volume, dat nu eenmaal belangrijk bijdraagt aan de transactieperformance.

Eenzelfde verhaal gaat op voor de J2EE-monitors. In aanvulling biedt OVTA in deze tak van sport zoals gezegd nog een extra module, de Java Diagnostics. Diagnostics kent zijn eigen agent per J2EE-server, en besnuffelt het gedrag intensief via JMX. Daarbij krijgen we niet alleen standaard element-measurements, zoals omvang van RAM-beslag en de memory heap, doch ook de allocatie van server-resources per Java-method. Ziedaar een belangrijke brug tussen de breakdown analyse en het servergedrag: methods met performancepro-



Figuur 2. Schema van OVTA.



Figuur 3. OVTA breakdown-analyse.

blemen vertonen ook vaak merkwaardig intern J2EE-gedrag, en gewapend met díe kennis kan worden gekeken naar de schuldige: de ontwikkelaar, de servertuning, middleware-bug of toch een sizing-probleem.

OVTA gebruikt intern ARM, maar daar merken we bij het gebruik weinig van. Wel direct zichtbaar is de integratie met OpenView Internet Services. OVIS is noodzakelijk indien we de service hiërarchie-informatie die we uit OVTA krijgen willen gebruiken voor een complete topologie-mapping, maar dat heeft weinig met profiling te maken. Doch OVIS biedt óók een belangrijke optie voor TA-metingen: composite transactions. Hij kan namelijk niet alleen het feitelijke gebruikersgedrag meten, met een functie die sterk op de OVTA client monitor lijkt, maar hij kan ook zelf robot-transacties afvuren. Voor profiling in productie en het meten van trends op service levels zijn die robotacties beter dan wat willekeurige eindgebruikers doen. Verder levert OVIS zelf ook logmetingen via ARM-bestanden, dus het correleren van gegevens uit beide tools kan het beste gerealiseerd worden in OpenView PM omdat dit immers alle ARM-info kan lezen.

OVTA: GEBRUIK IN ONTWIKKELFASE EN TUSSENSTAND Voordat we naar de Performance-module kijken vatten we eerst samen wat OVTA nu wel en niet kan binnen onze wensenlijst. De insteek is breakdown-analyse maar ook reeds een behoorlijk stuk element-measurements - voorzover de metriek toerekenbaar is aan losse transacties. Overall measurements, die weer meer informatie geven over eventuele sizing-problemen, horen in die andere OpenView modulen thuis. Aan onze wens 'analyse per individueel gedeployed stuk C# of Java code binnen een service' wordt dus behoorlijk voldaan, wat voor een tool met een productieachtergrond knap is. Beperking vergeleken met de 'deep inspection'-opties van de eerder besproken leveranciers is dat we niet vanuit de IDE kunnen analyseren, en dus problemen niet per statement zien doch alleen per class/method. Ook voor simulaties verwijst HP ons

Application Response Measurement

ARM is breder dan alleen HP, en we konden niet om deze standaard heen in onze vogelvlucht-inkijk in de profiling-markt. Hij dateert al van rond 1996, valt tegenwoordig onder de Open Group, en is uitermate simpel qua architectuur:

a) De applicatie, of onderliggende middleware, schrijft in een soort logbestand de elementaire operaties: 'start transactie X' met timestamp en 'einde transactie X', eveneens met timestamp.

b) Dat bestand, op de lokale machine dan wel server, moet door een ARM-agent worden opgehaald en naar de beheerserver verplaatst.

c) Beheertooling consolideert en correleert alle metingen (bij voorkeur met verschillende meetpunt-servers of desktops) en genereert rapportage. Die is gericht op zowel de gemiddelde responstijd per transactie per meetserver, mogelijk gesplitst over dagdelen, als op trends. De kracht van zo'n beheertool kan zijn dat de rapporten ook andere meetdata meenemen, zoals het aantal gelijktijdige gebruikers op elk meetmoment. Enige correlatie tussen die metingen en de responstijd is immers niet onwaarschijnlijk.

