Avdelning för datavetenskap

Erik Andersson och Marcus Larsson

Nätverkslagring: SAN/NAS-lösning för VM-miljö

Network Storage: SAN/NAS solution for VM environment

Examensarbete (15hp)
Datavetenskap

Datum/Termin: 2010-06-08
Handledare: Kerstin Andersson
Examinator: Martin Blom
Ev. löpnummer: C2010:12

Karlstads universitet 651 88 Karlstad
Tfn 054-700 10 00 Fax 054-700 14 60
Information@kau.se www.kau.se

Nätverkslagring: SAN/NAS-lösning för VM-miljö

Erik Andersson & Marcus Larsson

Denna rapport är skriven som en del av det arbete som krävs för att erhålla en kandidatexamen i datavetenskap. Allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och inget material är inkluderat som tidigare använts för erhållande av annan examen.

Erik Andersson & Marcus Larsson

Godkänd, 2010-06-08

Handledare: Kersin Andersson

Examinator: Martin Blom

Sammanfattning

Nätverkslagring är nu för tiden en populär metod för att lagra data. Därför väljer många företag att använda nätverkslagring för att centralisera sin lagring för att få en effektivare lagring där flera användare har tillgång till samma data.

Denna uppsats behandlar detta ämne och beskriver ett arbete som gjordes på uppdrag av Compare Testlab. Uppdraget handlade om att förbättra deras nätverkslagring. Denna nätverkslagring användes först och främst av deras virtualiseringsplattform, XenServer. Arbetet utfördes i tre olika steg; utredning, implementation och dokumentation.

Utredingen gick ut på att undersöka vilka lösningsalternativ och vilka operativsystem som passade bäst till lösningen. Implementationen bestod av implementation av den lösning som utredningen ledde till och dokumentationen består av en konfigurationsmanual som Compare Testlab kommer att få.

De punkter som skulle tas upp i utredningen var driftsäkerhet, pålitlighet, prestanda, skalbarhet och användarvänlighet. Utifrån dessa punkter togs en lösning fram med två servrar där en agerar primär och den andra som en redundant sekundär server. Om den primära servern skulle gå ner så tar den sekundära över. Lösningen skiljde också lagringsnätet från det huvudnät som finns på Compare Testlab. Detta för att göra lösningen så snabb som möjlig.

Network Storage: SAN/NAS solution for VM environment

Network storage is nowadays a popular method for storing data. Therefore, many companies opt to use networked storage to centralize their storage in order to obtain a more efficient storage where multiple users have access to the same data.

This paper deals with this topic and describes a work that was commissioned by Compare Testlab. The mission focused on improving their network storage. The network storage were used primary by their virtualization platform, XenServer .The work was carried out in three steps: inquiry, implementation and documentation.

The investigation was to explore which solution alternatives and which operating systems that were best suited for the solution. The implementation consisted of the implementation of the solution that the investigation had led to and the documentation consists of a configuration manual that Compare Testlab will receive.

The points that were included in the inquiry were dependability, reliability, performance, scalability and ease of use. From these points one solution were presented which consisted of two servers where one is acting as primary and the other is acting as a redundant secondary server. If the primary server goes down the secondary server will take over. The solution also splits the storage network from the main network that Compare Testlab uses. This is for making the solution as fast as possible.

Innehåll

Figurer

1 Inledning

Idag är det allt vanligare att företag och även privatpersoner använder sig av en centraliserad lagring för att förvara data. Detta för att effektivisera lagringen och göra den tillgänglig för fler personer samtidigt. Vanliga lösningar för nätverkslagringar är SAN (Storage Area Network, se Sektion 2.3.2) och NAS (Network Attached Storage, se Sektion 2.3.1).

Denna uppsats ska behandla nätverkslagring och beskriva ett arbete som gick ut på att förbättra Compare Testlabs (se Sektion 2.2) nätverkslagring. Denna nätverkslösningen är först och främst till för att ge lagring för deras virtuella plattform XenServer (se Sektion 2.5). Den lösning som fanns hade många brister. Därför skulle en utredning göras på områdena; driftsäkerhet, pålitlighet, prestanda, skalbarhet och användarvänlighet. Ett av de största problemen var att den nätverkslagring som var i bruk bestod av endast en enda server. Skulle den ha slagits ut skulle många servrar som var beroende av nätverkslagringen inte ha fungerat längre. För att förbättra detta kommer den här uppsatsen bland annat att behandla hur en lagringsserver till kopplas in. Denna ska agera backup om den andra slås ut. Detta att en dator tar över från en annan dator, som tagits ut ur drift, kallas för failover (Sektion 2.7).

Arbetet på Compare Testlab utfördes i tre olika steg nämligen utredning, implementation och dokumentation.

- **Utredning**

  Utredningen gick ut på att söka efter information om ämnet. Frågor som krävde svar var vilket operativsystem som skulle fungera bäst och vilka ändringar hos den nuvarande lösningen som skulle behöva göras.

- **Implementation**

  När utredningen var gjord och ett operativsystem hade valts ut började implementationen. Under implementationsdelen utfördes alla installationer och konfigurationer till den nya lagringsmiljön.

- **Dokumentation**

  Dokumentationen är en guide på hur man konfigurerar upp failover mellan två lagringsenheter. Dokumentationen ska Compare Testlab använda vid driftsättningen av den nya lösningen.

Uppsatsen är strukturerad i sex olika kapitel:

- Kapitel 1 är en introduktion till arbetet som ska utföras.
- Kapitel 2 behandlar bakgrunden. Detta kapitel beskriver all den information som behövs för arbetet som ska utföras kring nätverkslagringen.
- Kapitel 3 består av utredningen. Denna utredning tar upp information om olika möjliga lösningar och vilka operativsystem som kan användas.
- Kapitel 4 tar upp den implementation som utförs. I utredningen föreslås olika lösningar och en lösning väljs ut för att implementeras i en testmiljö.
- Kapitel 5 tar upp utvärdering och resultat och beskriver några tester och resultaten från dessa. Detta kapitel tar också upp några problem som uppstod under projektets gång och möjliga lösningar till dessa.
- Kapitel 6, som är det sista kapitlet, utvärderar själva arbetet och beskriver alternativa lösningar och eventuella framtida påbyggnader.

2 Bakgrund

Detta projekt utfördes på uppdrag av Compare Testlab och gick ut på att utveckla en ny lösning för deras nätverkslagring. I detta kapitel beskrivs termer och annan information som används inom detta ämne och som har koppling till projektet. Sektion 2.1 innehåller en kort bakgrund om projektet. Sektion 2.2 beskriver företaget Compare och avdelningen ute på Hammarö, Compare Testlab. Sektion 2.3 introducerar konceptet nätverkslagring och några olika arkitekturer som går att implementera inom detta område. Protokoll som dessa arkitekturer använder beskrivs i Sektion 2.4. I nästa sektion, 2.5, beskrivs virtualisering. Open source-mjukvara till nätverkslagring beskrivs i Sektion 2.6. Sektion 2.7 beskriver failover och high availabilty-lösningar. Sektion 2.8 beskriver hårdvara som vanligtvis används till nätverkslagring. Kapitlet avslutas med en sammanfattning.

2.1 Inledning

Projektets huvuddelar gick ut på att utreda och ta fram en ny nätverkslagringslösning för företaget Compare Testlabs virtuella miljöer. För att få en väl fungerande lösning skulle en utredning göras på följande områden:

- **Driftsäkerhet**

  Skulle ett eventuellt haveri uppstå i det ursprungliga systemet så blir det stopp för samtliga miljöer. Den nya servermiljön ska alltså kunna undvika dessa driftstopp med en failover-funktion mellan två enheter. Man använder två servrar som agerar som primär respektive sekundär. Skulle en dator slås ut så tas den andra i bruk.

- **Pålitlighet**

  All data som lagras ska kunna vara lätt tillgänglig och vara i ett säkert förvar. För att kunna höja säkerheten i lagringen kan man använda sig av RAID (Redundant Array of Independent Disks) (se Sektion 2.8.2), som är en teknik för att spegla eller öka hastigheten hos en samling hårddiskar.

- **Prestanda**

  Dataöverföringen kan lätt bli en flaskhals i en stor servermiljö. Därför ska lagringsnätverket vara skilt från all annan trafik för att inte denna trafik ska störa nätverkslagringstrafiken och sänka dess hastighet.

- **Skalbarhet**

  Med stora datalagringar så krävs det att man kan underhålla och eventuellt bygga ut systemet utan driftstopp.

- **Användarvänlighet**

  Att kunna administrera servrarna är viktigt. För detta krävs ett gränssnitt som är kraftfullt och begripligt.

2.2 Compare Testlab

Den stiftelse som detta arbete gjordes för, Compare Karlstad, bildades år 2000 och står för Competence Area Karlstad. Compare är en stiftelse som hjälper IT-företag att samarbeta. Detta gör de genom ett antal projekt där de hjälper olika företag att växa och arbeta tillsammans mot kompetens- och affärsutveckling. I organisationen finns det runt 100 företag med 2500 ingenjörer och tekniker.

Den del av Compare som själva projektet gjordes för kallas Compare Testlab och är en satsning av Compare inom ICT-branschen tillsammans med Hammarö kommun och Karlstads universitet, som stöds av bland andra Region Värmland och Länsstyrelsen. De tillhandahåller resurser i form av datorhallar, där företag kan hyra in sig och utföra tester av bland annat affärs- och produktkvalitet [3].

2.3 Nätverkslagring

Nu för tiden ställs det mer och mer krav på att all data ska vara samlat på en och samma plats och att man ska kunna komma åt data var man än befinner sig. Allt fler företag väljer

därför att centralisera sin data i en så kallad nätverkslagring. Nätverkslagring betyder att en viss lagringsenhet förvarar all data för alla användare i ett nätverk. Detta betyder då att en användare kan lagra något på denna lagringsenhet, byta dator och ändå ha tillgång till de filer som användaren har lagrat. Detta projekt handlar huvudsaklingen om nätverkslagring och två olika metoder för nätverkslagring, NAS (Network Attached Storage) och SAN (Storage Area Network) som beskrivs i nedanstående sektioner [18].

2.3.1 NAS - Network Attached Storage

NAS är en nätverkslagring med speciella typer av datorer kallade NAS-enheter. Dessa enheter har ofta endast som uppgift att lagra data åt andra enheter på nätverket. En NAS-enhet konfigureras vanligtvis genom en webbläsare. Detta gör att NAS-enheter oftast inte har några I/O-enheter såsom mus eller tangentbord. Operativsystemen som körs på NAS-enheter är enkla med endast de mest nödvändiga funktionerna. För att förebygga dataförluster så är NAS-system vanligtvis uppdelade över flera redundanta logiska diskar eller över RAID-system (se Sektion 2.8.2).

Användare får tillgång till filer genom att fråga NAS-enheter om delar av abstrakta filer som sedan förs över med hjälp av protokollen NFS eller SMB/CIFS (se Sektion 2.4). NAS-enheterna tar hand om all filhantering själv och detta gör att det tydligt syns att lagringen i ett NAS-system är avlägset.

Förespråkare för NAS-system brukar säga att NAS har dessa fördelar jämfört med vanliga filservrar:

- **Billigare**

  En NAS innehåller endast de nödvändigaste komponenterna som en lagringsenhet behöver. Oftast är NAS-enheter också billigare i drift då dessa förbrukar mindre ström.

- **Lättare att administrera**

  Det gränssnitt som kommer med en NAS-enhet är oftast lätt att förstå. Konfigureringen brukar endast ta några minuter att utföra innan man kan använda NAS-enheten.

- **Mindre nertid**

  NAS-enheter är byggda med komponenter som är anpassade för varandra. Detta gör att stabiliteten är högre än hos vanliga datorer. Bättre stabilitet leder till mindre systemkrascher och därmed mindre nertid.

- **Bättre säkerhet**

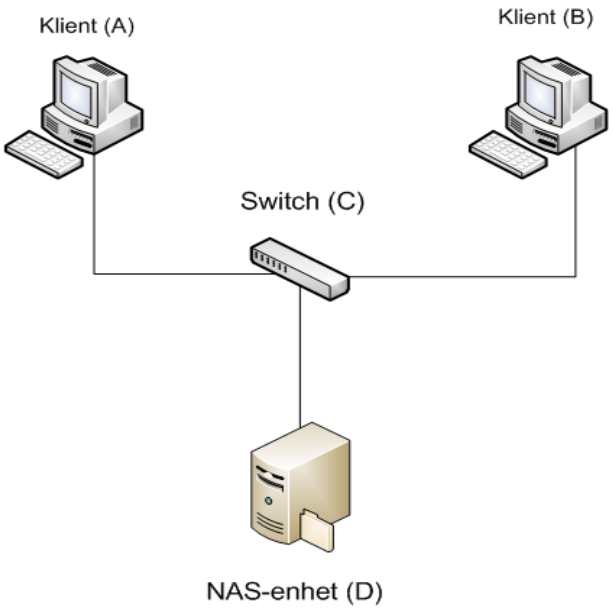
  Med RAID (Sektion 2.8.2), som NAS-enheterna ofta kommer med, kan man sätta upp diskarna med spegling. Skulle en disk gå sönder så har man fortfarande kvar all data hos en annan disk [22].

En bild på ett typiskt NAS följer i Figur 2.1, där det finns två datorer och en NAS-enhet som är kopplad till en switch. NAS-enheten (D i bilden) innehåller data som klientdatorerna A och B ska kunna komma åt. Switchen C gör så att dator A och B kan komma åt samma data som ligger hos NAS-enheten.

2.3.2 SAN - Storage Area Network

SAN är en arkitektur för datalagring som gör det möjligt för flera servrar att dela på ett gemensamt lagringsutrymme. SAN är det nätverk som binder ihop servrarna med lagringsutrymmet. Till skillnad från NAS, så ses diskarna som lokala på de olika datorerna som är anslutna till nätverket. Eftersom SAN inte utför någon filhantering, utan bara hanterar diskblock, så sker detta på de anslutna datorerna i nätveket. De protokoll som används i SAN-arkitekturen kan vara SCSI, Fibre Channel, ATA over Ethernet eller iSCSI (se Sektion 2.4) [24].

Ett typiskt SAN består oftast av tre olika lager (se Figur 2.2):



Figur 2.1: Network Attached Storage

- **Host layer**

  På host-lagret ligger servrar och de komponenter som gör det möjligt för dessa servrar att kommunicera med fabric layer. Servrarna kör de högnivåapplikationer som använder själva lagringen [24].

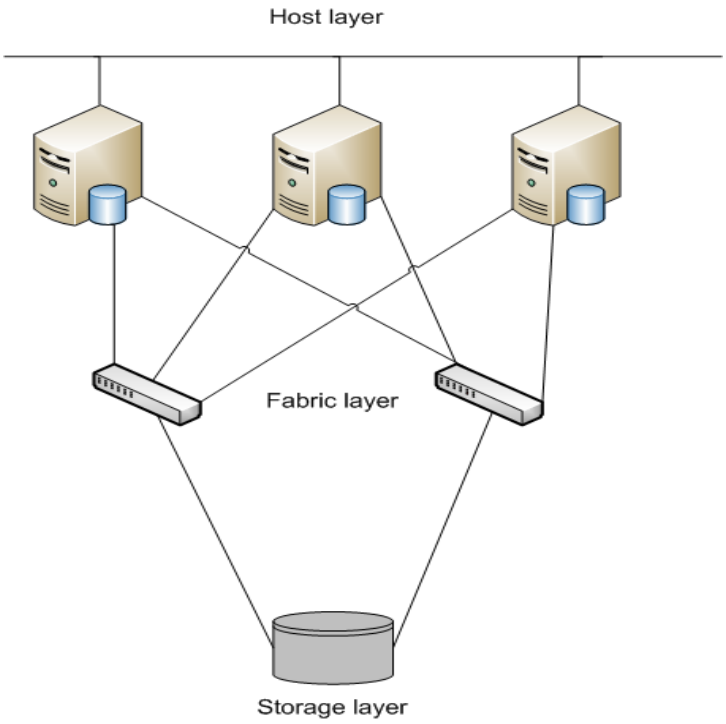
- **Fabric layer**

  Själva nätverksdelen i ett SAN. Det här lagret kallas även för SAN-lagret på grund av att det är detta lager som binder ihop host-lagret med nästa lager, storage-lagret [24].

