Unix failover clusters
If you've seen one, you've seen 'm all

Jos Visser
josv@osp.nl

Open Solution Providers
http://osp.nl

OPEN SOLUTION PROVIDERS

23 mei 2001

Inhoudsopgave

1 Introductie

Laten we eerlijk zijn: Wie durft er vandaag aan de dag nog te beweren dat zijn applicatie zodanig onbelangrijk is dat het niet nodig is speciale maatregelen te nemen teneinde de beschikbaarheid van die applicatie te waarborgen. Niemand toch? Aldus is de setting geschapen van een ongekende markt op het gebied van hogere, verhoogde of zelfs continue beschikbaarheid. Zoals de meeste problemen is die van beschikbaarheid er eentje waarbij vermeerdere toepassing van de productiefactor "geld" leidt tot een verhoging van de beschikbaarheid. Echter, voor niets gaat de zon op, en om de pret die voornoemde constatering met zich mee kan brengen weer enigzins te drukken is ook op beschikbaarheid de ijzeren wet der niet proportionele meeractiviteit (onder de meesten waarschijnlijk bekend als de "Wet van de toe- en afnemende meeropbrengsten") van toepassing. Een andere economische wet die ik hier graag wil noemen is die van het "Afnemend Grensnut". Niet dat die er nu zo bijzonder veel mee te maken heeft, maar ik wil u als lezer toch het gevoel geven dat ik vroeger nog wel eens heb opgelet op school.

Hoe het ook zij, om de beschikbaarheid van een applicatie te verhogen wordt meestal gegrepen naar het wapen der redudante hardware. Het is best jammer als je bedenkt dat dit van de redenen der ongeplande downtime nu juist de minst frequente van die redenen oplost, maar je kunt dan ook niet alles hebben (waar zou je het laten). Het oplossen van de meest voorkomende redenen zou echter vereisen dat we structureel slimmere programmeurs en beheerders engineeren, en dat is niet mogelijk, danwel wellicht niet wenselijk[1].

2 Ik wil meer!

Het verbeteren van de beschikbaarheid door middel van het toepassen van meer hardware kan op drie manieren:

1. Standby systemen
2. Clusters
3. Fout tolerante systemen

Bij de eerste oplossing zetten we gewoonweg een duplicaat systeem neer en zorgen we er handmatig voor dat bij het instorten van een systeem de applicatie naar het andere systeem wordt overgeheveld. Qua transparantie scoort een dergelijke oplossing natuurlijk niet zo heel erg hoog. Maar, het is makkelijk, relatief goedkoop, en zelfs voor het IT management nog te begrijpen...

Fout tolerante systemen zitten helemaal aan de andere kant van het spectrum. Hier hebben we de redundante hardware dusdanig slim in elkaar gesoldeerd dat, wat er ook gebeurt, we altijd kunnen (blijven) lachen. Alle mogelijke storingen worden volledig transparant opgelost, en dat is fijn. Echter, qua een stukje prijsbeleving is een fout tolerant systeem nogal een aanslag op het spaarvarken van de gemiddelde organisatie.

[1] In het kader van deze opmerking kan ik de geïnteresseerde lezer verwijzen naar het boek "Brave New World" van Aldous Huxley.

Blijft over: clusters: een groepje systemen die nauw samenwerken met als doel om één of meer applicaties verhoogd beschikbaar te houden. In het Nederlandse poldermodel zijn clusters natuurlijk sterveloos populair; clusters vormen de concensus tussen losstaande en fout tolerante systemen: qua kostenplaatje niet veel duurder dan standby systemen, en qua transparantie dicht genoeg bij fout tolerante systemen.

3 Unix clusters

Iedere zichzelf serieus nemende Unix leverancier heeft natuurlijk iets van een clustertechnologie op de markt gebracht. Daarbovenop zijn er zelfs leveranciers die clusteroplossingen in de markt hebben gezet die op meerdere platformen werken. Het doel van al die oplossingen is om door applicaties van node naar node te migreren die applicaties verhoogd beschikbaar te maken. In de populaire lectuur rondom clusters wordt die aanpak doorgaans "failover" genoemd[2].