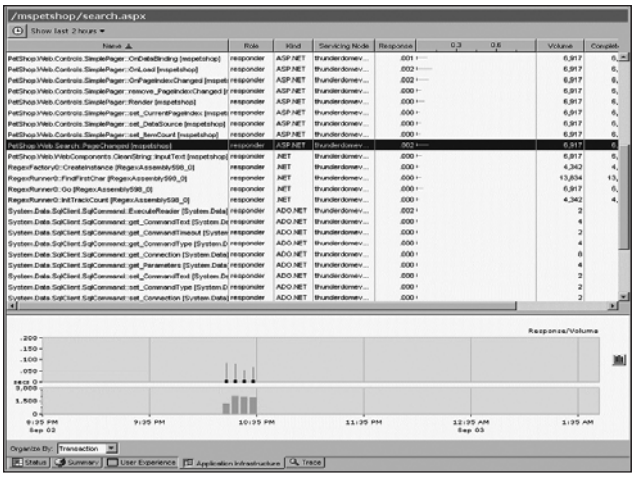
Beheertooling voor taak b) en c) werd en wordt alleen geleverd door HP (OpenView Performance Manager) en Tivoli (onder meer Monitoring for Transaction Performance). Om ARM-logbestanden te kunnen aanmaken zijn toolkits beschikbaar voor Java en C - en die zijn natuurlijk in te kapselen in andere talen zoals C#. Leveranciers met performance-kritische transacties, zoals SAS Institute, hebben ook altijd het nodige aan ARM-evangelisatie gedaan. Die evangelisatie heeft niet kunnen verhinderen dat ARM, zeker de eerste zes à zeven jaar van zijn bestaan, niet erg is aangeslagen.

Dat heeft niet eens zozeer te maken met eventuele concurrerende standaards. Leveranciers zoals BMC en (destijds) CA zagen er inderdaad weinig in en leunden liever op het meten van responstijden via screenscraping op de schermen, maar dat legt het in kwaliteit en met name granulariteit altijd af tegen ARM. De tot op heden vaak onoverbrugbare horde zit in de kosten voor het ver-ARMen van een applicatie. Juist omdat de calls in een applicatie vastgelegd moeten worden, en vaak ook nog eens met een 'debugniveau' (fijngranulair) en een 'productieniveau' (veel beperkter), neemt het voor een wat serieuze applicatie al gauw honderden uren extra in beslag. Zo'n investering in toekomstige beheer-optimalisatie en trending doet veel opdrachtgevers schrikken.

Een beperkte doorbraak in deze patstelling is de keuze van één

naar derde partijen: zaken zoals load-aanpassingen zijn zeer beperkt met OVIS te simuleren, maar voor sizing en netwerk-zaken zouden we toch tools zoals LoadRunner moeten toevoegen. Tot slot moeten we ook constateren dat voor .NET de oorzaak-analyse na de breakdown-meting beperkter is dan voor J2EE, omdat er nog geen .NET-equivalent van OVTA Java Diagnostics is. Deze onevenwichtigheid zagen we ook duidelijk bij Compuware en Mercury, en net als deze partijen hoopt ook HP hem spoedig in de volgende versie opgelost te hebben.

OVO en Performance Manager OpenView Operations is het HP-tool voor totaalbeheer van servers en de daarop draaiende middleware. Er is een basale server agent (voor onder meer Windows- en vele Unix/Linux-smaken) en daarnaast een uitgebreide performance agent, die zelf metingen verricht maar ook ARM logbestanden verwerkt (deze 'performance agent' is een OVO optie, standaard installeert men de Operations Agent inclusief de Embedded Performance Component). Hierbovenop zijn dan tientallen Smart PlugIns (SPI's) leverbaar voor middleware: van database en groupware-product tot applicatie- en webserver. OVO zelf toont ook al een behoorlijke reeks performance 'element measurements', voor Windows onder andere vrijwel de hele PerfMon. ARM-metingen zitten daar dan weer niet bij. Als grote broer positioneert HP daartoe



Figuur 4. Detail-analyse van .NET-applicatie in OVTA.

