Cluster-omgeving vindt commerciële toepassing

Shared Data Clusters

Bram Dons

Cluster-systemen zijn al geruime tijd bezig de bestaande traditionele super-computers te vervangen. Tot voor kort volstrekte deze transitie zich voornamelijk binnen de exclusieve wereld van high-performance computing. Door nieuwe ontwikkelingen op het gebied van computer-architectuur zal er de komende jaren echter ook een doorbraak in de toepassing van cluster-systemen binnen de commerciële omgeving plaatsvinden.

Hedendaagse op Intel-gebaseerde cluster-systemen zijn goedkoper dan super-computers en kunnen door middel van aggregatie óók een hoge mate van computing power bereiken. De nieuwe cluster-systemen worden samengesteld op basis van meerdere PC's of blades, die meestal via een snel netwerk (Myrinet, InfiniBand, Gigabit Ethernet) met elkaar verbonden zijn. Een tweede public netwerk zorgt dan voor de verbinding met de 'buitenwereld'. Tot op heden werden PC-gebaseerde cluster-systemen voornamelijk in de wetenschappelijke omgeving gebruikt, zoals nucleair onderzoek, moleculaire simulatie, weervoorspelling en car crash-simulatie. De in onderzoekslaboratoria gebruikte cluster-systemen kunnen daarbij uit duizenden PC's bestaan waarbij, vanwege de bekende licentie-voordelen, meestal grote aantallen Linux servers worden toegepast.

Shared Data

Clusters zonder 'shared storage' waren tot voor kort niet erg populair voor gebruik in de zakelijke omgeving, omdat deze applicaties vaak meer I/O- dan 'computing'-intensief waren. Sinds de komst van het IP- en FC-gebaseerde Storage Area Network (SAN) is daar nu verandering ingekomen. In een dergelijke architectuur heeft elke node, bij koppeling van iedere cluster-node aan een SAN, direct toegang tot de binnen het SAN opgenomen opslagsystemen. Deze koppeling transformeert de traditionele cluster-architectuur tot een zogenaamde *Shared Data Cluster*. Een dergelijk type cluster-systeem wordt dan ineens interessant voor allerlei zakelijke toepassingen, zoals database-, file-, mail- en web-servers, op basis van een parallel werkend en in hoge mate schaalbaar en beschikbaar systeem. Het is dus niet zo verwonderlijk dat Shared Data Clusters de komende jaren een steeds belangrijker rol gaan spelen binnen de enterprise data center. Het delen van data door meerdere computer-systemen biedt talrijke voordelen. Applicaties kunnen op iedere node worden gedraaid, omdat de door een node benodigde data overal binnen een SAN beschikbaar zijn. Het voorkomt daarmee het onnodig repliceren van data en fragmentatie van opslagruimte. Alle beschikbare opslagruimte bevindt zich in een enkelvoudige grote opslag-pool die, al naar gelang de behoefte, in verschillende opslagvolumes kan worden opgedeeld. Toepassing van shared storage bespaart dus in aantal toegepaste disks en maakt een beter en efficiënter beheer van data mogelijk, waaronder: backups, archivering in het algemeen en remote replicatie voor disaster recovery. De beschikbaarheid van data kan, onafhankelijk van het gebruik daarvan, beheerd worden. Bijvoorbeeld, een bepaald logisch opslagvolume kan meer betrouwbaarder worden gemaakt door toevoeging van mirroring.

Toepassing shared data kent natuurlijk ook nadelen. Een belangrijk nadeel is zijn inherente zwakheid vanwege het feit dat het logisch gezien slechts een enkelvoudige kopie is van een bepaald informatiedeel. Het is daarom mogelijk dat 'zich slecht gedragende' programma's of gebruikers de enkelvoudige logische kopie in een keer 'om zeep' kunnen helpen. Alhoewel data tot op een hoog niveau kunnen worden beschermd tegen hardware-fouten (door bijvoorbeeld de implementatie van redundante hardware-componenten), geldt dit niet voor software-matige of menselijke fouten. Het behoeft dan ook geen betoog dat het uitermate belangrijk is dat er in de software extra voorzieningen worden opgenomen om de gelijktijdige toegang van meerdere gebruikers tot de gedeelde databron in goede banen te leiden.