- **Storage layer**

  Lagret där all förvaring äger rum. Det är här all hårdvara som har hand om lagringen ligger, som till exempel diskarrayer eller tape drives [24].

På senare år har definitionen mellan SAN och NAS blivit allt mer oklar. Man brukar säga att det som skiljer SAN och NAS åt är att SAN använder protokoll som jobbar på diskblocknivå medan NAS använder protokoll som jobbar på filnivå [22]. Dessa olika protokoll kommer att beskrivas mer ingående i nästa sektion.



Figur 2.2: Storage Area Network

2.4 Protokoll

I detta avsnitt kommer de protokoll som används inom nätverkslagringlösningar och även inom detta arbete att förklaras.

- **NFS - Network File System**

  NFS är ett protokoll utvecklat av Sun Microsystems. Dess syfte är att skapa en filbaserad tillgång till Unix-servrar över IP-nätverk. Detta görs genom ett server/klient-gränssnitt där den ena sidan exporterar en fil och den andra sidan importerar filen. NFS bygger på protokollet RPC (Remote Procedure Call) [27]. Detta protokoll är mest förknippat med NAS [30].

- **SMB/CIFS - Server Message Block/Common Internet File System**

  SMB och CIFS är protokoll som gör det möjligt att skapa filbaserad access till Windows-servrar över IP. SMB/CIFS är som NFS uppbyggt i en server/klient-arkitek-

tur där klienten frågar efter resurser och servern ger ut resurser [10].

- **FCP - Fibre Channel Protocol**

  FCP är ett transportprotokoll (liknande TCP) som först och främst används i nätverkslagring. Dess huvudsakliga funktion är att transportera SCSI-instruktioner över fibernätverk [28].

- **SCSI - Small Computer System Interface**

  SCSI är ett gränssnitt för anslutning och överföring mellan hårddiskar och datorer. Överföring kan även ske över ett nätverk med till exempel FCP och iSCSI [24].

- **iSCSI - internet Small Computer System Interface**

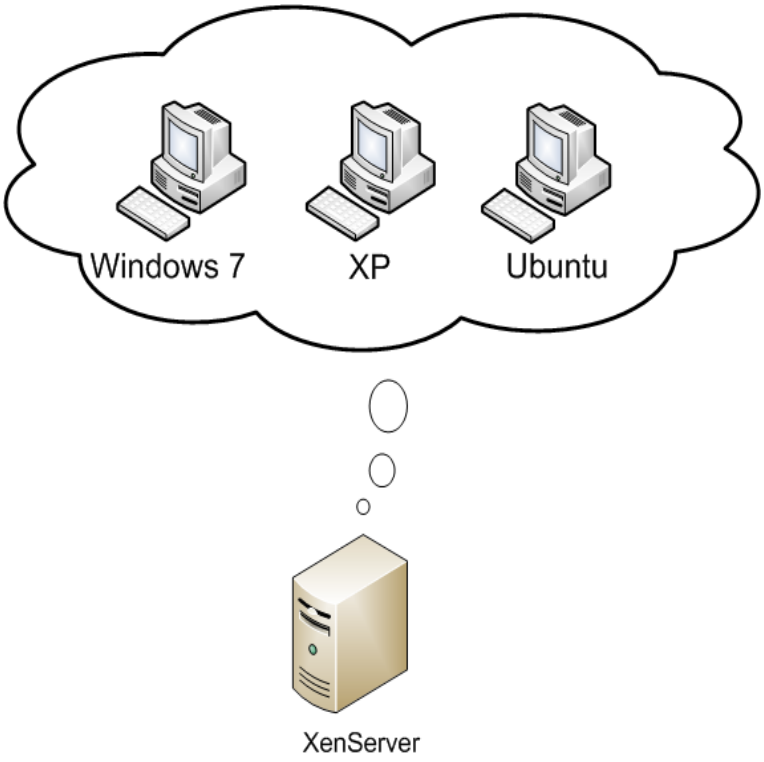
  Om ett enkelt och mindre kostsamt SAN önskas, kan man använda iSCSI. Detta protokoll skickar vanliga SCSI-meddelanden över ett vanligt Ethernet. Istället för att köpa dyra utrustningar till ett fibernätverk, kan man använda sig utav vanlig Ethernet med hjälp av iSCSI [24].

- **AoE - ATA over Ethernet**

  AoE är ett protokoll med hög prestanda för att komma åt SATA-lagring över Ethernet. Det kan användas för att bygga billiga SAN-lösningar. AoE använder inget lager ovanför Ethernet, såsom TCP eller IP [2].

2.5 Virtualisering

Servervirtualisering är något som blivit mycket populärt det senaste decenniumet. Med virtualisering så fördelas datorkraften mellan olika operativsystem på samma server. En server idag är så pass kraftfull att en massa beräkningskraft ofta blir oanvänd. Istället för att köpa flera servrar för att utföra olika uppgifter, behöver man endast använda en server med hjälp av virtualisering (se Figur 2.3) [1]. Compares Testlabs lagring ska användas till deras virtuella servrar som körs genom en server med XenServer, som är en plattform för



Figur 2.3: Virtualisering

servervirtualisering. Man installerar XenServer på en server för att sedan kunna komma åt denna server från datorer som har kontakt med den. Detta sker med hjälp av applikationen XenCenter, som är till Windows, eller OpenXenCenter, som är till Linux [8]. Genom XenCenter eller OpenXenCenter kan man då använda XenServer för att installera virtuella maskiner. De virtuella maskiner som körs på XenServern ska använda sig av den lagring som tas fram i denna lösning.

2.6 Operativsystem

För att kunna implementera en nätverkslagring så behövs ett operativsystem som är speciellt anpassat för detta. De operativsystem projektet kommer att använda är endast open source-produkter, det vill säga mjukvaror som är helt gratis att få tag i. Uppsatsen kommer att fokusera på de två operativsystemen Openfiler och FreeNAS.

Openfiler är ett operativsystem som används till nätverkslagringsservrar och har Linuxdistributionen rPath som grund. Gränssnittet är webbaserat och alla konfigurationer kan göras från en annan dator i samma nätverk [7]. Openfiler var det operativsystem som användes i den dåvarande lösningen hos Compare Testlab.

FreeNAS är ett operativsystem som bygger på FreeBSD. Likt Openfiler är det ett serveroperativsystem för nätverkslagring. Den stora skillnaden är att FreeNAS kommer med färre funktioner men är också mindre systemkrävande [4].

Dessa två operativsystem jämförs med varandra i Kapitel 3. Den bäst lämpade för den nya lagringslösningen väljs ut för att sedan implementeras i en testmiljö (se Kapitel 4).

2.7 Failover

Ett av de viktigaste kraven för detta arbete var att implementera en failover-funktion mellan två servrar. Failover kallas den funktion som går ut på att koppla över tjänster från en server till en annan redundant server om ett systemhaveri skulle ske. Nedan följer några termer som används inom failover.

- **Heartbeat**

  Hearbeat är en tjänst som skickar signaler eller pulser mellan huvudservern och den redundanta servern. Skulle signalen inte komma fram längre så kommer den redundanta servern att starta sina system och ta över huvudserverns tjänster (till exempel webbserver eller databasserver) [6].

- **DRBD - Distributed Replicated Block Device**

  DRBD är den tjänst som synkroniserar data mellan de två servrarna så att ingen data går förlorad om en av datorerna havererar [6].

- **HA - High Availability**

  High Availabilty är den term som används till att beskriva system som alltid är

tillgängliga. Detta kan appliceras på många olika system, till exempel flygplan, sjukhus och datorer. I denna uppsats används denna term hos servrar [6].

Implementation av failover-funktionen kommer att beskrivas ingående i Kapitel 4.

2.8 Hårdvara

Den hårdvara som NAS och SAN använder gås igenom i denna sektion.

2.8.1 Diskarrayer

En vanlig lagringsenhet i ett SAN är diskarrayer. Diskarrayer är lagringssystem bestående av ett antal diskar som är sammankopplade till ett stort lagringsutrymme. De vanligaste gränssnitten för hårddiskar som används för lagring är antingen SCSI (se Sektion 2.4) eller SATA (Seriell Advanced Technology Attachment). SATA är en seriell databuss som först och främst används till att förflytta data till och från hårddiskar [19].

Att använda diskarrayer är ett bra sätt att skydda informationen som är lagrad. Detta kan göras med bland annat RAID.

2.8.2 RAID - Redundant Array of Independent Disks

RAID är en teknologi för att få en pålitligare och effektivare lagring av data genom att ordna diskar i arrayer av redundans. Det finns tre olika koncept inom RAID:

- **Spegling**
  Spegling går ut på att identisk data skrivs till mer än en disk i en array. Man får på detta sätt en säker informationslagring eftersom data finns på mer än en plats.

- **Striping**
  Striping används för att sprida ut data på flera diskar. Detta påskyndar hämtningsprocessen eftersom när data hämtas från en disk så kan en annan disk leta efter nästa segment.

- **Felkorrigering**

  Felkorrigering fås genom att redundanta paritetsbitar lagras på disken. Dessa kan sedan användas för att återställa diskarna.

Dessa tre koncept används i olika grad beroende på vilken RAID-nivå som används. De grundläggande RAID-nivåerna som används mest idag är:

- **RAID 0**

  Denna nivå använder striping men ingen spegling eller felkorrigering. En diskarray med RAID 0 har hög hastighet på grund av striping, men har ingen hög tillgänglighet på grund av bristen på paritetsbitar som gör att data kan gå förlorad. Skulle en disk gå sönder blir övriga diskar obrukbara.

- **RAID 1**

  RAID 1 använder spegling men ingen striping. RAID 1 har ingen koll om data är korrumperad, vilket kan leda till virus. Om en disk får virus kan det spridas till de andra genom spegling. I och med att samma data lagras på flera diskar så blir läshastigheten högre, medan skrivhastigheten blir lägre eftersom all data måste skrivas till mer än en disk.

- **RAID 5**

  Denna nivå använder striping på blocknivå med paritetsdata utspritt över alla diskar. Med hjälp av paritetsdata så kan man återskapa en disk som gått sönder. Denna nivå används ofta i servrar.

- **RAID 10**

  Denna nivå kombinerar RAID 0 och RAID 1 och får då en diskarray med höga hastigheter samtidigt som man får hög tillgänglighet [24].

2.8.3 LVM - Logical Volume Manager

LVM är ett grundläggande sätt att hantera lagringssystem i Linux på ett skalbart sätt. Detta betyder att man kan ändra storleken på volymen under drift. Hårddiskarna i en LVM-pool är uppdelade i volymgrupper, som i sin tur är uppdelade i virtuella diskar som kallas för logiska volymer [5].

2.9 Sammanfattning

I detta kapitel har grunderna i nätverkslagring och bakgrunden till den lösning som ska tas fram presenterats. Detta inkluderar en genomgång av olika protokoll, operativsystem och hårdvara som används inom nätverkslagring. I nästa kapitel beskrivs själva utredningen.

3 Utredning

I detta kapitel ska olika nätverkslagringsmöjligheter som finns ute på marknaden utforskas. Eftersom kravet var att lösningen för Compare Testlab ska vara gjord på open source-produkter så fokuserades utredningen först och främst på de open source-alternativ som finns. Det gjordes även efterforskningar över hur mycket färdiga SAN-lösningar som finns på marknaden kostar och jämförelser mellan olika produkter utfördes. I Sektion 3.1 redovisas hur den ursprungliga lösningen på Compare Testlab såg ut och vad som behövde förbättras. I Sektion 3.2 utreds SAN och NAS och vilken av dem som passar bäst till lösningen. I Sektion 3.3 utreds valet av operativsystem. Nästa sektion beskriver en utredning om vilka diskar som ska användas i lösningen. I Sektion 3.5 beskrivs sedan designen på det nya systemet. Efter det kommer en sammanfattning i Sektion 3.6.

3.1 Nuvarande uppbyggnad av systemet

Innan utredningen kunde börja så krävdes det att man tog reda på hur den existerande nätverkslagringslösningen såg ut hos Compare Testlab. Denna sektion kommer beskriva uppbyggnaden av den dåvarande lösningen och vilka problem som fanns i den.

Compare Testlabs nätverk består av ett huvudnätverk som är kopplat mot Internet. På detta nät sitter personal och användare och arbetar. Så som den ursprungliga lösningen var uppbyggd så fanns även nätverkslagringen och en server som körde XenServer på detta nät. Det fanns även ett LAN, TestLAN, där olika tester körs. Se Figur 3.1 för en bild av den ursprungliga lösningens uppbyggnad.

Den dåvarande lösningen använde sig av operativsystemet Openfiler, som är ett open source-alternativ, på servern för att fördela lagringen på nätet. På Openfiler-servern kördes två olika metoder för lagring. Den ena var iSCSI, som gjorde att lagringen blir blockbaserad, och användes till lagring för de virtuella maskiner som kördes på XenServer. Men det fanns även en NFS-koppling till lagringsservern, för att de vanliga datorerna på nätet skulle kunna

få tillgång till servern på filnivå och för att XenServern skulle kunna komma åt ISO-filer för olika operativsystem som den skulle virtualisera.

Den ursprungliga uppbyggnaden på lösningen hade vissa brister enligt kraven som ställdes. En av de största bristerna i denna lösning var att nätverkslagringen inte var isolerad från huvudnätverket, vilket betyder att hastigheten kunde bli långsam på grund av den höga trafiken i nätet. En annan stor brist var att om lagringsservern, Openfiler, gick ner i drift så drabbades många miljöer.

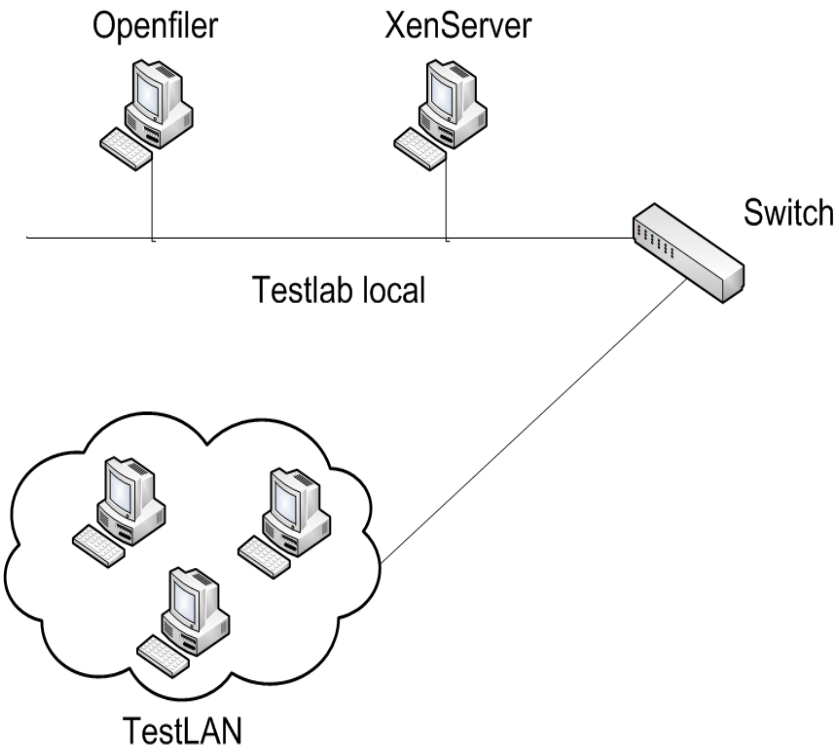
I denna utredning ska vi gå igenom lösningar på dessa problem och jämföra olika lösningars för- och nackdelar med varandra. De open source-alternativ som valdes att fokusera på var Openfiler och FreeNAS. I detta kapitlet ska dessa operativsystem jämföras med varandra och det ska redovisas varför just dessa två valdes ut. Först ska däremot olika alternativ till nätverkslagring beskrivas och utredas. Dessa två är NAS och SAN, vilka kommer att utredas i nästa sektion.