OpenView PM. Het leunt op dezelfde agent (vroeger bekend als MeasureWare) maar haalt álle informatie daaruit, en is veel uitgebreider in rapportages inclusief baselines en trends. Verder kan PM ook, via ARM logfiles, informatie van OVIS in grafieken meenemen. Het tool kent een server en een browser-client, wat gezien de hoeveelheid grafiek-mogelijkheden een knappe prestatie is.

OVPM vult al onze wensen qua 'element measurements' in, zowel voor het operating system als voor de middleware waarop onze applicaties draaien. Naast

middleware-leverancier, zij het dan IBM, om verARMen in de eigen middleware te doen. DB2 en WebSphere, om precies te zijn vanaf versie 5, staan dit toe. Natuurlijk moeten dan aannames gemaakt worden over wat de start van een transactie is, maar het granulariteitsniveau voor 'automatic ARM' is instelbaar. Het diepst bereikbare niveau is elk aangeroepen Java-object, of het nu een EJB of servlet is of de totale te serveren pagina. Het uitvoerproduct van WebSphere-ARM zijn de standaard ARM-logfiles, die dan met Tivoli of OpenView vergaard en veredeld moeten worden. Automatic ARM kent kinderziekten, zoals het automatisch inserten van Java-code tijdens deployment (hoort u dat, heren EDP-auditors?) en worstelingen met teveel overhead en de 63K-grens van een enkele JVM, maar die lijken allemaal oplosbaar. Als nu ook andere middleware-leveranciers IBM's voorbeeld zouden volgen dan kan met automatic ARM het maximaal mogelijke uit de klassieke ARM-standaard gehaald worden.

Een nóg hoopgevender ontwikkeling voor ARM is analoog aan de middleware-benadering maar op een plek die daar nog beter geschikt voor is: als onderliggende engine voor profiling-beheertools. Als de applicatie-middleware immers zelf kan detecteren wanneer een transactie start, dan kunnen de agents van een CompuWare of OpenView op de applicatieserver dat ook. Als deze nu gewoon hun metingen in ARM-logfiles vastleggen, dan biedt dat rapportage-opties die verder gaan dan het tool zelf. Verder dan wat met handmatig verARMen kan, want zo'n tool-agent kan ook meten op de HTTP-server en in die tier heeft handmatige ARM-code geen effect omdat er geen applicatiecode fysiek 'draait'.

Deze ontwikkeling zien we nu bij OpenView Transaction Analyzer 3.1, dat tot versie 2 een eigen metingen-formaat gebruikte en nu op ARM overgestapt is. Eén van de beweegredenen hiervoor is betere integratie met andere OpenView-metingen, zoals het al genoemde aantal gelijktijdige sessies. Ook Tivoli gebruikt nu meer en meer ARM voor de eigen metingen, al leunt men gedeeltelijk nog op handmatig ver-ARMen. Zelfs CA, nooit een groot ARM-fan geweest, kreeg de technologie binnen door de Concord-overname en zal ARM in het nieuwe Unicenter Application Response Measurement volop gaan ondersteunen.

.NET worden vele J2EE-producten via SPI's ondersteund, en ook het interne functioneren van de HTTP-server en de database kan via deze wijze bewaakt worden. Verder kan HP natuurlijk ook element measurements over het netwerk leveren, via OpenView Network Node Manager. Dat laatste hadden we min of meer buiten onze wensenlijst, want wanneer de breakdown analyse op een netwerkprobleem duidt dan moet sowieso een heel arsenaal aan tooling van deze specifieke ICT-afdeling worden ingezet.