Single System Image

Een cluster-systeem doet zich voor als een enkelvoudige grote computer, een zogenaamd *Single System Image* (SSI). Bij een cluster-systeem draait op iedere node een software 'stack'. Op het laagste niveau van dit stack bevindt zich het operating system (OS) en op het hoogste niveau de frond-end applicatie, die

uiteindelijk met de gebruiker communiceert. Op elke laag moet de software geschikt zijn om SSI te kunnen toepassen. Een hypothetische cluster OS zou in staat moeten zijn om alle systeembronnen (CPU, geheugen en devices) op een cluster-wide niveau te kunnen beheren. Echter, een dergelijk cluster OS is (zeker commercieel) nog niet beschikbaar, waardoor clusters thans geen SSI op OS-niveau kunnen bieden. Er zijn echter twee programma's die zich, conceptueel gezien, op OS-niveau bevinden en die wel een shared data-architectuur mogelijk maken.

> Een database server kan draaien op een voor clusters ontworpen systeem met 'instances' op meerdere nodes

Ten eerste, de Volume Manager. Deze moet dusdanig zijn ontworpen dat het SSI op volume management-niveau kan bieden.
Een speciaal voor clusters ontworpen bestandssysteem moet dan ook SSI ondersteunen op het niveau van bestandsobjecten.
Het volgende voor SSI vereiste niveau zijn de zogenaamde back-end applicatie-services. Een database server kan op één node draaien, of een voor clusters ontworpen database-systeem (zoals Oracle Parallel Server) waarbij 'instances' op meerdere nodes kunnen draaien. Als de front-end applicatie van de back-end services gebruik maakt, dan weet het niet op welke node de back-end server draait, zodat op dit niveau ook sprake is van SSI.
Op het hoogste niveau kunnen gebruikers op elke node inloggen en in principe elk willekeurig programma draaien. Als de gebruiker niet weet op welke node hij is ingelogd, dan spreekt men ook op dit niveau van SSI.
Er zijn verschillende beheerprogramma's voor de cluster-omgeving beschikbaar. Meest bekend is de Beowulf cluster suite, het is free software op basis van Linux en bevat alle tools voor beheer van een cluster-omgeving. Een ander bekend software-pakket is Parallel Virtual Machine (PVM), ook free software, dat wordt gebruikt voor gedistribueerde applicaties binnen een cluster-omgeving.

Introductie Cluster File System

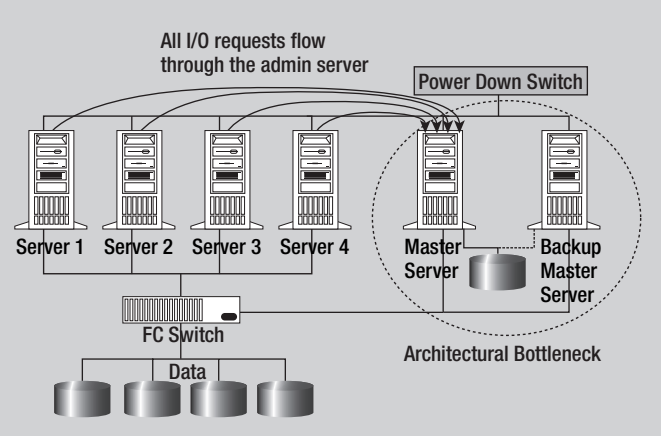
Een van de mogelijkheden om data te delen in een cluster-omgeving is de implementatie van een zogenaamd *Clustered File system* (CFS). Een CFS biedt meerdere nodes de mogelijkheid om applicaties, bestanden, en data te delen waarbij read/write consistentie, en dus de integriteit van data, wordt gewaarborgd.
Cluster file systems zullen in de toekomst steeds vaker binnen data center-architectuur worden toegepast. Daarbij zullen de kosteneffectieve Intel-gebaseerde servers de plaats gaan innemen van de inmiddels te duur geworden proprietary RISC-gebaseerde servers. De verwachting is dat het CFS in de toekomst een meer centrale rol gaat vervullen binnen het data center, meer dan in het verleden het geval was met de single-node file systems. Om deze rol binnen het enterprise data center te kunnen vervullen, moet een cluster file system aan een aantal stringente technische eisen voldoen, te weten:

- compatibel met bestaande applicaties;
- ondersteuning van commerciële werkbelasting, waaronder: intensieve lees/schrijf-bewerkingen, transactionele updates en in groten getale kunnen aanmaken en verwijderingen van bestanden;
- hoge schaalbaarheid en efficiënte samenwerking van meerdere nodes;
- in hoge mate herstelbaar, de mogelijkheid tot herstel bij fouten in hardware of netwerken, met behoud van data-integriteit;
- eenvoudig te installeren, configureren, monitoren en onderhoud.

Alhoewel de noodzaak van de hiervoor genoemde eisen nogal vanzelfsprekend lijkt, is de ontwikkeling van een file system dat aan al deze eisen moet voldoen uitermate complex. Daarbij moeten moeilijke problemen op het gebied van basis-computer-wetenschappen worden opgelost. De meeste uitdaging ligt bij het vinden van een oplossing van een fundamenteel probleem, namelijk: hoe kan de 'state consistency' over meerdere servers worden gewaarborgd bij wijzigingen van deze state door een van deze servers, met daarbij de garantie van een totale recovery.
Er is nog slechts een handvol commerciële implementaties van een CFS voor een deel met succes in de praktijk gebracht. Alle in het verleden ondernomen pogingen hebben geen CFS opgeleverd dat volledig geschikt was voor toepassing binnen het enterprise data center. De tekortkomingen zijn hoofdzakelijk het gevolg van de toegepaste architectuur, dat in een tweetal categorieën kan worden opgesplitst: master server-gebaseerd ontwerp en wetenschappelijk computing file system.

Gelijktijdige toegang

De meeste bestaande CFS's zijn voortgekomen uit de traditionele High-Availability (HA) oplossingen. Deze oplossingen werden oorspronkelijk voor database-applicaties ontwikkeld, waar HA kon worden verkregen door een stel servers in een primairy/backup-, active/passive-configuratie op te nemen die verbonden was met een dual-path SCSI-gebaseerd opslagsysteem. Op elk moment had daarbij slechts één server toegang tot het gemeenschappelijke opslagsysteem. Bij uitval van de primaire server nam de backup-server bezit van het opslagsysteem en vervolgde met het afhandelen van de applicatie-requests. In de loop van de tijd werd deze architectuur uitgebreid door meerdere servers in de cluster-architectuur op te nemen, waarbij men er echter nog steeds vanuit ging dat elke opslagbron op elk moment slechts door één server kon worden gebruikt.
Een vanzelfsprekende uitbreiding voor een dergelijk systeem zou de gelijktijdige toegang vanuit het cluster naar het opslagsysteem moeten zijn. Echter, de bestaande bestandsystemen in deze omgevingen gaan er vanuit dat er geen gelijktijdige toegang kan en mag plaatsvinden: alle bestandssysteem-datastructuren



Afbeelding 1: Master server-gebaseerde CFS architectuur.