3.2 NAS eller SAN?

Nätverkslagring utgörs huvudsakligen av SAN eller NAS. För att få en bra nätverkslagringslösning så krävs det att man utreder olika alternativ som finns inom området. I denna sektion beskrivs både SAN och NAS mer utförligt. Det kommer bland annat att utredas skillnader, för- och nackdelar, kostnader samt när man ska använda sig av respektive lösning. I Sektion 3.2.1 beskrivs NAS medan SAN utforskas i Sektion 3.2.2.

3.2.1 NAS-lösningar

En NAS är en enhet som delar ut filer i ett nätverk, som andra klienter under samma nätverk kan komma åt. De är lätta att installera och administreringen sker ofta med hjälp av ett webbgränssnitt. En NAS är ur ett ekonomiskt perspektiv en billigare variant som lämpar sig bäst för privatpersoner och mindre företag. På marknaden finns det speciella datorer som är anpassade för NAS-lagring, NAS-enheter.



Figur 3.1: Nuvarande system

I detta arbete så behövdes NAS-protokollet NFS användas för att ge vanliga användare på Compares nät möjlighet till att komma åt filer på lagringsservern och för XenServer att komma åt olika ISO-filer för operativsystemen.

3.2.2 SAN-lösningar

SAN är en arkitektur som består av olika komponenter som utgör ett SAN-nätverk. Stora företag med många servrar drar nytta av SAN, eftersom hastigheterna är betydligt högre än hos ett vanligt nätverk. Men det finns också en stor nackdel med en SAN, nämligen kostnaden. Det är därför det endast lönar sig med en SAN för stora företag med stor plånbok. Komponenterna i en SAN kostar upp mot 100 gånger mer än komponenterna för ett vanligt hemmanätverk. Detta beror på att det är uteslutande fiber som används och stora switchar med många in- och utgångar. Dessa komponenter kan kosta stora summor

vilket gör att ett sådant system inte är lämpat för privatpersoner eller små företag.

Det finns många färdiga lösningar som man kan köpa till företag. Stora leverantörer som erbjuder färdiga SAN-lösningar är Netgear, Sun, IBM och HP. Dessa tillhandahåller alla komponenter som behövs för att bygga upp en SAN.

Det kostnadsintensiva hos en SAN är att allt måste vara kompatibelt med fiberoptik. Men med hjälp av iSCSI så går det att skicka data över ett vanligt kopparnät. iSCSI har blivit mycket populärt de senaste åren på grund av att kostnaden blir betydligt lägre om man redan har ett färdigt kopparnätverk. Denna utredning gick ut på att hitta en så billig lösning som möjligt, och därför var iSCSI som protokoll ett väldigt bra alternativ. Eftersom iSCSI är mer anpassat för att användas av en användare åt gången, på grund av dess blockbaserade lagring, så behövdes detta protokoll användas tillsammans med det filbaserade protokollet NFS för lösningen. Eftersom NFS ofta förknippas med NAS och iSCSI ofta förknippas med SAN så kan man säga att lösningen i detta arbete var en blandning av både NAS och SAN. För att kunna implementera NFS och iSCSI behövdes ett operativsystem som stödde båda protokollen och som var speciellt anpassat för NAS- och SAN-nätverk.

Efter att en utredning på olika lagringslösningsalternativ hade gjorts var det dags att bestämma vilket operativsystem som passade bäst för lösningen. I nästa sektion beskrivs två olika operativsystem, Openfiler och FreeNAS, och vilka för- och nackdelar de har.

3.3 Val av operativsystem.

De operativsystem som ingick i utredningen var Openfiler och FreeNAS. Dessa två ligger i lite mer framkant än andra och har en stadig grund med många funktioner. För att få en fungerande lösning behövdes nedanstående funktioner finnas med:

- **Failover**

  För att en sekundär server ska kunna ta över den primära serverns funktioner om denna kraschar, så är det nödvändigt att stöd för en failover-funktion finns. Detta

betyder att två servrar är sammankopplade där en agerar som en sekundär server som är redundant och en server agerar som en primärserver. När den primära servern dör så tar den sekundära över.

- **iSCSI**

  För att kunna använda SAN över ett TCP/IP-nätverk krävs det att man använder protokollet iSCSI. Operativsystemet behöver då ha stöd för iSCSI. Alternativet är Fibre Channel Protocol, men det är betydligt dyrare på grund av den speciella utrustning som behövs.

- **NFS**

  Lösningen ska också kunna dela ut filer med hjälp av protokollet NFS för att XenServer ska kunna komma åt ISO-filer till operativsystem som ska användas vid virtualisering.

- **RAID**

  Stöd för RAID var tvunget att finnas för att kunna få en pålitligare och snabbare lagringslösning.

- **NIC-bonding**

  NIC-bonding är ett sätt att binda ihop två nätverksportar så att dessa kan lastbalansera trafiken så att hastigheten höjs. Driftsäkerheten höjs också eftersom om ett nätverkskort går ner så kommer den andra fortfarande att fungera [31]. Eftersom lösningen behandlar både driftsäkerhet och prestanda så är detta en bra punkt att ta med i utredningen.

- **Systemkrav**

  Ett önskemål var att ta fram hur höga systemkraven var hos de båda operativsystemen.

Efter att kraven var uppsatta så var det dags att reda ut hur bra de två operativsystemen stödde dessa krav. I efterföljande sektion kommer de båda operativsystemen att analyseras för att ta reda på vilket som passar bäst.

3.3.1 Egenskaper hos Openfiler

I den här sektionen utforskas några viktiga egenskaper hos Openfiler för att se om det passar till den nya lösningen.

Nedan följer några av de egenskaper som finns hos Openfiler:

- **Bassystem**
  Openfiler baseras på operativsystemet rPath [26] och har ett fullständigt webbgränssnitt för konfigurering. Autentisering i Openfiler sker med hjälp av Pluggable Authentication Modules som kan konfigureras via webbgränssnittet.

- **Protokoll**
  Det finns stöd för dessa protokoll i Openfiler:

  - **SMB/CIFS**
  - **HTTP/WebDAV**
  - **NFS**
  - **FTP**
  - **iSCSI target**
  - **SSH**

  Openfiler stödjer iSCSI-target med stöd för även virtuella iSCSI-targets.

- **Hårdvara och volymhantering**
  I Openfiler finns det stöd för mjukvaru-RAID. Det finns även stöd för iSCSI initiator.

Det finns stöd för filsystemen; Ext2/3 [15], FAT [20] och NTFS [21]. Volymresursallokering kan ske i olika nivåer. Man kan allokera volymutrymme till grupper eller användare och även gäster.

- **Nätverk**

  Openfiler stödjer NIC-bonding men har ingen officiell stöd utav failover. För att kunna få failover att fungera måste man få igång det manuellt via utföranden av kommandon i terminalen.

- **Systemkrav.**

  Minimumsystemkraven för Openfiler:

  32-bit 1 GHz processor.
  512 Mb RAM.
  512 Mb Diskutrymme för minnesswap.
  1 Gb Diskutrymme för installation.
  100 Mb Ethernet-nätverkskort.
  Separata lagrings-volymer/diskar för export av data.

  Rekommenderade systemkrav är:

  64-bit 1.6 GHz processor.
  1 Gb RAM.
  1 Gb Diskutrymme för minnesswap.
  2 Gb Diskutrymme för installation.
  1 Gb Ehernet-nätverkskort.
  Separata lagrings-volymer/diskar för export av data.
  Hårdvaru-RAID-kontrollerare.

För att få den lösning med de funktioner som ställdes så krävdes det att Openfiler hade stöd för de funktioner som listades i Sektion 3.3. Som man ser i ovanstående punktlista så har Openfiler stöd för iSCSI, RAID och NIC-bonding. Däremot så saknas det en enkel lösning för failover. Det finns inga inställningar i webbgränssnittet som aktiverar failover, utan detta får göras helt manuellt. Även om failover får fixas manuellt, så stödjer Openfiler alla de andra funktionerna som krävdes enligt kravspecifikationen (se Sektion 2.1) [23].

3.3.2 Egenskaper hos FreeNAS

I den här sektion ska vi granska de egenskaper som finns hos FreeNAS och se om de överenstämmer med den lösning som ska fås fram.

Några av FreeNAS egenskaper är följande:

- **Bassystem**

  FreeNAS baseras på operativsystemet FreeBSD [16] och har ett fullständigt webbgränssnitt för konfigurering.

- **Protokoll**

  Det finns stöd för dessa protokoll i FreeNAS:

  - **SMB/CIFS**
  - **AFP**
  - **NFS**
  - **FTP**
  - **TFTP**
  - **RSYNC(client/server)**
  - **Unison**
  - **iSCSI target**

- **SSH**

- **Hårdvara och volymhantering**

  I FreeNAS finns det stöd för mjukvaru-RAID i bland annat nivåerna RAID0, RAID1 och RAID5. Det finns även stöd för diskkryptering, ZFS [32] och iSCSI initiator. Det finns stöd för filsystemen; UFS [29], Ext2/3, FAT och NTFS.

- **Nätverk**

  FreeNAS stödjer NIC-bonding.

- **Systemkrav.**

  Minimum systemkraven för FreeNAS är:

  Moderkort med x86 processor.
  128 Mb RAM.
  32 Mb Diskutrymme.
  Nätverkskort.

FreeNAS stödjer många funktioner, bland annat RAID, NIC-bonding, iSCSI. En funktion som däremot fattas är failover vilket är en stor nackdel [17].

3.3.3 Fördelar och nackdelar med Openfiler/FreeNAS

Denna sektion ska utifrån de egenskaper som beskrevs i föregående sektion sammanfatta för- och nackdelarna med både Openfiler och FreeNAS. Det operativsystem som passade bäst till lösningen implementerades sedan för att användas i den slutgiltiga produkten.

- **Fördelar med Openfiler**

  Några fördelar med Openfiler är att det finns support att köpa, vilket är bra för att få professionell hjälp istället för att behöva förlita sig enbart på forum. Openfiler är

också rikt på funktioner och verkar ha alla de funktioner som krävs för lösningen. Administrationsgränssnittet har också något bättre struktur vilket gör det lättare att hitta var man konfigurerar de olika inställningar som behövs.

- **Nackdelar med Openfiler**

  En nackdel med Openfiler är att det i administrationsgränssnittet inte finns stöd för failover. Man behöver därför konfigurera mycket manuellt genom att använda kommandon och ändra inställningar i filer. Detta kan lätt leda till fel som är svåra att hitta. Det ska dock sägas att FreeNAS inte har stöd för failover överhuvudtaget, vilket gör detta till en inte allt för stor nackdel.

- **Fördelar med FreeNAS**

  FreeNAS är mindre systemkrävande och kräver mycket lite utrymme på hårddisk jämfört med Openfiler. FreeNAS har också en något smidigare konfiguration av iSCSI och NFS. Som beskrivs i föregående sektion så har också FreeNAS mer protokoll att välja bland. Alla dessa protokoll kommer dock inte att användas för detta arbete.

- **Nackdelar med FreeNAS**

  En väldigt stor nackdel med FreeNAS är att det inte finns stöd för failover, vilket är ett önskemål i kravspecifikationen. Det verkar heller inte finnas stöd för att partitionera upp diskarna vilket gör att man inte kan köra iSCSI och NFS samtidigt på samma disk.

Som ses ovan så har Openfiler mer av den funktionalitet som ingick i kravspecifikationen för det här arbetet. Detta gjorde att det var ett bättre alternativ till lösningen, vilket ledde till att Openfiler valdes ut som det operativsystem som implementerades i arbetet. I nästa sektion kommer en utredning om vilka diskar som var bäst lämpade till lösningen. Mer om implementationen av Openfiler kommer sedan i Kapitel 4. I nästa sektion kommer olika typer av håddiskar att tas upp och det kommer tas upp vilken typ av hårddisk som passade bäst till lösningen.

3.4 Utredning av diskar

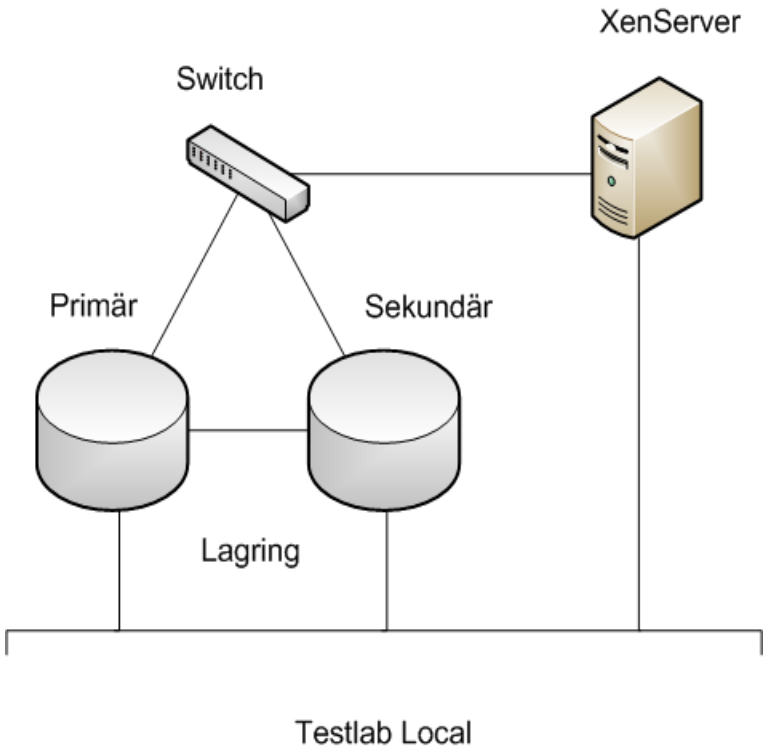
En begäran som kom upp under arbetet var att göra en liten utreding på vilken typ av hårddiskar som var bäst att använda för nätverkslagring. De hårddiskar som skulle jämföras var SATA-diskar och SCSI-diskar.

Eftersom SCSI-diskar saknades kunde inga tester utföras för att jämföra de olika diskarna. Istället fick man läsa sig till information om vilka för- och nackdelar de två hårddisktyperna har.

Genom den information som hittades så blev slutsatsen att det förmodligen var bättre att satsa på SATA-diskar. Hastighetsmässigt så är SATA inte mycket långsammare än SCSI nu för tiden. Även om SCSI-diskar är pålitligare så är de dyrare och storleken på dem är mindre. Eftersom priset var en viktig faktor i detta arbete så var det billigare alternativet bättre, det vill säga SATA.