Niet toevallig lijken de meeste Unix clustertechnologieën als drie druppels water op elkaar. Sterker nog, de ondertitel van deze presentatie luidt zelfs: "If you've seen one, you've seem 'm all". De structuur van de meeste cluster produkten is min of meer hetzelfde, met wat interessante variaties zo links en rechts. Alleen als je clusters vergelijkt op de hoofdelementen van die structuur kun je een beetje intelligent uitspraken doen over het ene product versus het andere.

Ik wil graag in de rest van dit document een overzichtje geven van hoe clusters grosso modo in elkaar steken, en hoe de verschillende cluster pakketten die ik ken op diverse onderwerpen scoren. Zie dit vooral niet als een officieel marktonderzoek of anderzins verantwoorde opsomming. Ik poog niets meer (maar ook iets minder) dan een klein beetje begrip te kweken van hoe Unix clusters in elkaar zitten en hoe verschillende Unix clusteroplossingen bepaalde aspecten hebben opgelost.

4 Uitsluitsel, wie hem toekomt...

Prijsvraag: uit welke Nederlandse TV serie komt de opmerking "Uitsluitsel, wie hem toekomt". Antwoorden kunnen worden ingestuurd naar josvosp.nl. Onder de goede inzendingen wordt een zak chips verloot...

4.1 Cluster lidmaatschap

Zoals ieder zichzelf respecterend groepje wil ook een cluster graag weten wie er eigenlijk lid zijn. Unix clusters hebben meestal geen echte leider, en dit betekent dus dat ieder lid continu druk bezig is om bij te houden wie er nog wel lid zijn van de cluster, en of hij/zij zelf nog wel lid is.

[2] De andere manier van verhoogde beschikbaarheid is via "load balancing". Dit vereist echter applicaties die op meerdere nodes tegelijk actief kunnen zijn.

4.1.1 Heartbeat

Iedere Unix clusteroplossing (die ik ken) lost de basis lidmaatschapsvragen op door een heartbeat protocol. De basis is dat iedere node regelmatig over n of meer netwerken laat weten dat hij/zij nog steeds leeft. Als van een node een tijdje geen hartslagen meer zijn gehoord gaan we er vanuit dat die node is overleden. De cluster reconfigureert dan en verhuist applicaties en resources naar de nog wel levende nodes.

Meestal worden de heartbeats uitgevoerd over twee verschillende TCP/IP netwerken. Er zijn echter nog wel wat interessante opmerkingen te plaatsen over hoe verschillende clusters dit aanpakken:

1. De Veritas Cluster Server (VCS) gebruikt geen TCP/IP, maar een eigen niet routeerbaar netwerk protocol. VCS kan ook heartbeat uitvoeren via een sectie van een gedeelde disk (dus geen netwerk I/O)!
2. Sun Enterprise Cluster vereist twee dedicated TCP/IP netwerken voor heartbeat en andere intracluster communicatie. De interfaces in die netwerken worden automatisch door de cluster software geconfigureerd (met IP adressen uit een speciaal hiervoor (v/d)oor Sun gereserveerd class C netwerk).
3. Naast ethernet worden ook nog wel andere infrastructuren als heartbeat netwerk ondersteund. Onder andere Linux heartbeat, SteelEye Lifekeeper en HP MC/ServiceGuard kunnen ook een seriële null modem kabel als heartbeat mechanisme tussen twee nodes gebruiken. Over die kabel wordt dan geen TCP/IP gedaan, maar een bedrijfseigen protocol. Sun Enterprise Cluster ondersteunt ook SCI (Dolphin) netwerken als heartbeat netwerken (Gigabit speed low latency netwerk). Daarnaast heb ik nog wel eens wat misleide geesten zien discussiëren over het implementeren van heartbeat over infrarood interfaces.

Alle clusters ondersteunen ook het configureren van time outs waarmee kan worden bepaald na hoe lang wordt geconcludeerd dat een node echt verdwenen is. De default waardes van die time outs willen nog wel eens te laag staan!