(metadata), en de code die van deze data gebruik maakt en wijzigt, gaan er vanuit dat bij de toegang tot de data slechts één server betrokken is. In plaats van een nieuwe architectuur te ontwikkelen, heeft men er voor gekozen om op de bestaande oplossing voort te borduren. Men koos voor de makkelijkste oplossing door een bepaalde node als 'master server' te benoemen, die als primaire taak had de bewaking en controle over alle systeem metadata op zich te nemen. De overige nodes zijn dan genoodzaakt om deze taak aan de master server over te geven, zodat alleen data disk-blocken nog maar over het SAN hoeven worden getransporteerd, zie afbeelding 1.

Beperkingen CFS

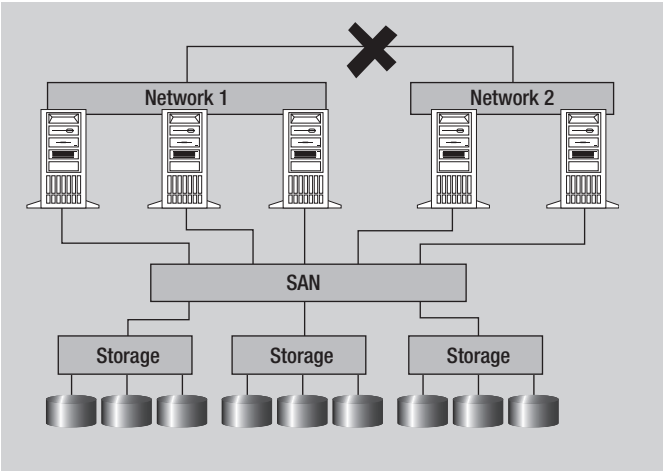
De hiervoor genoemde architectuur kent echter een fundamentele beperking ten aanzien van schaalbaarheid, prestaties en beschikbaarheid. De master server is te allen tijde nodig voor de afhandeling van al het I/O-verkeer (bij de consultatie van file system metadata om bestanden te lokaliseren en creëren, doorlopen van directory's enzovoort). Hierdoor ontstaat, als gevolg van een toenemende belasting, al snel een flessenhals (speciaal bij schijfintensieve applicaties). Dit probleem wordt nog eens verergerd door bijplaatsing van meerdere servers. Omdat het file-systeem volledig afhankelijk is van de master server, vormt deze een single point of failure binnen de cluster-omgeving. Om dit probleem te verzachten kan weliswaar een backup master server worden bijgezet, maar dit leidt weer tot een verslechtering van de beschikbaarheid; bij uitval van de master kan de omschakeling naar de backup server meerdere seconden in beslag nemen. De inbreng van een backup server verhoogt bovendien de complexiteit van de architectuur, want de beheerder wordt dan al met een drietal type server-configuraties geconfronteerd: master-, backup- en client servers.
In ieder bestandssysteem, of dat nu single-node of cluster is, vormt het gebruik van cache de sleutel tot snelle toegang tot de data. Een deel van het lokale server-geheugen wordt voor cache gebruikt waarin de data tijdelijk worden opgeslagen die door applicaties al eerder zijn gelezen op de betreffende node. In een cluster-omgeving heeft iedere server zijn eigen cache en de vraag is natuurlijk: hoe wordt deze consistent tussen alle server-caches binnen de cluster gehouden? De eenvoudigste oplossing is om al het cache uit te zetten, maar dat gaat onherroepelijk ten koste van de prestaties. Een betere methode is om een locking-systeem te implementeren.

Gedistribueerde locking versus gecentraliseerd beheer

Een belangrijke keuze bij het ontwerp van een locking-systeem voor een cluster-omgeving is de vraag hoe het beheer van locking over een of meerdere nodes moet plaatsvinden. Dat betekent in de praktijk: *gedistribueerd* of *gecentraliseerd*. De verschillende CFS-implementaties hebben daartoe verschillende strategieën toegepast. De meeste ontwerpen voegen lock manager-functionaliteit aan de master server toe. Daarbij stuurt iedere node een request naar de master server, voor het verkrijgen van een lock van de data in het cache (op dezelfde manier waarop het bestandssysteem omgaat met de toegang tot de metadata). Echter, deze methode verhoogt ook de belasting op de master server en voegt extra functionaliteit daaraan toe, waarvan weer de beschikbaarheid (door de backup server) gewaarborgd moet worden. Bovendien beperkt deze methode de prestaties van het lokale cache (immers er moet eerst via het netwerk een lock worden verkregen voordat de bewerking pas doorgang kan vinden).