3.5 Design på den nya lösningen

När utredningen var gjord skulle en design göras på den lösning som skulle implementeras. Den nya designen kan ses i Figur 3.2. Det nya systemet skulle skilja på nätverkslagringen och det vanliga nätverk som användare på Compare Testlab var uppkopplade mot och som i sin tur är uppkopplat mot Internet. Nätverket för lagringen skulle vara skiljt från detta system för att få en snabbare hastighet på överföringen. Den nya lösningen skulle också ha en failover mellan två servrar. Lösningen kommer därför att bestå av två lagringsservrar, en primär och en sekundär. Dessa kopplas till en switch, som i sin tur är kopplad till en server som kör XenServer. XenServer skulle använda sig av iSCSI-kopplingen till lagringsservrarna för att lagra virtuella maskiner. NFS-enheterna hos lagringsservrarna ska också användas av XenServer för att komma åt ISO-filer med operativsystem som ska virtualiseras.



Figur 3.2: Nya systemet

3.6 Sammanfattning

I detta kapitel har lösningens nuvarnade uppbyggnad beskrivits och olika alternativ till en förbättrad lösning tagits upp. Frågor som har diskuterats är om man ska använda sig av NAS eller SAN och vilket operativsystem som ska användas. Det som man kom fram till var att lösningen kommer bli en blandning av en SAN och en NAS eftersom både NFS (som är filbaserad) och iSCSI (som är blockbaserad) kommer att användas. Openfiler ansågs sig vara bäst lämpad efter de krav som ställdes hos operativsystemen. Det har också kommit fram till hur den nya designen ska se ut och vilka komponenter som ska ingå. För att kunna testa och demonstrera den nya lösningen så behöver lösningen implementeras. I nästa kapitel kommer denna implementation att beskrivas i detalj.

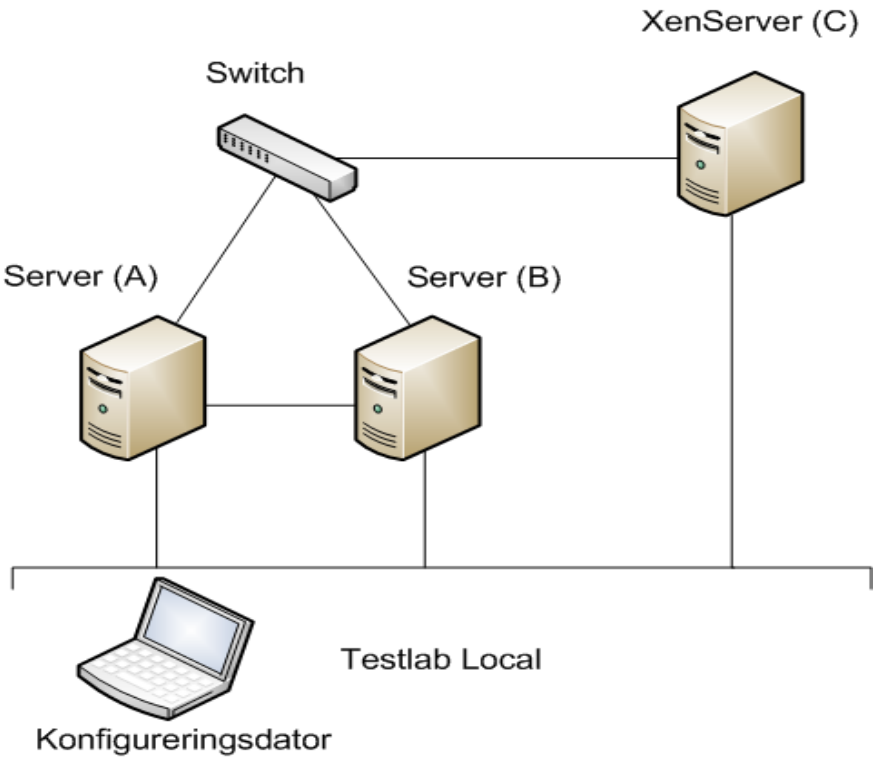
4 Implementation

Efter det att utredningen lett fram till slutsatsen att Openfiler är det bäst lämpade operativsystemet till den lösning som skulle tas fram, så började den riktiga implementationen i testmiljön. I den första sektionen beskrivs de förberedelser som krävdes innan start. Detta följs av en sektion som förklarar installationen av Openfiler och XenServer. Hur man får igång ett fullt fungerande High Availability-kluster med en failover-funktion förklaras i Sektion 4.3. Det uppkom ett problem i den första testmiljön, vilket var tvunget att åtgärdas. Problemet och åtgärden tas upp i Sektion 4.4. Hur man installerar virtuella maskiner med XenCenter förklaras i Sektion 4.5. I slutet sammanfattas kapitlet i Sektion 4.6.

4.1 Förberedelser

Innan man började installera och konfigurera Openfiler, så behövdes några åtgärder utföras. För det första behövdes det tas fram en testmiljö som liknar den miljö som Compare Testlab hade just då. Två servrar, som Openfiler installerades på, ställdes bredvid varandra (se Server A och Server B i Figur 4.1). Dessa två servrar hade tre nätverkskort. Det ena av nätverkskorten användes till att koppla samman servrarna med en korsad kabel. Denna nätverkskabel var till för att känna av om den andra noden levde. Från det andra nätverkskortet drogs en nätverkskabel till en router. Detta gjordes på båda servrarna. En tredje kabel drogs sedan, från båda servrarna, ut på huvudnätverket för att servrarna skulle var åtkomliga utifrån för konfigurering. En tredje server behövdes också (se Server C i Figur 4.1). På denna server installerades XenServer. Från denna server drogs två kablar, en till routern för att få kontakt med lagringsservrarna och en ut till huvudnätverket för att användare skulle kunna komma åt den. Anledningen till att man skulle ha ett litet lokalt nät med en router, var för att trafiken på lagringsnätet inte skulle behöva gå ut på det vanliga nätet och beblanda sig med trafiken där och minska hastigheten. Med ett litet lokalt nät kunde lagringsservrarna och XenServer kommunicera med varandra utan trängsel

i nättrafiken. För att kunna konfigurera lagringsservrarna och XenServer så kopplades en konfigurationsdator in till huvudnätverket.

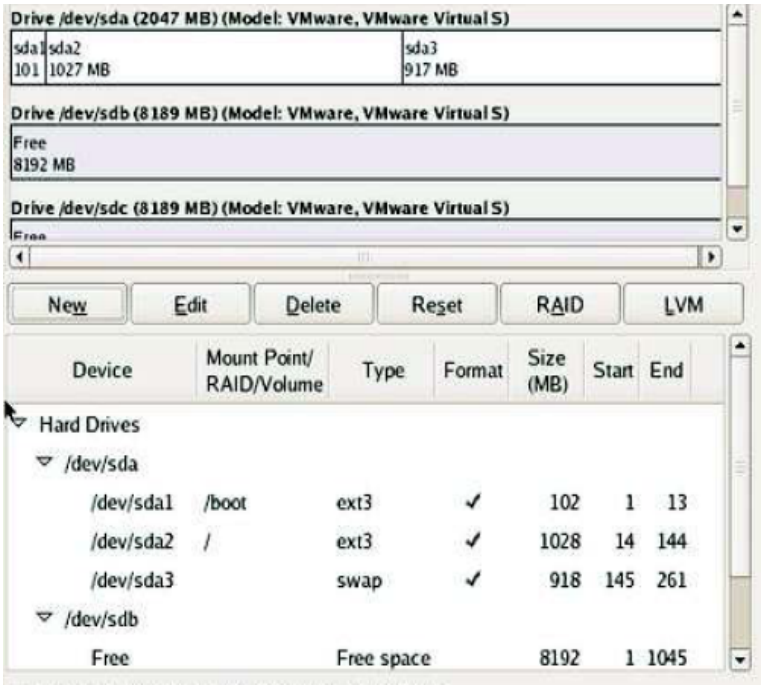


Figur 4.1: Testmiljön

4.2 Installation av Openfiler och XenServer

När allt var på plats och ihopkopplat så skulle Openfiler installeras. Detta är inte så komplicerat men det finns två saker som man ska tänka på. I och med att lagringsservrarna skulle konfigureras ihop till ett HA-kluster så skulle partitioneringen se ut på ett visst sätt. Hur den grafiska delen av partitioneringen såg ut kan man se i Figur 4.2. Man kan om man vill skapa partitionerna under installationen, eller som det valdes här, efter installationen. Anledningen till att det valdes att partitionera efteråt var att man slipper få partitionerna monterade automatiskt efter uppstart, det vill säga att man laddar in partitionen till en mapp som gör att man enkelt kan komma åt den. Om man väljer att partitionera

vid installation så behöver man ta bort en rad för varje partition i filen /etc/fstab. Hur partitioneringen ska se ut tas upp i Sektion 4.3.1.



Figur 4.2: Partitioneringen

Sedan skulle nätverkskonfigurationen utföras, som man kan se i Figur 4.3. Det fanns tre nätverkskort som var namngivna eth0, eth1 och eth2, dessa kryssades för så att nätverksporten aktiverades vid uppstart. Den som var kopplad till den andra noden, eth1, var till för igenkänning av den andra noden. Eftersom eth1 skulle vara kopplad till nod två, så kunde man sätta den direkt till en statisk IP-adress. Men man kunde också sätta den statiska IP-adressen efter installationen i webbgränssnittet, vilket gjordes i det här arbetet. Denna statiska IP-adress skulle användas av den andra noden för att de skulle hitta varandra. Till sist skulle man skriva in ett värdnamn till servern. Här valdes node1 och node2 till de två lagringsservrarna.

Det sista som var kvar av installationen av Openfiler var att ange ett root-lösenord (superlösenord), som används vid inloggning i textterminalen. Efter det började instal-

Figur 4.3: Nätverkskonfigurationen

lationen. Denna procedur upprepades för nod två. Webbgränssnittet kom man åt via en IP-adress, som Openfiler angav efter uppstarten, för all konfiguration. För att logga in, använde man Openfiler som användarnamn och password som lösenord. Lösenordet kunde man ändra när man väl var inloggad.

XenServer installerades på den tredje servern. Installationen krävde att man hade två CD-skivor, ett för grundinstallationen och den andra för att installera ett extrapaket. Extrapaketet bestod av komponenter som gjorde att man kunde virtualisera Linux. Installationen var rak på sak och bestod inte av några svårigheter. Det behövdes ingen partitionering då XenServer tog upp en hel hårddisk. Likt Openfiler så behövdes ett root-lösenord, som angavs innan installationen började.

När XenServer installerats och startats upp så får man upp ett terminalfönster med olika menyval (se Figur 4.4). Under ”Virtual Machines” kunde man se alla virtuella maskiner som kördes på servern. På lagringsservrarna finns både iSCSI och NFS tillgängliga för

XenServer. NFS användes till att lagra ISO-filer, som är en avbildad CD/DVD-skiva som innehöll ett operativsystem. För själva lagringen av virtuella maskiner användes iSCSI. Innan man kunde använda iSCSI och NFS i XenServer var man tvungen att först hitta dessa. Detta gjorde man under ”Disk and Storage Repositories”. En mer ingående förklaring på detta kommer i Sektion 4.5.

Figur 4.4: XenServer

När XenServer installerats och startats upp så får man upp ett terminalfönster med olika menyval. Här kan man låta XenServer leta efter iSCSI- och NFS-lagringar, samt se över alla virtuella maskiner. All iSCSI- och NFS-lagring är lagring som konfigureras hos Openfiler, men som XenServer kommer använda sig av. Till lagring av ISO används NFS-lagringen, medan iSCSI kommer att lagra alla installationer av virtuella maskiner.

4.3 Få igång High-Availability

När Openfiler och XenServer hade installerats var det dags att få igång ett fungerande HA-kluster mellan de båda lagringsservrarna. Innan man började var det några förberedelser man skulle göra. I första sektionen, 4.3.1, behandlas partitionering av hårddisken samt nätverkskonfigureringen. Detta följs av hur man får igång DRBD i 4.3.2 och sedan en sektion om vilka filer som skulle flyttas över till den partition som skulle synkroniseras

med partitionen hos den andra noden i Sektion 4.3.3. Detta följs av en sektion om hur man får igång Heartbeat. Detta beskrivs i Sektion 4.3.4. När väl själva HA-klustret var färdigkonfigurerat så skulle iSCSI och NFS ställas in på ett korrekt sätt, detta görs i Sektion 4.3.5. Dessa sektioner tar endast upp konfigureringen på en mer lättförståelig nivå. En mer detaljerad beskrivning finns i Appendix A.

4.3.1 Partitionering och nätverkskonfigurering

Skulle den ena noden gå ner så behöver den andra noden veta vilka filer och tjänster som ska sättas i drift. För detta behövs en partition som innehåller all viktig data som Openfiler behöver. Sedan behövs ytterligare en partition som ska användas till att lagra filer på. Partitionen för lagringen ska vara av typen LVM.

Eftersom det hade valts att partitionera efter installationen så gör man detta via kommandot *fdisk*. Först får man ta reda på var disken ligger i systemet. Diskenheterna ligger alltid i katalogen */dev* och enheterna brukar vara namngivna efter vilken typ av hårddisk det är. SCSI- och SATA-diskar brukar börja med ”sd” och vanliga hårddiskar med ”hd”. Sedan följs en bokstav i bokstavsordning där ”a” är den första och står för vilken hårddisk det är. Har man till exempel två SATA-hårddiskar så får den första namnet ”sda” och den andra ”sdb”. Sist i namnet finns en siffra som anger vilken partition det är på disken. Har man en SATA-hårddisk med tre partitioner så kommer dessa namn att vara ”sda1”, ”sda2” och ”sda3”. I det här fallet hos lagringsservrarna låg hårddisken på */dev/sda*, där ”sda1” var root-partitionen och ”sda2” var partitionen för swap. För att starta partitioneringen så började man med kommandot *fdisk*, följt av vilken disk som skulle partitioneras: *fdisk /dev/sda*.

Att konfigurera nätverket var lite lättare. Detta kan man enkelt göra via webbgränssnittet. Gick man in via ”System” så hittade man alla nätverksgränssnitt som finns i datorn. Det gränssnitt som ska användas till att koppla direkt till den andra noden ska ha en statisk adress. Den andra noden ska använda sig utav den statiska IP-adressen så att

noden vet vem den skulle kommunicera med. De övriga två skulle vara i DHCP (som är ett nätverksprotokoll som tilldelar IP-adresser dynamiskt), det vill säga få en IP-adress tilldelad från den lokala routern samt routern från Compares nät.

Efter det att man har satt upp en statisk IP-adress till de båda lagringsservrarna så ska man lägga till den andra nodens statiska adress i en textfil som ligger i */etc/hosts*. Längst ner i filen lägger man till den andra nodens IP-adress följt av den andra nodens värdnamn, som sattes till node1 och node2. Anledningen till man gör detta är för att noderna ska kunna hitta varandra, utan att behöva ange varandras IP-adresser. Istället för att skriva root@***ip_adress*** så kunde man skriva root@**node2**. Skriver man det sistnämnda går filen */etc/hosts* igenom för att kolla om node2 finns med. Finns den med så används den definierade IP-adressen för node2.