4.1.2 Split brain

Als een node niet meer op een heartbeat reageert is er sprake van een interessant dilemma. Er kunnen namelijk twee dingen aan de hand zijn:

1. De node is down.
2. De node is nog up, maar alle heartbeat paden zijn verbroken.

Het interessant hieraan is dat niet eenduidig is te bepalen welke van deze twee situaties er aan de hand is. En, stel dat situatie 2 zich voordoet, dan vraagt de afgesneden node zich hetzelfde af van de rest van de cluster. Een dergelijke situatie noemen we een “split brain”. Iedere cluster moet een of ander mechanisme bevatten om te bepalen wat er aan de hand is. Dit mechanisme moet dan functioneren zonder dat de nodes met elkaar kunnen communiceren, en nodes moeten zichzelf uit het cluster kunnen nemen als wordt besloten dat dat nodig is om de consistentie te herstellen.

De voorkomende strategieën om dit op te lossen zijn:

1. Lock disk
   Veel clusteroplossingen (waaronder Sun Enterprise Cluster en HP MC/ServiceGuard) gebruiken een gedeelde disk in de cluster om dit probleem op te lossen. Zodra een mogelijk split brain wordt ontdekt racen de nodes om wie het eerst de lock disk heeft gereserveerd. De winnaar van die reservering mag door, en de andere node wordt geacht zich uit de cluster terug te trekken. Veelal wordt gebruik gemaakt van het SCSI Reserve/Release mechanisme om de reservering te plaatsen. MCSG gebruikt echter een ander algoritme waabij de lock disk wordt gebruikt om een soort semafoor te implementeren.

   De verliezende nodes worden geacht zich uit het cluster terug te trekken. Meestal gebeurt dit door een panic te forceren. Het systeem geeft dan namelijk met gezwinde spoed al zijn resources op. Sun Enterprise Cluster doet geen panic maar geeft via een versnelde procedure alle gedeelde resources (file systemen en dergelijke) vrij. Dit gaat helaas niet in alle gevallen goed! SteelEye's LifeKeeper gebruikt sowieso SCSI reserveringen voor alle gedeelde schijven. Daar het met name om de gedeelde schijven gaat werkt deze oplossing altijd goed. Helaas heeft SteelEye heftig in de Linux SCSI mid-tier drivers in moeten grijpen om Linux op een beetje sociale wijze met reserveringen om te laten gaan. Deze wijzigingen zitten onder andere in Red Hat Linux 6.2 (enterprise edition).

2. STONITH (Shoot The Other Node In The Head)
   Een ander mechanisme wat je nogal eens tegenkomt in Unix clusters is als cluster nodes in staat zijn om andere cluster nodes uit te schakelen. We komen dit hoofdzakelijk tegen bij Heartbeat (eenvoudige Linux clustering), GFS (het Linux Global File System) en Sun Enterprise cluster (bij meer dan 2 nodes). Het idee is dat een cluster node op n of andere manier een andere node tot stoppen kan dwingen. In de Linux wereld is er een heuse STONITH API waarbij clusters via een gemeenschappelijke API andere nodes op configureerbare manieren "plat" kunnen leggen.

   De (in de Linux wereld) meest voorkomende manier van STONITH is via een "network power switch". Dit is een voeding annex UPS die over het netwerk kan worden benaderd en waarbij via een eenvoudig (TELNET gebaseerd) protocol de spanning op een bepaalde poort kan worden onderbroken. Indien zich een "split brain" voordoet racen de nodes naar de power switch en proberen ze elkaar uit te schakelen. Het moge duidelijk zijn dat slechts n node daarin kan slagen[3]. Indien het "split brain" wordt veroorzaakt doordat n van de nodes plat ligt dan wint de overgebleven node dit natuurlijk altijd. Interessant genoeg is bij een dergelijke manier van STONITH de voeding in kwestie een "Single Point Of Failure". Maar dit terzijde, je kunt immers niet alles hebben.

   Sun Enterprise Cluster hanteert een andere manier van STONITH. In een Sun cluster zitten de console poorten van alle nodes verbonden met een ANNEX terminal server. In het geval van een "split brain" proberen de masters in de cluster partities de andere nodes te resetten door met TELNET naar het console van de plat te leggen host te gaan en daar op "Stop-A" te drukken. In tegenstelling tot hierboven is in een Sun cluster de ANNEX terminal server geen "Single Point Of Failure". Het bewijs hiervan wordt als oefening aan de lezer gelaten.