> Een cluster-systeem doet zich voor als een enkelvoudige grote computer

Een ander bekend cluster-probleem is 'split brain' waarbij de coördinatie, als gevolg van een falende netwerkverbinding, tussen de nodes verloren gaat maar waarbij de toegang tot het gedeelde opslagsysteem blijft gehandhaafd. Dat dit tot ongecontroleerde schrijfacties kan leiden, hoeft geen betoog. Split brain en ongecontroleerd gedrag van servers zijn de gebieden waar binnen de bestaande CFS-implementaties nogal verschillen in bestaan in de manier waarop nodes aan een draaiende cluster worden onttrokken (zodat er geen ongecoördineerde updates naar het gedeelde opslagsysteem plaatsvinden).
Kortom, de op master server gebaseerde cluster file systems zijn wel relatief eenvoudig te implementeren als uitbreiding op de traditionele single node-systemen, maar ze hebben wel te leiden onder afnemende prestaties bij een toenemend aantal cluster nodes. Bovendien is de beschikbaarheid niet optimaal.
Al deze tekortkomingen zijn wel in tegenspraak met de algemene doelstelling van een cluster file system: de ondersteuning van meerdere onafhankelijke servers met de waarborg van een goede schaalbaarheid/beschikbaarheid, waarbij de uitval van een individuele cluster node geen beperkingen oplevert voor de cluster-prestaties.



Afbeelding 2: Split brain-voorbeeld.

Gedistribueerde locking

Het alternatief is gedistribueerde locking, waarbij de locking service gelijkelijk over alle cluster nodes wordt verdeeld, waarbij een *symmetrisch cluster-model* ontstaat. Wanneer een of meer lock nodes falen, dan wordt de overeenkomstige lock service door andere active nodes tijdens een failover-proces overgenomen, zonder merkbare gevolgen voor de gebruiker. Gedistribueerde locking biedt een hogere mate van parallellisme dan gecentraliseerde locking, op voorwaarde dat de verschillende nodes op verschillende delen van de data/metadata werken. Echter, bij frequente toegang tot en wijziging van data of metadata, is het beheer bij een gecentraliseerde oplossing optimaler dan bij een gedistribueerde oplossing: wanneer er vaak locking 'conflicten' optreden kan de overhead van gedistribueerde locking hoger zijn dan bij de centrale methode. Ook de lock 'granularity' is van invloed op de prestaties: een kleinere lock granularity betekent meer overhead omdat er meer lock requests optreden, in tegenstelling tot een grote mate van granularity waar zich meer frequent locking-conflicten voordoen.

Sommige producten, waaronder IBM's GPFS, maken afhankelijk van het type data van een aantal technieken gebruik, bijvoorbeeld: byte-range locking voor wijziging van gebruikersgegevens, dynamisch verkozen 'meta nodes' voor centraal beheer van bestands-metadata en gedistribueerde locking voor toewijzing van disk-ruimte.