4.3.2 Konfigurering av DRBD

Efter partitioneringen och nätverkskonfigurationen så är nästa steg att sätta upp DRBD. För att sätta upp DRBD så behövs en konfigurationsfil, som ligger i */etc/drbd.conf*. För vad som ska stå i konfigurationsfilen, se Appendix B. I den här filen anger man vilka partitioner som ska synkroniseras mellan noderna. Man skriver ned vad man vill döpa DRBD-partitionen till samt var den ska lagras, till exempel */dev/sda3*. Slutligen anger man IP-adressen till den andra noden. När man har skrivit klart i drbd.conf så kopieras filen över till den andra noden.

DRBD-partitionerna skapades inte av sig själva utan gjordes manuellt. Detta gjorde man med två kommandon, ett för partitionen med viktiga data och ett för LVM-partitionen. När dessa var skapade så skulle tjänsten DRBD startas på båda noderna. I Figur 4.5 kan man se statusen för DRBD efter att det har startats.

Noderna har nu hittats men tillståndet är i Secondary/Secondary hos båda noderna, vilket innebär att båda noderna är inaktiva och att ingen utav dem är satt som primär. För att ändra så att tillstånden blir Primary/Secondary, det vill säga att nod 1 blir primär

```
[root@node1 ~]# service drbd status
drbd driver loaded OK; device status:
version: 8.2.7 (api:88/proto:86-88)
GIT-hash: 61b7f4c2fc34fe3d2acf7be6bcc1fc2684708a7d build by phil@fat-tyre, 2008-11-12 16:47:11
m:res              cs         st                 ds                         p  mounted  fstype
0:cluster_metadata  Connected  Secondary/Secondary  Inconsistent/Inconsistent  C
1:vg0drbd          Connected  Secondary/Secondary  Inconsistent/Inconsistent  C
```

Figur 4.5: DRBD-status

och nod 2 blir sekundär, så talar man om för DRBD vilken nod man skulle sätta som primär. I Figur 4.6 kan man hos den primära noden, när den precis satts som primär, se hur synkroniseringen har påbörjat. Figur 4.7 visas hur synkroniseringen ser ut hos den sekundära noden.

```
[root@node1 ~]# drbdsetup /dev/drbd0 primary -o
[root@node1 ~]# drbdsetup /dev/drbd1 primary -o
[root@node1 ~]# service drbd status
drbd driver loaded OK; device status:
version: 8.2.7 (api:88/proto:86-88)
GIT-hash: 61b7f4c2fc34fe3d2acf7be6bcc1fc2684708a7d build by phil@fat-tyre, 2008-11-12 16:47:11
m:res              cs          st                 ds                     p  mounted  fstype
...                sync'ed:    56.5%              (222984/505992)K
0:cluster_metadata  SyncSource  Primary/Secondary  UpToDate/Inconsistent  C
1:vg0drbd          PausedSyncS  Primary/Secondary  UpToDate/Inconsistent  C
```

Figur 4.6: DRBD-synkronisering på nod 1

```
[root@node2 ~]# service drbd status
drbd driver loaded OK; device status:
version: 8.2.7 (api:88/proto:86-88)
GIT-hash: 61b7f4c2fc34fe3d2acf7be6bcc1fc2684708a7d build by phil@fat-tyre, 2008-11-12 16:47:11
m:res              cs          st                 ds                   p  mounted  fstype
0:cluster_metadata  Connected   Secondary/Primary  UpToDate/UpToDate    C
...                sync'ed:    14.0%              (41037/47691)M
1:vg0drbd          SyncTarget  Secondary/Primary  Inconsistent/UpToDate  C
```

Figur 4.7: DRBD-synkronisering på nod 2

När synkroniseringen var klar stod det UpToDate/UpToDate vilket betyder att båda noderna är konsistenta och därmed innehåller samma data.

4.3.3 Konfigurering av Openfiler

Openfiler har en hel del konfigurationsfiler som behövs för att allt ska fungera. Dessa måste också finnas på bägge noderna. Görs en ändring i webbgränssnittet, så måste ändringen också kopieras till den andra noden. Till detta har man partitionen *cluster_metadata* till. Alla viktiga filer kopieras över till denna partition för att sedan skapa en symbolisk länk tillbaka. I den nod som är primär skapas först en mapp i root, det vill säga längst upp i filträdet. Sedan monteras partitionen så att den laddas in till den nya mappen. När detta är gjort så kopieras alla viktiga systemfiler till den nya mappen. Man skapar också samma mapp hos den sekundära noden, däremot så ska man inte montera partitionen, eftersom den endast ska monteras när en failover sker, alltså när den sekundära noden ska ta över driften.

Hos de båda noderna så ändrar man i filen emph/opt/openfiler.local/etc/rsync.xml. I denna konfigurationsfil så anger man de filer som ska kopieras över till den andra noden när en ändring sker, till exempel en ändring i inställningarna i webbgränssnittet. Hur hela filen ser ut kan man se i Appendix D.

4.3.4 Konfigurering av Heartbeat

Heartbeat behövs för att veta om den andra noden lever. Skulle en nod gå ner, tar den andra noden över. Det behövs skapas några filer för att få igång Heartbeat. Först så skapas textfilen */etc/ha.d/authkeys*. Den här filen är till för att sätta gemensamma rättigheter till noderna. Man väljer vilken typ av krypteringsmetod som ska användas. Det finns SHA1 [13], MD5 [25] och CRC [11] att välja mellan, där den förstnämnda är den säkraste utav dem. CRC är den minst säkra, men så länge miljön är fysiskt säker så går det att använda krypteringsmetoden CRC.

Den andra filen var */etc/ha.d/ha.cf*. Här anges information om klustret. Man anger vilka noder som tillhör klustret och nätverksgränssnittet som används för igenkänning av den andra noden. Dessa två filer ska vara identiska hos båda noderna.

Vad som ska hända vid en failover bestämmer man i *cluster.xml*. I den ska man ange vilka partitioner som ska sättas i bruk vid ett driftstopp hos ena noden. En HA-adress ska också anges samt den primära nodens värdnamn, i det här fallet node1. HA-adressen är den IP-adress man ska använda sig av när man ska komma åt webbgränssnittet. Även om primärnoden är nere så ska man kunna komma åt webbgränssnittet från samma IP-adress. Filen *cluster.xml* genererar filen */etc/ha.d/haresource*, efter att en ändring gjorts i webbgränssnittet. Det kunde till exempel vara att sätta på NFS- och iSCSI-tjänsten. När en tjänst satts på så tvingar man */etc/ha.d/haresource* att skrivas. Efter att */etc/ha.d/haresource* genererats så ska man kopiera över den till den andra noden, så att den sekundära noden vet vad den ska göra vid en failover.

4.3.5 iSCSI och NFS

I det här läget så fungerade övertagningen av tjänsterna när den primära noden stängdes ned. Det var endast när man skulle komma åt webbgränssnittet som övertagingen märktes av. När själva failover-processen var konfigurerad och färdig så ville man också att NFS- och iSCSI-konfigurationen skulle finnas kvar efter det att en nod hade dött. För att få detta att fungera så behövdes ytterligare fler konfigurationer (se Figur 4.8). För att få igång iSCSI så behövdes det skrivas en rad olika kommandon hos båda noderna. Likadant gällde för NFS. Samma metod gällde för både iSCSI- och NFS-konfigureringen. Metoden gick ut på att hos den primära noden kopiera över alla konfigurationsfiler till den mapp där alla viktiga filer från Openfiler ligger och skapa en symbolisk länk tillbaka. På den sekundära noden skulle man ta bort filerna och skapa en symbolisk länk tillbaka.

```
[root@node1 ~]# mv /etc/ietd.conf /cluster_metadata/etc/
[root@node1 ~]# ln -s /cluster_metadata/etc/ietd.conf /etc/ietd.conf
[root@node1 ~]# mv /etc/initiators.allow /cluster_metadata/etc/
[root@node1 ~]# ln -s /cluster_metadata/etc/initiators.allow /etc/initiators.allow
[root@node1 ~]# mv /etc/initiators.deny /cluster_metadata/etc/
[root@node1 ~]# ln -s /cluster metadata/etc/initiators.deny /etc/initiators.deny
```

Figur 4.8: Kommandon för att få igång iSCSI med failover

4.4 Problem med testmiljön

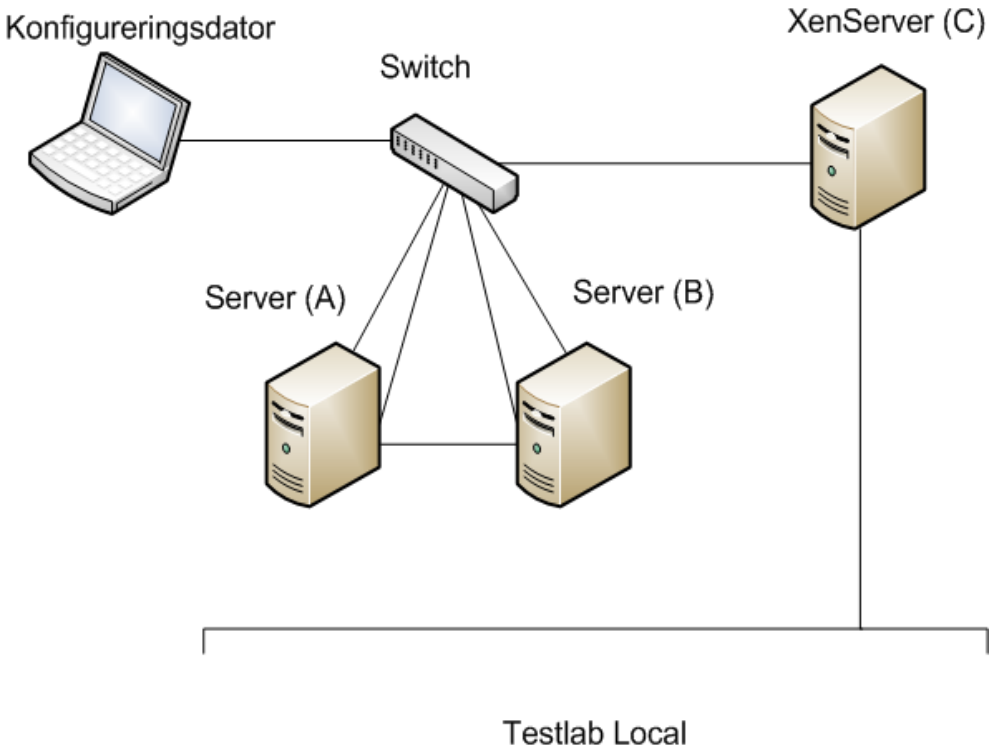
Det uppstod problem ett flertal gånger vilket gjorde att man fick installera och konfigurera om många gånger. De flesta problemen tas upp i Kapitel 5, men ett problem fann man i själva testmiljön. Detta problem kommer att förklaras i den första sektionen nedan, och i den andra sektionen kommer lösningen på problemet att beskrivas.

4.4.1 Problem med flera HA-adresser

När allt var uppsatt så upptäcktes ett problem med den testmiljö som använts. Som man ser i Figur 3.2 så krävdes det två HA-adresser. Den ena för att man skulle nå lagringsservrarna utifrån, från Compares nät, och den andra till XenServer. Ingen lösning kunde hittas, för att få fram två HA-adresser till två olika subnät till den testmiljö som först konfigurerats upp. Detta gjorde att man fick tänka om och göra några ändringar.

4.4.2 Modifierad lösning

De ändringar som fick göras var inte så stora och den lösning som man till slut kom fram till blev bättre. Istället för att ha en nätverkssladd ut till Compares nät för att komma åt webbgränssnittet, så kopplade man den till en större switch i det lokala nätet, det vill säga två nätverkskablar till samma switch från båda noderna. I början användes endast en 4-portars router, men när den nya lösningen togs fram så fick man tillgång till en 8-portars switch, vilket möjliggör att man kan skapa en NIC-bonding mellan dessa två nätverksgränssnitt. Detta var inte möjligt i den första lösningen, eftersom man kopplade de två kablarna till två olika subnät. Bild på den modifierade lösningen kan ses i Figur 4.9. När allt var på plats efter modifieringen och allt var ominstallerat från början så var det dags att installera några virtuella maskiner.

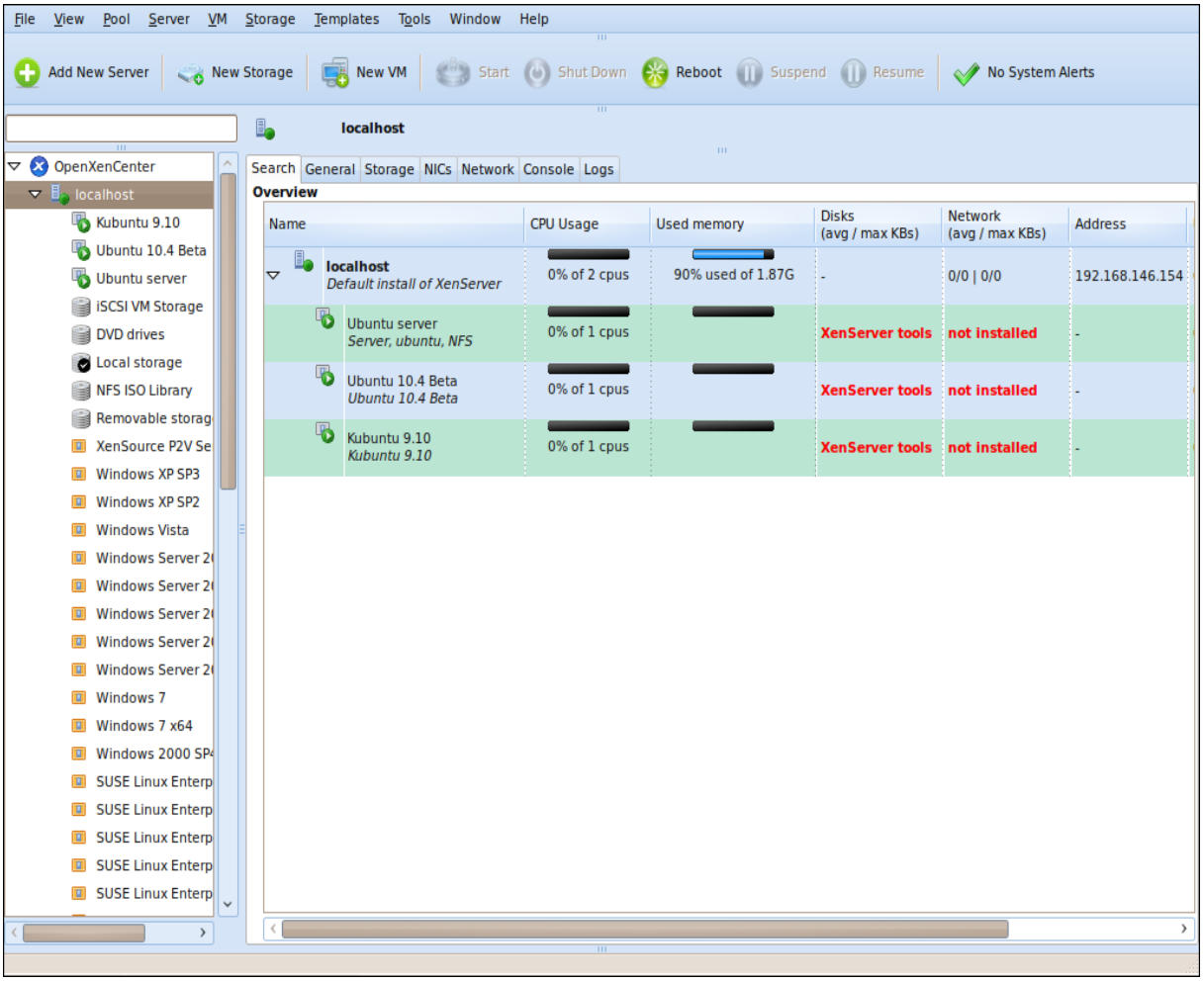


Figur 4.9: Den modifierade lösningen

4.5 Installation av virtuella maskiner med XenCenter

I denna sektion kommer det att tas upp hur man installerar en virtuell maskin med XenCenter. Det program som används är OpenXenCenter eftersom det är Linux som körs på klientsidan. I Figur 4.10 ser man en bild på hur OpenXenCenter ser ut. Till vänster ser man alla lagringsenheter och alla virtuella maskiner. Den stora tabellen till höger kan man se information om lagringsenheterna. Hos virtuella maskiner kan man se vilka nätverkskort som är tilldelade dem, vilken plats de är installerade på och en grafisk vy över operativsystemet.