[3] Voor de liefhebbers is hier een interessante "race" conditie te vinden…

4.1.3 Watchdog

In een interessant randgeval betreffende cluster lidmaatschap heeft te maken met wat er dient te gebeuren indien de cluster software zelf niet goed meer functioneert. Met name indien de centrale processen (daemons) uitvallen die het cluster lidmaatschap bewaken "weet" een node niet meer of hij nog lid is van de cluster of niet. In dat geval is er maar één ding te doen: zo snel mogelijk het cluster verlaten.

Alle cluster pakketten lossen dit op een soortgelijke manier op: ze verpakken in de kernel een driver die als enige doel heeft het systeem te resetten (panic). Echter, we kunnen de driver voor luttele seconden van zijn snode plan afhouden indien we het een koekje geven. De driver peuzelt dan het koekje op, en als het op is wordt het systeem gereset, behalve als we het dan weer een koekje hebben gegeven[4]. De cluster software (en dan meestal de centrale cluster lidmaatschap service) activeert deze driver (de "failfast" driver) en zorgt ervoor dat er met een vaste regelmaat een koekje wordt gegeven. Als de software onverwacht stopt (bug, "kill") of door performance problemen een tijdje niet meer aan de bak komt, dan reset de failfast driver het systeem, en wordt het cluster dus via de snelle weg verlaten.

In Linux is de failfast faciliteit een standaard onderdeel van de Linux kernel. Naast een software faciliteit om dit te doen (de "softdog") worden er ook diverse hardware watchdogs ondersteund die het systeem met een rechtstreekse bus reset tot nieuwe inzichten proberen te brengen.

4.2 Disks, volumes, file systems

Met betrekking tot de hogere beschikbaarheid van disk gebieden zoals file systems en swap spaces vertrouwen alle mij bekende cluster oplossingen op los functionerende volume manager en/of disk mirroring software. Het is de taak van die software om de disk gebieden die essentieel zijn voor het functioneren van het systeem en de verhoogd beschikbare applicaties te beschermen tegen uitval van schijven, controllers en kabels. Meestal functioneert deze HA disk oplossing volledig los van de failover gedeeltes van de cluster software. Wel kan de cluster software de disk management software besturen teneinde deze ervan te overtuigen dat schijven (disk groepen) nu op de ene, danwel de andere, node ter beschikking moeten worden gesteld.

Een aantal Unix leveranciers heeft het in de goedheid van het hart gevonden om met het basis besturingssysteem een beetje ordentelijke disk management software mee te leveren. Denk hierbij aan AIX (Logical Volume Manager), HP-UX (alhoewel mirroring een separaat aan te schaffen feature is (MirrorDisk/UX)) en Linux (LVM voor partitionering en MD voor mirroring en RAID-5). Sun Enterprise cluster kan "het" zowel met Solstice Disk Suite (SDS), of met de separaat te verkrijgen Veritas Volume Manager (ook wel Sun Enterprise Volume Manager en Sun StorEdge Volume Manager geheten). Bij Sun clusters gaat mijn absolute voorkeur uit naar het werken van de Veritas Volume Manager, maar dit terzijde.

Naast mirroring en RAID-5 achtige functies ligt het meestal ook in de functieomschrijving van de volume manager software dat het bijhoudt welke disks (disk groepen) momenteel door welk systeem worden gebruikt. Sommige software is daar beter in

[4] Deze metafoor is met name bedoeld voor jonge vaders.

dan andere. Met name de Linux Logical Volume Manager is nog niet zo sterk in het gebruik ervan in clusters. Andere software (MD, SDS) ondersteunt dit concept totaal niet en laat het aan de cluster software en de beheerder over om ervoor te zorgen dat bestanden niet vanaf twee of meer nodes tegelijk worden gebruikt. SteelEye's LifeKeeper gooit om die reden SCSI RESERVE's in de strijd om op het laagst denkbare niveau ervoor te zorgen dat schijven uniek voor één node gereserveerd zijn. HP-UX's LVM kent expliciete integratie met cluster software (de "-c" optie op diverse LVM commando's) om de toekenning van volume groups aan nodes door de cluster software te laten synchroniseren.