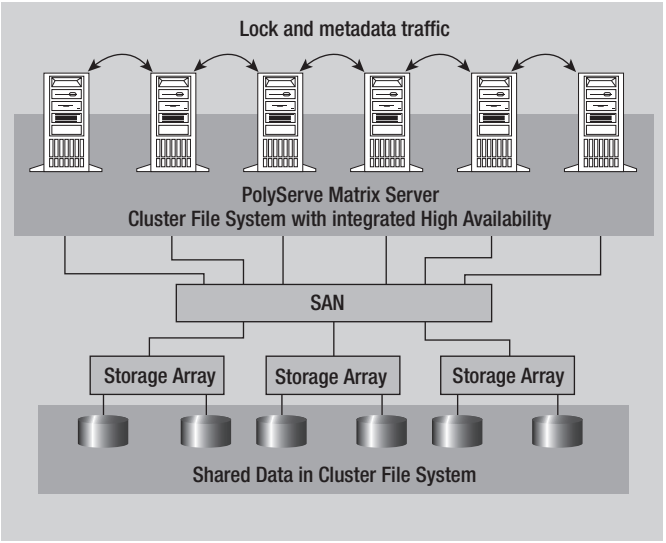
Cluster File Systems

Sommige bestandssystemen bieden een uitbreiding van hun traditionele file server-architectuur met een SAN door de file server clients rechtsreeks toegang te verschaffen tot de data op de SAN disks. Voorbeelden van dergelijke SAN file systems zijn IBM/Tivoli SANergy en Veritas SANPoint Direct. Dergelijke file systems bieden weliswaar een efficiënte toegang tot de data, maar daarbij moeten alle metadata-updates door een centrale metadata-server worden afgehandeld, wat dit type architectuur inherent minder schaalbaar maakt. Sistina's GFS en Vertitas SANpoint Foundation Suite HA zijn andere voorbeelden van centraal beheerde metadata-servers. De Sistina OmniLock-architectuur kent twee type locking-systemen: Single Lock Manager (SLM) en Redundant Lock Manager (RLM), waar in het laatste geval meerdere nodes als backup server kunnen worden bijgeschakeld.

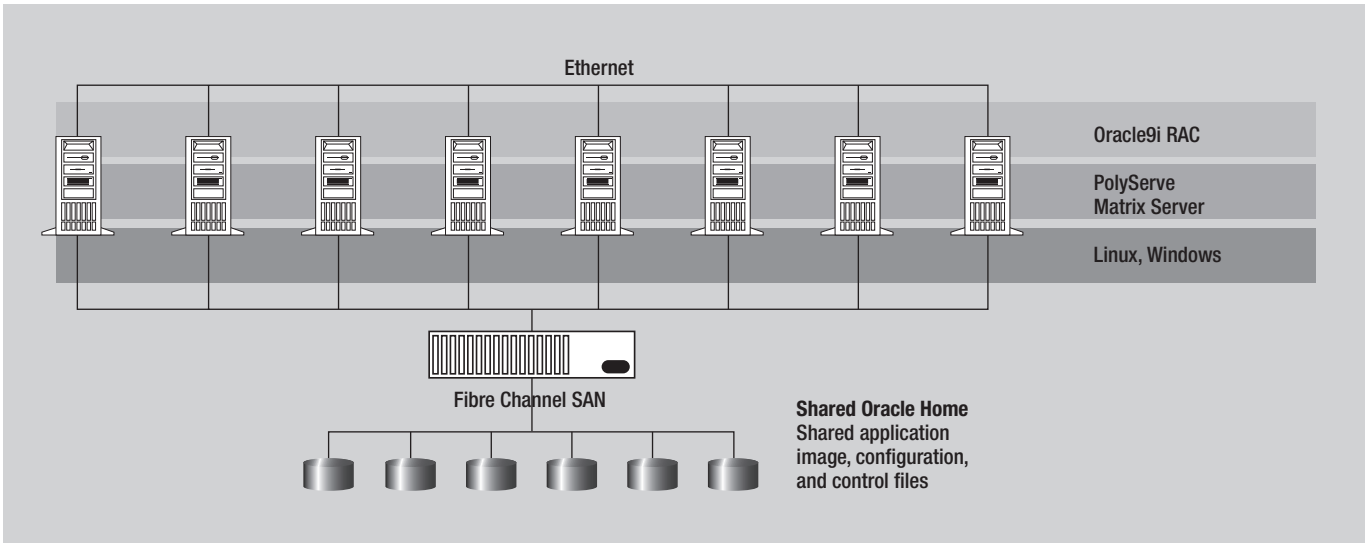
De firma PolyServer heeft met het product Matrix Server een alternatieve oplossing gekozen door van een symmetrische CFS-architectuur gebruik te maken. Zoals eerder beschreven, de locking-taken worden bij het master server-gebaseerde model centraal uitgevoerd, bij het symmetrische model worden ze gelijkmatig over alle nodes verdeeld. Bij PolyServe's volledig symmetrische Distributed Lock Manager (DLM) participeert iedere node in de coördinatie van locks binnen de cluster, zodat de belasting gelijkmatig wordt verdeeld, en wel op een zodanige manier dat geen enkele node een flessenhals oplevert. In geval dat een node uitvalt, wordt de verantwoordelijkheid voor de afhandeling van uitstaande locks door de andere nodes overgenomen. Zelfs bij opeenvolgende fouten, waarbij onverwachts meerdere nodes uitvallen, wordt dit door de resterende nodes opgevangen.