Rond OVPM levert HP nog twee interessante profiling-gerelateerde tools. De eerste heet JMX Metrics Builder, en vergemakkelijkt het aanroepen van de JMX beheer-API vanuit onze applicatie. Die API, gewoonlijk gebruikt door tools als OVO en OVTA om de J2EE-server te bevragen over zijn gezondheid, kan nu worden gebruikt om aan onze applicatie extra informatie te vragen specifiek over het gedrag op bepaalde tracepoints. Daarna kan OVPM ook deze informatie opvragen en in zijn analyses meenemen.

Het tweede tool heet AMIT, oftewel de mond vol van OpenView Application Management Integration Toolkit. AMIT ondersteunt het bouwen van beheerbare .NET-applicaties, en de bouwersversie is dan ook geïntegreerd met Visual Studio. AMIT staat toe om extra gezondheidsinformatie binnen de applicatie te meten en die vervolgens weer via WMI te laten opvragen door de OpenView Performance Agent. Ook kan een operator via WMI bepaalde processen in onze applicatie herstarten. Beide tools zijn vrij uniek in de markt en kunnen een applicatie zeker meer beheersbaar maken, maar lijken toch slechts voor een nichegebruikersgroep nuttig. Ze eisen immers handmatige aanpassing per applicatie, net als het uitstervende 'handmatig verARMen'. Zoiets werkt slecht als business case, het is immers een "investering in toekomstige beheer-optimalisatie en trending" per losse applicatie in plaats van eenmalig door een systeembrede tool-aanschaf.

ARM fungeert als een 'sterke lijm' - al heeft het natuurlijk niet de allure van de sterke arm der wet

Een laatste wens voor de element-metingen is dat ze "koppelbaar moeten zijn met een breakdown-analyse binnen de productie". Dat is precies wat OpenView doet door de wijze van meten binnen OVPM en OVTA. Ten eerste biedt OVTA zelf ook element measurements specifiek voor de bewaakte services. Ten tweede bieden de OVTA Java Diagnostics aanvullende metingen tot op Java method-niveau (dus nog niet voor .NET). De koppeling is dus niet zozeer van OVTA-breakdowns naar OVPM-elementmetingen, maar binnen OVTA zelf. Daarbij is het wél weer essentieel dat de element measurements van OVTA te relateren zijn aan de server-brede metingen van OVPM en dat alle metingen via ARM in een enkel rapport zijn samen te brengen. Zonder het totaalplaatje van PM zouden de oorzaak-aanwijzingen van OVTA, als ze niet naar developer-fouten wijzen, immers onvoldoende specifiek zijn.

De sterke ARM Als we de hele OpenView-suite in profiling-perspectief bekijken, dan fungeert ARM als een 'sterke lijm' - al heeft het natuurlijk niet de allure van de sterke arm der wet. Die lijm helpt om de brug tussen de profiling en het totale productiesysteembeheer uitermate sterk te maken, ietsje beter bij Java dan bij .NET. Bij een leverancier zoals HP zitten deze tools tenminste allemaal in één hand, terwijl bij een partij als Compuware alsnog een ander tool voor het productionele infrastructuur- en applicatiebeheer nodig is.

Ook zagen we het bredere perspectief van ARM, dat mede bij leveranciers als IBM en CA een toenemend belang heeft en kan helpen om profiling meer gezicht te geven binnen ons ICT-beleid. Het enige minpuntje dat we bij de profiling-kant van OpenView kunnen geven is dat het voor pure ontwikkelaars-profiling, met simulatie van omstandigheden en analyse tot op losse coderegels, minder diep gaat dan het eerder besproken aanbod vanuit twee development-ondersteuners. Maar dat is een bewuste positionerings-keuze van HP, die wordt gecompenseerd door de sterkten aan de productiekant. Al met al biedt OpenView, en vooral het pionierende Transaction Analyzer, exceptioneel goede profiling-informatie.

Erik de Ruijter RI.