4.2.1 Journaling file systems

Om in geval van een crash en overname van applicaties door een andere node de integriteit van de betrokken file systemen zo snel mogelijk weer op orde te krijgen verdient het aanbeveling om een Journaling File System (JFS) toe te passen. Alle mainstream Unix leveranciers ondersteunen momenteel in hun basis besturingssysteem een of ander JFS: AIX (IBM's eigen JFS), HP-UX (Veritas JFS (VXFS), alhoewel de volledige functionaliteit wederom het aanspreken van een apart spaarvarken vereist), IRIX (SGI's eigen XFS), Solaris (UFS met logging, ook VXFS als optie (extra fondsenwerving nodig)) en Linux (Reiser FS, IBM JFS (beta), XFS (versie 1.0 net uit), ext3 (beta)).

4.2.2 Cluster file systems

Een hele stoere ontwikkeling van de laatste tijd is het verschijnen van zogenaamde cluster file systemen. Dit is file system technologie die het mogelijk maakt om een file system op een gedeelde hard disk set vanaf meer dan één node tegelijk te benaderen. Via een snel intra cluster netwerk wordt de toegang tot het gedeelde file system (locks, cache) gesynchroniseerd. Veritas heeft een dergelijk cluster file system en in de open source wereld doet het Global File System (http://www.globalfilesystem.org) opgang. Cluster file systems zijn meestal tevens Journaling File Systems, en in het geval van een node crash draagt één van de nog levende nodes zorg voor het terugdraaien van de openstaande file system transacties. Dientengevolge is het eigenlijk nooit nodig om een file system check te draaien.

Bedenk echter wel dat het met een cluster file system niet zonder meer mogelijk is om een applicatie op meerdere nodes tegelijk te draaien. De meeste applicaties ondersteunen het namelijk niet dat meerdere instances tegelijk in de data set van de applicatie aan het werk zijn!! Cluster file systems kunnen echter wel handig worden gebruikt voor read mostly data zoals programmatuur en configuratie files (voorkomt synchronisatie en versieprobleem).

4.3 Standby networking

De meeste clusteroplossingen streven ernaar om in geval van problemen de zaak lokaal op te lossen, zonder een failover te hoeven doen. Een failover impliceert namelijk ongeplande downtime (al is het maar een paar minuten). Door voor voorkomende problemen een "local recovery" te plegen kan zelfs die minimale downtime worden

vermeden. Het in de vorige sectie toegelichte gebruik van mirroring en RAID-5 valt bijvoorbeeld ook onder “local recovery”.

Een ander voorbeeld van dit concept is het gebruik van meerdere netwerkkaarten om naar hetzelfde netwerk te verbinden. Mocht een netwerkkaart (of kabel, of hub) uitvallen dan is de clusteroplossing in staat om alle verkeer naadloos naar de andere kaart te leiden (veelal door het MAC adres en/of de IP adressen naar de standby kaart over te hevelen). Deze feature gaat er overigens wel van uit dat het netwerk verder “goed” (zonder Single Point of Failure) in elkaar zit.

In een Sun Enterprise Cluster (SEC) gaat deze feature door het leven als “Public Network Management”. Andere clusteroplossingen hebben niet noodzakelijkerwijs een aparte naam voor dit verschijnsel. Merk trouwens op dat in een SEC altijd twee aparte netwerken nodig zijn als heartbeat/private netwerk; PNM wordt niet gebruikt om tussen deze netwerken te schakelen. In Linux clusters is standby networking meestal niet of nauwelijks geïmplementeerd daar er geen standaard manier is om status informatie over een netwerkadapter vanuit de driver naar een userland applicatie te krijgen. In HP-UX MC/ServiceGuard clusters is voor standby networking nog een aparte netwerk bridge nodig die tussen het standby netwerk en het data (public) netwerk staat opgesteld.

4.4 Resources / applicaties

Uiteindelijk is het de functie van de cluster om in geval van problemen applicaties van node naar node te laten verhuizen. Om zo’n applicatie te laten functioneren zijn meestal ook nog wat andere hulpbronnen nodig die dan ook mee moeten verhuizen, denk hierbij hoofdzakelijk aan IP adressen en file systemen. Het is de taak van de cluster software om de applicatie en de hulpbronnen te deactiveren op het systeem waar het geheel vandaan komt en te activeren op het doelsysteem. Om dit te kunnen bewerkstelligen moet de software weten hoe resources en applicaties kunnen worden gestuurd (activeren, deactiveren, controleren). Daarnaast moet het natuurlijk voor een klant mogelijk zijn om hier uitbreidingen op te plegen. Het theoretisch model hiervoor verschilt enorm van clusteroplossing tot clusteroplossing.