Innan man kan installera en virtuell maskin så är man tvungen att finna en plats som man kan lagra den på. Dessa är konfigurerade hos lagringsservrarna så det är bara till att ange IP-adressen till dem. Man trycker på ”New Storage” och där kan man välja mellan iSCSI och NFS. Hos Compare Testlab använder man NFS till att lagra ISO-filer. Det finns ett specielt alternativ som heter ”NFS library” som man kan välja. I Figur 4.10 kan man



Figur 4.10: Bild på OpenXenCenter

se att man lagt till en NFS-lagring som innehåller ett ISO-bibliotek. När man väljer att installera en ny virtuell maskin så kan man välja bland dessa att installera ifrån. När man ska lägga till en iSCSI-enhet så anger man först IP-adressen till lagringsservern. Efter det får man upp en lista med iSCSI-enheter som finns konfigurerade hos lagringsservrarna som sedan kan läggas till i XenServer via OpenXenCenter.

För att installera en ny virtuell maskin så trycker man på knappen ”New VM”. Först får man upp en ruta där man kan välja vilket OS man ville installera. Väljer man ”other media” så får man välja mellan ISO:n som finns i ISO-biblioteket. Sedan får man välja

vilket/vilka nätverksportar som ska finnas med. Vart det ska installeras väljer man mellan de lagringar som man har funnit med hjälp av ”New storage”. När man är klar så startas allt upp och installationen börjar. Väljer man tabben ”Console” så kan man se en grafisk vy av den virtuella maskinen.

4.6 Sammanfattning

I det här kapitlet har det skrivits om hela processen från det att man kopplade ihop den första lösningen, till alla konfigurationer som gjorts samt de ändringar som gjordes till den nya lösningen. Det har också tagits upp hur man installerar en virtuell maskin med hjälp av OpenXenCenter. Denna process, från installation av Openfiler till det sista konfigurationskommandot, har gjorts ett antal gånger eftersom många problem har uppstått. När allt väl har fungerat så har tester kunnat utföras. Tester och problem kommer att tas upp i nästa kapitel.

5 Resultat

Det här kapitlet kommer huvudsakligen att fokusera på tester som har utförts under projektets gång. Den första sektionen kommer att jämföra den slutliga lösningen med de krav som ställdes i Sektion 3.3. I Sektion 5.2 beskrivs de olika tester som har genomförts och vilka resultat som har uppkommit av dessa tester. Nästa sektion, 5.3, handlar om de tester som kunde ha utförts om tid hade funnits. Efter detta kommer problem och lösningarna till dessa att tas upp i Sektion 5.4. Kapitlet kommer sedan att avslutas med en sammanfattning i Sektion 5.5.

5.1 Jämförelse mellan lösningen och funktionskraven

I den här sektionen ska den lösning som har utarbetats under arbetets gång jämföras med de funktioner som listades i Sektion 3.3. Nedan följer en redovisning av hur väl dessa funktioner blev uppfyllda.

- **Lösningens stöd för failover**
  Den slutgiltiga lösningen har gott stöd för failover för både vid användning av NFS och iSCSI. Failover-funktionen fungerar såväl i XenServer-miljö som i vanlig användning. Detta krav är uppfyllt.

- **Lösningens stöd för iSCSI**
  Lösningen har stöd för iSCSI. Detta iSCSI-stöd används till att förse olika virtuella maskiner med lagringsutrymme, vilket gör att detta krav är uppfyllt.

- **Lösningens stöd för NFS**
  NFS-kopplingen fungerar bra till lösningen. Det går att hämta ISO-filer till XenServern och användare kan komma åt NFS-diskarna för att lagra filer. Detta krav är uppfyllt.

- **Lösningens stöd för RAID**
  Även om Openfiler har stöd för RAID så har detta inte implementerats eller testats

på grund av tidsbrist och brist på hårddiskar att testa det på. Men Openfiler har stöd för detta och Compare Testlab har använt RAID förut så slutsatsen är att det förmodligen går att implementera även om det inte har gjorts under detta projekt.

- **Lösningens stöd för NIC-bonding**

  Lösningen kunde konfigureras med NIC-bonding i Openfilers webbgränssnitt. Den NIC-bonding som användes var lastbalansering, på grund av att kraven som ställdes var att NIC-bondingens främsta uppgift var att öka hastigheten på nätet. Detta fungerade väl och även detta krav är uppfyllt.

- **Lösningens sytemkrav**

  De systemkrav som krävs för lösningen verkar vara förhållandevis låga, som ses i Sektion 3.3.1. Med tanke på att de testdatorer som implementationen kördes på inte var några monsterdatorer borde inte systemkraven spela någon stor roll.

5.2 Tester

Under projektets gång har en hel del tester utförs i testmiljön för att kunna se att alla krav som ställdes på lösningen är uppfyllda. Nedan ska dessa tester redovisas ingående samt hur resultaten blev.

5.2.1 Test av failover i webbgränssnittet

För att testa om failover-funktionen fungerade så testades det om webbgränssnittet var åtkomligt även när den primära datorn var avstängd. Detta gick bra och man såg tydligt vilken nod som var aktiv. Man kunde se det när värdnamnet ändrade sig i webbgränssnittet. När man satte på den primära servern igen och stängde ner den sekundära servern, så tog den primära servern tillbaka tjänsterna. Ett annat test som utfördes var att kolla och se vad som hände om man drog ur den nätverkskabel som var kopplad mellan de båda lagringsservrarna, det vill säga den kabel som känner av om den andra noden lever. Det

som hände var att den primära datorn förblev primär och den sekundära förblev sekundär. Skulle denna nätverkskabel åka ur så är det alltså ingen fara.

5.2.2 Test av iSCSI/Failover

För att kunna testa iSCSI behövdes först och främst en utomstående dator kopplas till nätverket. Med denna dator kunde sedan en anslutning till iSCSI-disken upprättas. I Linux körs dessa kommandon:

sudo iscsiadm -m discovery -t st -p ***ip_adress****:3260*
sudo iscsiadm -m node -T ***initiator_id*** *-p* ***ip_adress****:3260*

Det första kommandot listar upp vilka iSCSI-enheter som finns att få tag i. För att upprätta anslutningen skriver man in det andra kommandot tillsammans med ett initiator ID som listades upp från den första kommandot. I Windows används programmet iSCSI initiator för att upprätta en anslutning. I programmet skriver man in en IP-adress för att sedan få upp en lista på iSCSI-enheter (om några) som man sedan kan ansluta sig till.

Båda dessa anslutningssätt fungerade och iSCSI-enheterna kunde kommas åt och formateras för att sedan användas till lagring.

Efter att en iSCSI-anslutning hade upprättats skulle sedan failover-funktionen testas i olika situationer. För att se så att den sekundära servern verkligen tog över den primära serverns arbete när denna eventuellt kraschade, så stängdes den primära datorn av för att se om den sekundära datorn tog över. Detta fungerade väl och disken var åtkomlig även när den primära datorn var avstängd.

Ett annat test som utfördes var att föra över filer till iSCSI-disken, låta den sekundära disken ta över lagringen, och sedan kolla om filerna fortfarande låg kvar. Filerna låg kvar och testet var lyckat.

En annan variant av detta test var att stänga av primärdatorn under tiden man förde

över en förhållandevis stor fil. Då kunde man se om den sekundära datorn återupptog filöverföringen. Detta test fungerade bra, man märkte dock en liten fördröjning i filöverföringen när den sekundära datorn tog över.

5.2.3 Test av NFS/Failover

För att testa NFS i Linux var man först tvungen att montera NFS-disken på datorn med kommandot:

mount -t nfs <ip_adress>:/nfs_volume monteringplats

Detta gjorde så att en NFS-volym monterades på datorn. För att testa om den kunde användas så överförde man filer till och från disken. Det fungerade bra. Hastigheten var dock märkbart sämre än när man överför till och från en iSCSI-enhet.

För att testa failover-funktionen för NFS så utfördes liknande tester som för iSCSI. Den primära datorn stängdes av och NFS-disken kunde fortfarande användas. Överföringar återupptogs också när man stängde av den primära datorn även om det tog längre tid för NFS än för iSCSI att återuppta en filöverföring. Dock uppstod ibland några problem med NFS-överföringen. Detta tas upp i Sektion 5.4.

5.2.4 Test av XenServer

Den huvudsakliga användningen av iSCSI-anslutningen var att använda den för lagring till de virtuella maskiner som kördes på XenServern. För att testa så att en användare inte skulle märka av något om en failover uppstod så installerades några virtuella maskiner som körde olika operativsystem genom mjukvaran XenCenter. För att installera dessa så användes den NFS-koppling som hade implementerats för att komma åt en NFS-disk för att hämta de ISO-filer av de operativsystem som skulle virtualiseras.

Efter detta så valdes en iSCSI-disk som lagringsenhet för det vitualiserade opera-

tivsystemet. Den primära datorn stängdes sedan av för att framkalla en failover. Den sekundära datorn tog över utan problem och man märkte knappast någon fördröjning när man använde det virtualiserade operativsystemet.

5.2.5 Test av NIC-bonding

De sammanbundna nätverkskorten behövdes också testas så att de fungerade på ett korrekt sätt. För att kontrollera så att anslutningen till lagringsservrarna fortfarande fanns kvar så kopplades en av nätverkskablarna ut ur switchen under en filöverföring. Om NIC-bondingen fungerade så skulle filöverföringen fortgå utan problem, vilket den också gjorde. Detta testades på båda noderna och inga problem kunde hittas. NIC-bondingen var inställd på lastbalansering, vilket var svårt att testa eftersom hastigheten aldrig var uppe i de nivåer som var nödvändiga för att göra ett riktigt hastighetstest.

5.3 Tester som ej utfördes.

I den här sektionen ska några tester, som borde ha utförts om tiden medgett det och om den utrustning som behövdes hade funnits, diskuteras.

5.3.1 Test av backup/restore

Ett test som inte kunde utföras på grund av tidsbrist var att återställa en kraschad dator till att ingå i en primär/sekundär-relation igen. Ett scenario skulle vara om den primära datorn kraschade totalt så att man blev tvungen att installera om Openfiler. Efter installationen så vore det bra om man kunde konfigurera upp allt igen utan att behöva installera om den sekundära datorn eller utan att förlora data. För detta fanns det ingen tid för att testa, men detta kanske kunde fungerat med funktionen backup/restore som Openfiler har i webbgränssnittet. Med backup/restore menar man att man tar en snapshot av systemets konfigurering till att kunna återställa systemet till hur det var innan.

Ett annat scenario är om en av hårddiskarna i servern kraschar och som man skulle behöva byta ut. Detta var inte möjligt att testa eftersom det inte fanns mer än en hårddisk i datorerna och att det även inte fanns några extra hårddiskar att byta ut med.

5.3.2 Test av överföring mellan virtuella maskiner

Ett annat test som aldrig blev utfört var att testa hur överföring mellan två virtuella maskiner fungerade. Man skulle kunna tänka sig att överföra filer och sedan framkalla en failover för att se om filöverföringen fortsatte. Men eftersom de virtuella maskinerna fortfarande fungerade efter att en failover hade skett så kan man dra slutsatsen att detta test borde fungera.

5.4 Problem

Under testerna har en hel del problem uppstått. I denna sektion ska dessa problem och deras lösningar beskrivas.

5.4.1 Floppy disk error

Floppy disk error uppstod när man skulle skapa nya volymer i webbgränssnittet:

end_request: I/O error, dev fd0, sector 0
Buffer I/O error, dev fd0, sector 0

Detta berodde på att Openfiler försökte gå igenom alla diskar och fastnade på diskettstationen eftersom denna inte fanns. Detta löstes genom att gå in i BIOS och avaktivera diskettstationen.

5.4.2 Automatiskt iSCSI i Linux

Ett problem, som visade sig inte vara något större problem, var att få en automatisk anslutning till iSCSI vid uppstart av Linux. Problemet här var nog att Linux vid uppstart startade nätverkstjänsterna efter det att det försökte montera iSCSI-diskarna. Detta gjorde att den inte fann diskarna och inte kunde montera dem. Detta blev till slut inget problem eftersom diskarna inte skulle användas på det här sättet.

5.4.3 Problem med NFS

Det uppstod ett flertal problem med NFS. Ett fel uppstod när man försökte lägga till NFS-tjänsten i webbgränssnittet. När man tryckte på enable NFS så blev sidan blank och tjänsten aktiverades inte. Detta var en stor gåta och en lösning på detta problem hittades aldrig. Det som gjordes för att lösa problemet var att installera om Openfiler. Efter ominstallationen fungerade allting väl och problemet uppstod aldrig igen.

Ett annat problem uppstod när en failover hade skett och den sekundära servern hade tagit över. När en klient försökte föra över filer till en NFS-disk så kom detta felmeddelande upp:

RPC:fragment too large

Vad detta berodde på är svårt att säga eftersom någon bekräftad lösning aldrig hittades. Detta hade säkerligen med klienten att göra eftersom när man bytte klient så fungerade det. Men samma fel uppstod senare också på den nya klienten vilket var mycket märkligt.

Ett tredje fel som uppstod med NFS var att varje gång man lägger till en ny NFS-volym så måste man kopiera över filen */etc/ha.d/haresource* från den primära servern till den sekundära servern. Man ska helst inte heller skapa en ny volym när den sekundära servern är aktiv. Om man gör det måste man manuellt skapa en volym med samma namn på den primära datorn eller kopiera över */etc/ha.d/haresource*. Detta hände i mindre grad ibland

med iSCSI också men då räckte det att bara starta om iSCSI-tjänsten. Varför detta problem uppstod var att synkroniseringen mellan servrarna inte fungerade till hundra procent. Den fil som hanterade denna synkronisering var filen *cluster.xml* (se Appendix E).

5.4.4 Problem med HA/Failover

Det uppstod några fel ibland när konfigureringsfilerna hade några felskrivna sökvägar eller diskar. Ett ganska stort problem i designen uppkom när man inte kunde ha två olika high availability-adresser, en inuti det interna lagringsnätverket och en utåt mot Compare Testlabs nätverk. Detta gjorde så att den nuvarande designen uppkom.

När något blir fel i anslutningen mellan den sekundära servern och den primära servern kan en så kallad "Split-Brain" uppstå. Detta betyder att båda servrarna tror att den andra är död och båda servrarna tar över resurser som de egentligen inte ska göra och blir primära servrar. När ett sådant fel uppstår får man detta meddelande:

Split-Brain detected, dropping connection!

Felet löses manuellt genom att skriva:

drbdadm secondary resource drbdadm – –discard-my-data connect resource

på den server som ska vara sekundär och:

drbdadm connect resource

på den servrer som ska vara prmär.