Applicaties voor Shared Data Clusters

Hiervoor is al uiteengezet dat voor volledige implementatie van een SSI-systeem ook de applicatie-laag moet worden aangepast. Zonder een voor parallelverwerking aangepaste applicatie, biedt het gebruik van een SSI-gebaseerd systeem geen enkel voordeel, sterker nog, in bepaalde gevallen kan de verwerking van een applicatie met SSI zelfs trager verlopen dan zonder. De meest voor SSI geschikte commerciële applicaties zijn thans te vinden in het database- en file server-segment, bijvoorbeeld: Oracle9i RAC en IBM DB2, NFS, CIFS, Microsoft SQL Server en IIS.



Afbeelding 3: PolyServer Symmetric CFS-architectuur.



Afbeelding 4: Combinatie Oracle9i RAC en Polyserver Matrix Server.

Oracle9i RAC cluster-systeem

Interessant is de combinatie van Oracle9i RAC met een Cluster File System. Oracle9i RAC kan in twee scenario's worden toegepast: gebruik makende van een gedeeld 'raw device' of een CFS. Belangrijkste voordeel bij toepassing van een raw device is weliswaar een hogere prestatie (in vergelijking met een file system gebaseerde database), maar daarbij heeft men bij Oracle destijds wel een afweging moeten maken tussen betere prestaties en slechtere beheerbaarheid. Omdat er namelijk geen bestanden of directory's op een raw device aanwezig zijn, introduceert de toepassing van een raw device voor de database-beheerder extra complexiteit.

Bij Oracle9i RAC kunnen alleen de control- en data-bestanden op een raw device worden opgeslagen. De binary's en configuratiebestanden moeten in het lokale bestandssysteem van de server worden opgeslagen (of in geval van een cluster-omgeving op iedere node). Oracle9i heeft meer dan 150.000 bestanden in elke Oracle Home. De beheerder moet binnen een cluster-omgeving voor iedere node de configuratie en onderhoud van Oracle Home ter hand nemen, een lastige en tijdrovende zaak dus! Dit lukt nog wel bij enkele nodes, maar bij een cluster-omgeving met tien of meer nodes wordt dit een ondoenlijke zaak.

Polyserve's CFS biedt Oracle9i RAC de mogelijkheid om slechts één kopie van Oracle Home's bestanden te delen over alle cluster-nodes, in plaats van een Oracle Home op iedere afzonderlijke node. Dit wordt mogelijk gemaakt doordat het Polyserver Matrix Server cluster file system zich aan Oracle9i RAC voordoet als een gedeeld raw device, voor de beheerder echter ziet het er als een gedeeld bestandssysteem uit. Daarbij wordt de standaard Oracle journalling en recovery gewoon ondersteund. In een vervolg-artikel wordt afzonderlijk uitvoerig in gegaan op de Polyserver Matrix Server in combinatie met Oracle9i RAC database Cluster.

Bram Dons is onafhankelijk IT-analist.