Verreweg het eenvoudigste model wordt geboden door HP MC/ServiceGuard (SG). SG kent het concept van het “package”, waarbinnen één of meer IP adressen, één of meer volume groups en één of meer file systems (in die volume groups) liggen. Tevens kent een package een start en een stop functie die door de SG implementator dient te worden ingevuld. Indien een package naar een ander systeem verhuist worden de geconfigureerde IP adressen, volume groups en file systemen meegenomen en worden de stop/start functies aangeroepen. Het is aan de SG implementator om ervoor te zorgen dat in de stop/start functies alles gebeurt wat nodig is om de applicatie (voor afname van, danwel beschikbaarstelling van, IP adressen en file systemen) te stoppen of te starten. Al deze functies worden uitgevoerd door het SG package control script, wat door SG voor een package wordt gegenereerd en wat alleen nog maar op kritieke punten hoeft te worden ingevuld. Het voordeel van deze aanpak is de enorme simpelheid van het model. Het grootste nadeel de starheid ervan, het is niet eenvoudig mogelijk om andere soorten resources elegant mee te schakelen.

Bij Sun Enterprise Cluster worden IP adressen, disk groepen en file systemen ingekapseld in een fenomeen genaamd de “logical host”. De logical host is een soort virtuele computer die de resources inpakt die nodig zijn voor een applicatie om te

draaien. Applicaties heten hier “data services” en worden gebruikt om instances van applicaties te stoppen en te starten zodra logical hosts door het cluster verhuizen. De cluster bewerkstelligt dit door op gezette momenten tijdens het logical host migratieproces callback scripts van de data service aan te roepen die de functie hebben om instances van de applicatie te stoppen en te starten. Een complicerende factor wordt gevormd door het feit dat de data service scripts normaal gesproken op alle nodes van de cluster worden aangeroepen, ook als er op die node niets hoeft te gebeuren; het is aan de data service scripts om uit te zoeken of er iets dient te gebeuren of niet! Dit brengt natuurlijk ook een bepaalde flexibiliteit met zich mee, omdat een data service zich hierdoor makkelijker over verschillende nodes kan uitstrekken, wat bijvoorbeeld met SG niet eenvoudig mogelijk is. Hierdoor wordt het schrijven van goede data service scripts echter wel redelijk complex, en het vereist een doorgewinterde shell script of Perl programmeur om dit voor elkaar te krijgen. In Sun Cluster 3.0 is dit feature trouwens vervangen door een GUI gedreven wizard die de scripts genereert. Het is de vooruitgang moeten we maar denken, je houdt het niet tegen...

Een aantal andere oplossingen (SteelEye LifeKeeper, SGI/SuSE FailSafe, Veritas Cluster Server en anderen) houden er een heel erg mooi hiërarchisch model op na waarin resources en applicaties (een applicatie is in dat model trouwens ook een resource) in een afhankelijkheidsboom aan elkaar worden gekoppeld. In zo’n boom kan dan worden aangegeven dat een applicatie afhankelijk is van een IP adres, en dat een IP adres afhankelijk is van een netwerkkaart; dat die applicatie ook afhankelijk is van een file systeem, en dat dat file systeem afhankelijk is van een volume, die weer afhankelijk is van een disk group, die weer ... et cetera (applicaties kunnen daarmee zelfs afhankelijk zijn van andere applicaties). Zodra een node in de boom faalt (vereist monitoring, straks hierover meer) wordt die hele boom (inclusief afhankelijke en super nodes) gedeactiveerd en op een andere node in het cluster geactiveerd. De cluster kent van iedere node in de boom het resource type en voor ieder type kent het de stop/start methode (script). Meestal is de set van resource types eenvoudig uitbreidbaar via een cluster API of iets dergelijks.