Ett annat konstigt problem som uppkom var att båda servrarna hade svårt att finna

varandra och synkningen tog lång tid efter en failover. Detta kan möjligen bero på fel i hårdvaran. Efter ett byte av nätverksport för den korsade nätverkskabeln som binder ihop de två servrarna så fungerade allt bra igen.

Andra hårdvarufel uppstod. Ett nätverkskort klagade på MAC-adressen och man kunde inte få någon kontakt med den. Efter ett byte av nätverkskortet så löstes detta problem.

5.5 Sammanfattning

I detta kapitel har den slutgiltiga lösningen granskats utgående från de krav som ställdes på lösningen. De tester som har utförts har också beskrivits och hur resultatet blev för dessa. Några tester som av olika skäl inte utförts, men som borde ha utförts, har även de redovisats. Slutligen har de problem som har uppstått under projektets gång tagits upp tillsammans med deras lösningar. I nästa kapitel ska de slutsatser som har dragits av projektet redovisas.

6 Slutsats

Det här kapitlet kommer att gå igenom arbetet som har utförts under det här projektet och sammanfatta det. Det kommer även beskriva olika alternativa lösningar och påbyggnader som man skulle kunna tänka sig att lägga till i lösningen. Sektion 6.1 består av en utvärdering av arbetet och Sektion 6.2 består av en utvärdering av själva lösning och jämför hur bra denna lösning uppfyller kravspecifikationen. Sektion 6.3 beskriver olika alternativa lösningar som kunde ha utförts. Sektion 6.4 går igenom om det eventuellt går att bygga ut lösningen på något sätt. Efter detta så sammanfattas projektet i Sektion 6.5. Kapitlet sammanfattas sedan i Sektion 6.6.

6.1 Utvärdering av projektet

Enligt kravspecifikationen för projektet så skulle arbetet utföras i tre olika steg; utredning, implementation och dokumentation. I denna sektion ska var och en av dessa punkter gås igenom för att undersöka hur väl arbetet har utförts utifrån dessa riktlinjer.

- **Utredningen**

  Utredningen gick ut på att göra en undersökning av olika lösningar inom nätverkslagring. De två arkitekturer som fokuserades på var SAN och NAS. Efter detta så gjordes en utredning om vilket operativsystem som skulle användas.

  Det första steget i utredningen var att undersöka vilken litteratur som finns om detta ämne. Det fanns inte många böcker som behandlade nätverkslagring och i synnerhet inte om SAN och NAS. Efter att ha fått tag i en bok [24] så kunde ämnet börjas studeras. För att få en så klar och bra bild av vad ämnet handlade om så lästes många webbsidor och artiklar om SAN, NAS och om nätverkslagringstermer i allmänhet, såsom olika protokoll och hårdvara. För att få en bättre förståelse för hur allt hängde samman installerades olika operativsystem på en testmiljö och sedan experimenterades det med olika metoder för nätverkslagring. Sammanfattningsvis kan man säga

att ämnet kunde ha lästs på mer innan själva experimenteringen påbörjades, även om man vanligtvis lär sig mer i praktiskt arbete än att studera teori.

Utredningen ledde till att lösningen som skulle implementeras bestod av operativsystemet Openfiler, som installerades på två servrar, och att lösningen blev en blandning mellan en SAN- och en NAS-arkitektur. De två lagringsservrarna agerade som en primär och en sekundär nod och som hade en failover-funktion.

- **Implementation**

  Utredningen och implementationen gick in i varandra lite. För att få en klar bild av vilka för- och nackdelar de olika operativsystemen och protokollen hade, behövdes en del experimentation i utredningen. För att kunna testa och sedan till slut kunna göra en slutgiltig implementation så behövdes en testmiljö sättas upp. Implementationsdelen krånglade en del ibland men efter ett tag togs ett fungerande system fram. Avslutningsvis kanske det skulle ha varit bättre att ha haft en längre utredningstid som skulle ha resulterat i en bättre och mer uttänkt design från början.

- **Dokumentation**

  Dokumentationen som lämnades in till Compare var en installationsmanual (se Appendix A) som innehöll information om hur man konfigurerade en failover-funktion i Openfiler. De ska också ha en kopia av uppsatsen. Dokumentationen har genom arbetet gått mycket bra.

6.2 Utvärdering av lösningen

Som ses i Kapitel 2 så ställde kravspecifikationen krav på följande områden; driftsäkerhet, pålitlighet, prestanda, skalbarhet och användarvänlighet. Denna sektion ska analysera vilka av dessa krav som blev uppfyllda i den slutgiltiga lösningen.

- **Driftsäkerhet**

  Lösningen implementerades med en failover-funktion som består av två servrar där

en agerar primär och en sekundär. Detta gör att driftsäkerheten har höjts märkbart med tanke på att den primära serverna kan krascha utan att användaren märker någon större skillnad, eftersom den sekundära tar över. NIC-bondingen bidrar också till driftsäkerheten, genom att om en nätverksport dör så tar den andra över.

- **Pålitlighet**

  I arbetets gång fanns det varken tid eller material för att testa RAID. Den information som kom fram under utredningen så har Openfiler stöd för RAID och Compare Testlab har använt RAID innan.

- **Prestanda**

  Den prototyp som togs fram har ett lagringsnät som är skilt från den vanliga nättrafiken som sker på Compare Testlab. Detta kommer förhoppningsvis att höja hastigheten. Tyvärr har detta inte hunnits testats. NIC-bondingen kan förmodligen också här bidra lite. Med NIC-bonding inställt som lastbalansering så kommer trafiken balanseras över flera nätverksportar och därmed förhoppningsvis skapa en högre hastighet.

- **Skalbarhet**

  Skalbarheten har inte testats eller utretts i det här arbetet. Detta mest på grund av tidsbrist och brist på extra hårddiskar för att kunna testa att bygga ut testmiljön.

- **Användarvänlighet**

  Det gränssnitt som Openfiler har är ganska klart och enkelt att förstå. Användarvänligheten är därför inget större problem. Det finns support att köpa till Openfiler om man så vill. En sak som inte är vidare enkel är failover-konfigurationen. Då detta görs manuellt kan man inte direkt påstå att det är användarvänligt. Men när allt är uppsatt så fungerar Openfiler utan problem.

6.3 Alternativa lösningar

Den lösning som har beskrivits i denna uppsats består huvudsakligen av servrar som kör Openfiler med iSCSI och NFS. Det finns ett flertal alternativa lösningar för en sådan lösning som har beskrivits i den här uppsatsen. En av de alternativa lösningarna som kunde implementerats var att använda FreeNAS. I utredningen i Kapitel 3 så var slutsatsen att denna lösning inte passade så bra till den aktuella uppgiften.

Några andra lösningar som finns på marknaden är såklart att köpa riktiga fibernätverk som kör Fibre Channel Protocol på speciella SAN-nätverk. Detta skulle nog vara avsevärt snabbare och säkrare men också extremt mycket dyrare.

Sammanfattningsvis så är nog denna lösning den bästa med tanke på omständigheterna. Den är både billig och förhållandevis bra. Den fungerade bra på testmiljön och kommer nog förmodligen fungera bra när den driftsätts.

6.4 Framtida påbyggnad

I framtiden skulle man kunna tänka sig att bygga ut lösningen för att göra den ännu säkrare och snabbare. Några punkter att titta närmare på är följande:

- Lagringsservrarna och XenServern kan ställas in så att de stödjer ”Jumbo Frames” [14], det vill säga att de ska kunna skicka 9000 bytes istället för 1500 bytes. Detta bör även ställas in hos switchen. Allt detta skulle öka prestandan märkbart.

- För att höja driftsäkerheten bör två switchar användas istället för en enda. Man skulle då kunna koppla varje maskin till båda switcharna och på det sättet garantera att lösningen fortfarande fungerade om den ena switchen skulle haverera.

- En annan eventuell utbyggnad är att använda två XenServrar i ett ”kluster”. Detta skulle kunna göra så att en sekundär XenServer skulle ta över den primära XenServerns uppgifter och starta upp de virtuella maskinerna om den primära skulle krascha.

- Heartbeat skulle kunna skickas över flera vägar. Till exempel via, seriekabel, nätverkskabel och switchen. Detta kan göras med multicast [12].

För att testa prestandan på de virtuella maskinerna skulle man sedan kunna göra ett hårddisk-prestandatest, till exempel ”Bonnie++” [9].

6.5 Sammanfattning av projektet

Detta projekt har varit väldigt lärorikt. Under projektets gång har större kunskaper inom nätverk i allmänhet men främst inom nätverkslagring uppnåtts. Också kunskaper inom Linux har ökat märkbart genom arbetet.

Projektet har flutit på förhållandevis bra även om det skulle kunna ha lagts upp bättre (se Sektion 6.1). En liten besvikelse inom projektet var att det knappt var någon programmering alls. Arbetsplatsen har varit bra, med trevlig personal och med den utrustning som behövdes.

Avslutningsvis kan man säga att det har varit ett projekt med mycket arbete men väldigt lärorikt. Om mer tid hade funnits kunde man ha utvecklat arbetet ännu mer men det arbete som utfördes blev bra i slutändan och kunden blev nöjd.

6.6 Sammanfattning

I detta kapitel har en slutsats av projektet beskrivits. Det har gjorts en utvärdering av arbetet, alternativa lösningar har diskuterats, framtida påbyggnader har redovisats och en sammanfattning av projektet som avslutar det hela har givits.

Referenser

[1] Svensk IT Funktion AB. Vad ar virtualisering? [cited 20100526]. http://www.virtualisering.nu/fakta-om-virtualisering/, 2009.

[2] Ed L. Cashin. Kernel korner - ata over ethernet: Putting hard drives on the lan [cited 20100526]. http://www.linuxjournal.com/article/8149, 2005.

[3] Compare [cited 20100526]. `http://www.compare.se/index.asp?id=3387&typ=&lang=`.

[4] FreeNAS [cited 20100526]. `http://freenas.org/freenas`.

[5] Jesper Wallin [cited 20100526]. Lvm: Logical volume manager. http://www.ifconfig.se/lvm-logical-volume-manager.

[6] LinuxHA [cited 20100526]. Linux-ha. http://www.linux-ha.org/wiki/Main_Page.

[7] Openfiler [cited 20100526]. `http://www.openfiler.com/about`.

[8] Citrix. Citrix xenserver [cited 20100526]. http://www.citrix.com/.

[9] Russell Coker. bonnie++[cited 20100615]. http://www.coker.com.au/bonnie++/.

[10] Microsoft Corporation. Microsoft smb protocol and cifs protocol overview [cited 20100526]. http://msdn.microsoft.com/en-us/library/aa365233(VS.85).aspx, 2010.

[11] CRC. Crc [cited 20100526]. http://www.mathpages.com/home/kmath458.htm.

[12] Juan-Mariano de Goyeneche. Multicast over tcp/ip howto[cited 20100615]. http://tldp.org/HOWTO/Multicast-HOWTO.html.

[13] Philip A. DesAutels. Sha1 [cited 20100526]. http://www.w3.org/PICS/DSig/SHA1_1_0.html.

[14] Phil Dykstra. Gigabit ethernet jumbo frames [cited 20100615]. http://sd.wareonearth.com/ phil/jumbo.html.

[15] ext. ext [cited 20100526]. http://tldp.org/HOWTO/Filesystems-HOWTO-6.html.

[16] FreeBSD. Freebsd [cited 20100526]. http://www.freebsd.org/.

[17] FreeNAS. Freenas features. `http://freenas.org/features`.

[18] Daniel Jonsson. Natverkslagring, san och nas [cited 20100526]. http://www.omwlan.se/artiklar/natverkslagring-san-nas.aspx, 2008.

[19] R. Kayne. What are disk arrays? [cited 20100526]. http://www.wisegeek.com/what-are-disk-arrays.htm, 2010.

[20] Microsoft. Fat [cited 20100526]. http://support.microsoft.com/kb/100108/.

[21] Microsoft. Ntfs [cited 20100526]. http://support.microsoft.com/kb/100108/.

[22] Bradley Mitchell. Introduction to nas - network attached storage [cited 20100526]. http://compnetworking.about.com/od/itinformationtechnology/l/aa070101a.htm.

[23] Openfiler. Openfiler features [cited 20100526]. http://www.openfiler.com/products/feature-summary.

[24] Christopher Poelker and Alex Nikitin. *Storage Area Networks for dummies*. Wiley publishing,Inc., 2nd edition, 2009.

[25] R. Rivest. Md5 [cited 20100526]. http://tools.ietf.org/html/rfc1321.

[26] rPath [cited 20100526]. rpath. http://wiki.rpath.com/wiki/rPath_Linux.

[27] RPC. Remote procedure call [cited 20100526]. http://www.redhat.com/mirrors/LDP/LDP/nag2/x-087-2-appl.rpc.html.

[28] Charlie Schluting. Storage networking 101: Understanding the fibre channel protocol [cited 20100526]. http://www.enterprisenetworkingplanet.com/netsp/article.php/3690921, 2007.

[29] Martin Streicher. Ufs [cited 20100526]. http://www.ibm.com/developerworks/aix/library/au-speakingunix11/index.html.

[30] Bill Swingle. Network file system (nfs) [cited 20100526]. http://www.freebsd.org/doc/handbook/network-nfs.html.

[31] Wikipedia. Nic-bonding[cited 20100615]. http://en.wikipedia.org/wiki/Link_aggregation.

[32] ZFS. Zfs [cited 20100526]. http://hub.opensolaris.org/bin/view/Community+Group+zfs/.

A Appendix A - Dokumentation

Denna dokumentaion kommer att ta upp hur man konfigurerar upp en failover-funktion mellan två noder. Först kommer en översikt på hur uppsättningen i testmiljön som vi använt oss av ser ut. Det kommer sedan en kort beskrivning på installationen av Openfiler. Sedan följs hur partitioneringen och nätverkskonfigurationen går till. Efter det börjar konfigurationen av själva failover-funktionen som är uppdelat i sju steg.

- Steg 1: Få igång DRBD.
- Steg 2: Konfigurering av LVM-partitionen.
- Steg 3: Få Heartbeat att fungera.
- Steg 4: Förflyttning av viktiga konfigurationsfiler, som Openfiler använder sig av, till cluster_metadata (för synkronisering med den andra noden).
- Steg 5: Konfigurering av NFS och Samba.
- Steg 6: Konfigurering av iSCSI.
- Steg 7: Slutskedet av konfigurationen.

Översikt

Här kommer det en liten beskrivning på hur lagringsenheterna är uppsatta. Båda enheterna har tre nätverksportar. En kopplas mellan varandra. Det är viktigt att nätverkskopplingen mellan noderna har gigabit-anslutning så att hastigheten för synkroniseringen ska vara hög. Så här ser uppsättningen ut för våran testmiljö efter att allt är konfigurerat:

Primärnoden (A)

Värdnamn: node1

eth0 och eth2 sätts ihop till bond0 (NIC-bonding). IP-adress: 192.168.0.102.
eth1 (kopplas direkt till node2) sätts till statisk IP till 192.168.10.1
cluster_metadata finns i /dev/sda3
LVM-partitionen finns i /dev/sda4

Sekundärnoden (B)
Värdnamn: node2
eth0 och eth2 sätts ihop till bond0 (NIC-bonding). IP-adress: 192.168.0.150.
eth1 (kopplas direkt till node1) sätts till statisk IP till 192.168.10.2
cluster_metadata finns i /dev/sda3
LVM-partitionen finns i /dev/sda4

HA-adress: 192.168.0.100 (den gemensamma IP-adressen).

Dessa värden ovan kan skilja sig från de värden ni har. Värden ovan kommer användas i den här dokumentaionen, så har ni andra värden så använder ni givetvis dem.