Het laatst genoemde hiërarchische model is, alhoewel het meest complex in configuratie en uitbreiding, verreweg het fraaist.

4.5 Monitoring

Een wezenlijk onderdeel van ieder cluster wordt gevormd door componenten die monitoren of een applicatie of resource nog wel functioneert. “Node down” is namelijk niet het enige probleem waar we mee te maken hebben, indien een applicatie om andere redenen niet beschikbaar is op een node (b.v. semaforen tabel vol, geen swap space meer, maximaal aantal open files bereikt) willen we ook een local recovery of failover bewerkstelligen. Om dit te kunnen waarnemen dient de cluster de beschikbaarheid van de applicatie en de resources te monitoren.

Monitoring is in de meeste gevallen een ondergeschoven kindje. In lang niet alle (zeg maar: de meeste) gevallen wordt er niet, of niet adequaat, gemonitored. Dit laatste geldt met name voor eigengebakken applicaties of resources. Alle clusteroplossingen bieden hiervoor mogelijkheden, die echter altijd optioneel zijn of zonder problemen “leeg” kunnen worden gelaten. Dit is sneu, maar dat terzijde...

Vrijwel alle clusters ondersteunen “local probes”. Dit zijn monitors die op hetzelfde

systeem draaien als de te monitoren resource/applicatie, en wiens functie het is om op wat voor manier dan ook vast te stellen dat alles nog steeds “hunky dory” is. Die local probes doen meestal één of meer van de volgende dingen:

1. Processen controleren
2. In log files kijken
3. Status commando’s uitvoeren

Zodra wordt vastgesteld dat een en ander niet meer functioneert zoals het hoort kan dit via een API (of een return code) worden gecommuniceerd met het cluster framework die dan de nodige herstelacties kan uitvoeren. De local probe kan ook trachten de situatie zelf te herstellen, maar zal dan een teller moeten bijhouden die hem stuurt om ingeval die lokale herstelactie niet slaagt (of te vaak vaut blijft gaan) alsnog een grovere herstelactie (failover) door het cluster uit te laten voeren.

Sun Enterprise Cluster kent (voor zover ik weet als enige, maar ook (met name) ik ben niet alwetend) het concept van de “remote probe”. Dit is een fault monitor die op een andere node in het cluster draait en die tracht over het public network de gezondheid van de applicatie te meten. Meestal gebeurt dit door over het netwerk een verbinding met de applicatie te maken en dan een paar maatgevende applicatie transacties uit te voeren (b.v. in geval van Oracle het aanmaken, vullen, ondervragen en verwijderen van een database tabel). Mocht deze controle falen dan verzoekt de remote probe om een failover. Dergelijke “end to end” controles zouden wat mij betreft veel vaker plaats mogen vinden.

4.6 GUI

Een ziekte van de moderne tijd waaraan ook clusters zich nauwelijks kunnen onttrekken is het grafische gebruikersinterface. Een beetje beheerder kan hier natuurlijk niks mee, maar het verkoopt zo lekker handig. Om die reden zijn alle clusters vandaag aan de dag voorzien van een grafische schil waarin de status van de cluster wordt weergegeven en bepaalde clusterhandelingen (stoppen, starten, failover, configuratie) plaats kunnen vinden. Daarnaast is het in alle gevallen ook nog mogelijk om de cluster met de hand te bedienen.

Cluster GUI’s lopen uiteen van volledig, overzichtelijk en functioneel (SteelEye LifeKeeper, SGI/SuSE FailSafe, Veritas Cluster Server) tot overbodig en nutteloos (Sun Enterprise Cluster, maar zou beter zijn in het volgende release).

5 If you’ve seen one, you’ve seen ’m all

Unix clusters zijn in zeer hoge mate identiek qua opbouw en functionaliteit. Op onderdelen verschillen ze enorm, maar van een iets hoger plan af gezien is de uniformiteit opvallende dan de verschillen. Het is met alle hier genoemde oplossingen (en ook met de niet zo frequent genoemde) mogelijk om een hoog beschikbare omgeving voor een applicatie te bouwen. De keuze voor de cluster technologie wordt meestal ingegeven

door reeds gemaakte keuzes voor een specifiek Unix operating systeem c.q. hardware platform.