Installation av Openfiler
Denna guide går efter att man har Openfiler nyinstallerat. Här följer en kort beskrivning av installationen av Openfiler. Mycket är rakt på sak, men det är några saker man ska tänka på.

1. När ni kommer till partitioneringen (välj manual partitioning) räcker det med en root-partition på ungefär 2 Gb och en swap-partition på 2 Gb. De andra partitioner som behövs konfigureras efter installationen. Däremot om ni använder RAID så kan det vara lättare att konfigurera dem redan nu.

2. Nätverkskonfigurationen räcker det att ni aktiverar (trycker i Active on Boot) nätverksporten som man ska använda för att komma åt webbgränssnittet från en annan dator. Den statiska IP-adressen och värdnamnet ställs in senare.

3. Sen väljer man i vilken tidszon man bor i följt av ett root-lösenord, innan installationen börjar.

När installationen är klar hos båda noderna är det dags för en hel del konfigurering. Det är många kommandon som ska skrivas och att trycka in allt manuellt hos varje nod är tidskrävande, så det är rekommenderat att ansluta sig till noderna med hjälp av SSH. Detta gör man via en klientdator som kan komma åt noderna. Man skriver ssh root@(IP-adress_till_node1) för att komma komma åt node1 och ssh root@(IP-adress_till_node2) för att komma åt node2.

Partitionering

I denna sektion tas det upp vilka partitioner som behövs och hur man skapar dessa. Detta ska utföras hos båda noderna. Först skapas partitionen cluster_metadata. Denna partition ska innehålla alla viktiga konfigurationsfiler som ska synkroniseras med den andra nodens cluster_metadata-partition. Först letar man upp vilken disk som ska paritioneras. Sedan skriver man in kommandot:

*fdisk /dev/**hdd***

där hdd är disken som ska partitioneras. Är man osäker kan man skriva kommandot fdisk -l för att få upp en lista på hårddiskar och partitioner som finns. I vårt fall heter hårddisken sda och då skriver man:

disk /dev/sda

Efter det startas partitioneringsprogrammet.

1. Skriv n för att skapa en ny partition.

2. Sedan väljer man p för att skapa en primär partition.

3. Om man har två paritioner sedan tidigare (en root och en swap) så väljer man 3 här.

4. I nästa steg väljer man startcylinder. Här finns ett default-värde man kan anväda så här kan man trycka på retur.

5. Slutcylinder bestämmer hur stor partitionen ska vara. Man börjar med ett plus-tecken följt av ett nummer på hur stor paritionen ska vara. I slutet lägger man till ett stort M. Det räcker att paritionen cluster_metadata är runt 512 Mb så då skrivs +512M.

6. För att skriva partitionen till partitiontabellen skriver man ett w. Man kan först kolla med ett p om partitioneringen ser bra ut.

Den andra partitionen, för själva lagringen, skapar man på liknande sätt. I vårt fall lägger vi LVM-paritionen på samma disk (sda). Om ni har en annan så väljer ni den. Har ni redan konfigurerat upp en LVM-partition så kan ni skippa stegen nedan.

1. Skriv först fdisk /dev/<hdd> för att starta konfigureringen, där <hdd> är den disk

som LVM-partitionen ska ligga på.

2. Skriv n följt av ett p för att skapa en primär partition.

3. Beroende på hur många paritioner ni har, så kan det skilja sig lite här. Vi skapar LVM-partitionen på samma hårddisk vilket gör att vi har tre partitioner sedan innan. I detta steg väljer vi 4.

4. Startcylinder kan man bara trycka vidare med retur.

5. Slutcylinder anger man på samma sätt: +sizeM där size är storleken och M står för megabyte. I denna experiment valdes storleken 50000 Mb, då skriver man +50000M.

6. För att skapa denna nya partition som en LVM, skriver ni först ett t.

7. En lista på olika filsystem får man upp med ett L. I den listan ser man att LVM har 8e. Här skrivs alltså 8e.

8. För att skriva partitionen till partitiontabellen skriver man ett w. Man kan först kolla med ett p om partitioneringen ser bra ut.

De nya partitionerna gäller när man först startar om datorn.

Nätverkskonfigurering

Under denna sektion kommer NIC-bonding att sättas upp samt den statiska IP-adressen som kommer användas till failover. Det kommer även se till att noderna kan prata med varandra utan att behöva ange lösenord varje gång.

1. Gå in till webbgränssnittet hos den första noden (den som ska bli primär). Gå sedan till System-tabben och skrolla ner till nätverksgränssnitten. Välj Configure bredvid den nätverksport som är kopplat mellan noderna (i vårt fall eth1). Välj att det ska vara en statisk IP-adress och gå vidare. Nu väljs en IP-adress som den andra ska kunna hitta, till exempel 192.168.10.1.

2. För de andra två nätverksgränssnitten ska dessa användas till NIC-bonding. Under ”Network Interface Configuration” så finns länken ”Create bonded interface”. Trycker ni på den kommer ni till en sida där man kan välja vilka som ska agera som ett nätverksgränssnitt. Går man sedan vidare väljs vilket IP samt netmask gränssnittet ska ha. Sen ska ni välja vilken typ av NIC-bonding ni vill ha. Välj ”Balance XOR” så kommer de båda nätverksgränssnitten agera som en lastbalanserare. Efter man tryckt på Confirm så behöver man starta om nätverkstjänsten i Openfiler. För att göra detta skrivs: service network restart. När nätverket startats om så kan ni komma in på den nya adressen som valts.

3. Sen för att underlätta saker kan man döpa om värdnamnet hos noderna. Att ändra värdnamnet gör man under Hostname. Här valdes värdnamnen node1 till den primära noden.

4. Logga sedan in hos den andra nodens webbgränssnitt och gör samma saker som i steg 1-3. Till den statiska IP-adressen väljs till exempel 192.168.10.2. Värdnamnet valdes till node2. För skapandet av NIC-bonding sker det på samma sätt. Först väljer man de två nätverksgränssnitt som ska agera som ett, sedan väljs IP-adressen för bondingen. För att den nya IP-adressen ska fungera så startar man om nätverkstjänsten: service network restart.

5. För att noderna ska hitta varandra utan att ange IP-adressen så modifieras filen /etc/hosts: nano /etc/hosts Längst ner i filen lägger man till den andra nodens IP-nummer och värdnamn. Hos node1 lägger man till: 192.168.10.2 node2 och hos node2: 192.168.10.1 node1. Om ni valt annan IP-adress eller värdnamn skriver ni in dem istället.

6. För att tillåta att noderna ska kunna prata med varandra utan att behöva lösenord så används en gemensam SSH-nyckel. För att generera en SSH nyckel skriver man kommandot: ssh-keygen -t dsa

7. På frågorna så trycker man bara på ENTER då det inte behövs sätta något lösenord. Den genererade nyckeln läggs i filen id_dsa.pub som ligger i mappen /.ssh/. Denna fil ska bytas ut med varandra. Detta görs med kommandot scp. För att flytta filen från node1 till node2 skrivs: scp /.ssh/id_dsa.pub root@node2 /.ssh/authorized_keys2 Detta görs när man är inloggad hos node1 via ssh. Om ett annat värdnamn valts så byter ni ut den.

8. Samma sak gör ni hos node2: scp /.ssh/id_dsa.pub root@node1 /.ssh/authorized_keys2

Konfigurering av DRBD

Nu är det dags att konfigurera DRBD. Det är DRBD som sköter speglingen mellan noderna.

1. Först skapas en backup på filen drbd.conf: mv /etc/drbd.conf /etc/drbd.conf.org

2. Skapa en ny drbd.conf: nano /etc/drbd.conf.

3. I filen kopieras innehållet som finns i Appendix B in i filen.

4. I vårt fall ligger partitionen cluster_metadata i /dev/sda3 och LVM-partitionen i /dev/sda4. Dessa kan skilja sig ifrån i erat fall, så dessa måste ändras till rätt enhet i konfigurationsfilen.

5. Även IP-adresser i drbd.conf ska ändras ifall ni valt annorlunda.

6. I filen står också värdnamnen node1 och node2, dessa ska också ändras ifall ni har något annat värdnamn.

7. Sedan sparar man filen (Ctrl+x i nano).

8. Båda noderna ska ha samma fil. Kopiera över filen med detta kommando: scp /etc/drbd.conf root@node2:/etc/drbd.conf

9. Nästa steg är att initiera metadata på /dev/drbd0 och /dev/drbd1. Skriv drbdadm create-md cluster_metadata och sedan drbdadm create-md vg0drbd. Detta ska göras hos båda noderna.

10. Ibland behöver man nollställa filsystemen. Det brukar blir det när man skapar vg0drbd. För att fixa detta skriver man dd if=/dev/zero of=/dev/sda4 där /dev/sda4 är den partition som LVM ligger i. Tänk på att data kommer att försvinna. Använd rätt enhet för erat system. Det räcker att man kör detta kommandot i några sekunder. För att avbryta trycker man på Ctrl+c.

11. När drbd0 och drbd1 är skapade så startar man tjänsten DRBD hos båda noderna: service drbd start

12. Tjänsten ska startas hos båda noderna. När det är igång kan man kolla statusen:

service drbd status. Utskriften ska se ut som i Figur x.x.

13. Om det står Connected Secondary/Secondary Inconsistent/Inconsistent hos både cluster_metadata och vg0drbd så är det dags att sätta diskarna till primära hos node1 (eller den nod som ska vara primär). Skriv drbdsetup /dev/ drbd0 primary -o och drbdsetup /dev/drbd1 primary -o hos endast den primära noden.

14. Kollar ni statusen igen (service drbd status) så borde det se ut något likande som i Figur x.x. Man kan se att synkroniseringen har påbörjat. Själva synkroniseringen kan ta en bra stund beroende på hur stor partitionen är.

15. För att låta DRBD startas upp under uppstarten skrivs detta kommando hos båda noderna: chkconfig –level 2345 drbd on

16. Till sist skriver ni detta kommando: mkfs.ext3 /dev/drbd0

Konfigurering av LVM

För konfigurering av LVM-partitionen, editera filen lvm.conf:

1. Öppna filen: nano /etc/lvm/lvm.conf.

2. Hitta raden där det står: filter = [”a/.*/”]. Ersätt denna rad med filter = [”r|/dev/sda4|”]. Ändra sda4 till den LVM-partition ni har ifall det inte är samma. Samma sak gör man även hos den andra noden.

3. Sedan skapar man en fysisk LVM volym hos node1: pvcreate /dev/drbd1.

Konfigurering av Heartbeat

Heartbeat är den tjänst som genomför en failover när en nod havererar.

1. Först ska man skapa filen authkeys hos båda noderna: nano /etc/ha.d/authkeys

2. I den skriver ni: auth 2
2 crc

3. Nästa fil som ska skapas är ha.cf (även här hos båda noderna) nano /etc/ha.d/ha.cf

4. I denna fil ska ni skriva det som står i Appendix C. Det som står vid bcast är den nätverksport som används mellan de två noderna. Det är den porten som heartbeat kommer skicka ut signaler för att känna av om den andra noden lever. Se också över de två nedersta raderna så att värdnamnen stämmer överens med det ni valt. Kom ihåg att detta ska skapas hos den andra noden också. Se även till där att det är rätt nätverksport som är skrivet i filen.

5. Sedan skriver ni in chkconfig –level 2345 heartbeat on hos båda noderna så att tjänsten Heartbeat startas automatiskt vid uppstart.

Konfigurering av Openfiler

Nu ska det kopieras över alla konfigurationsfiler till cluster_meta-partitionen så att båda noderna är synkroniserade med varandra och att alla tjänster är tillgängligt efter en failover. När ni sedan flyttat filerna till partitionen så skapar ni en symbolisk länk tillbaka till ursprungsplatsen.

1. Det är några kommandon som ska utföras. Dessa ska endast utföras hos node1:
mkdir /cluster_metadata
mount /dev/drbd0 /cluster_metadata
mv /opt/openfiler/ /opt/openfiler.local
mkdir /cluster_metadata/opt
cp -a /opt/openfiler.local /cluster_metadata/opt/openfiler
ln -s /cluster_metadata/opt/openfiler /opt/openfiler
rm /cluster_metadata/opt/openfiler/sbin/openfiler
ln -s /usr/sbin/httpd /cluster_metadata/opt/openfiler/sbin/openfiler
rm /cluster_metadata/opt/openfiler/etc/rsync.xml
ln -s /opt/openfiler.local/etc/rsync.xml /cluster_metadata/opt/openfiler/etc/
mkdir -p /cluster_metadata/etc/httpd/conf.d

2. Filen /opt/openfiler.local/etc/rsync.xml ska sedan redigeras (hos node1) och innehållet ska ändras till det som står i Appendix D. nano /opt/openfiler.local/etc/rsync.xml Under remote hostname så ska IP-adressen till node2 stå.

3. Nu över till node2. Utför dessa kommandon:
mkdir /cluster_metadata
mv /opt/openfiler/ /opt/openfiler.local
ln -s /cluster_metadata/opt/openfiler /opt/openfiler

4. Filen /opt/openfiler.local/etc/rsync.xml ska även här redigeras. Innehållet ska vara samma fast remote hostname ska vara IP-adressen till node1. Vad som ska stå i filen finns i Appendix D.

5. För att tjänsten heartbeat ska veta vad som ska utföras vid en failover så behövs en fil som talar om det. Detta görs endast hos node1. Ändra i filen cluster.xml: nano /cluster_metadata/opt/openfiler/etc/cluster.xml

6. I den filen skrivs det som står i Appendix E. Under nodename value skrivs det in den nod som ska vara primär (node1 i detta fallet). Här skriver man också in vilket IP-adress som ska användas för att komma åt webbgränssnittet. Denna IP-adress kommer fungera även om en nod skulle haverera.

Konfigurering av Samba och NFS

För att fixa stöd för Samba och NFS gör ni på liknande sätt när ni förde över Openfilers konfigurationsfiler. Innan man börjar konfigurera NFS och Samba ska man stänga av några tjänster.

1. Utför dessa hos båda noderna:
service nfslock stop
service nfs stop
service rpcidmapd stop
umount -a -t rpc_pipefs

2. Mappen /var/lib/nfs kommer att flyttas snart, och då kan man inte montera rpc_pipefs

igen som vanligtvis ligger i /var/lib/nfs/rpc_pipefs. Då skapar ni en ny mapp för rpc_pipefs: mkdir /var/lib/rpc_pipefs Mappen skapas hos båda noderna.

3. Sedan ska två filer modifieras. När man nu försöker montera rpc_pipefs igen så kommer den försöka montera den till /var/lib/nfs/rpc_pipefs. Denna sökväg kommer att försvinna, så sökvägen ska ändras i dessa två filer: nano/etc/modprobe.conf och /etc/idmapd

4. Börja med den första filen. I nano söker man med Ctrl+w. Sök efter strängen "/var/lib/nfs/rpc_pipefs" och ta sedan bort "/nfs" så att det endast står "/var/lib/rpc_pipefs". Gör samma sak med den andra filen. Detta ska göras hos båda noderna.

5.När man stoppat tjänsterna så utför man dessa kommandon hos node1:
mkdir /cluster_metadata/etc
mv /etc/samba/ /cluster_metadata/etc/
ln -s /cluster_metadata/etc/samba/ /etc/samba
mkdir -p /cluster_etadata/var/spool
mv /var/spool/samba/ /cluster_metadata/var/spool/
ln -s /cluster_metadata/var/spool/samba/ /var/spool/samba
mkdir -p /cluster_metadata/var/lib
mv /var/lib/nfs/ /cluster_metadata/var/lib/
ln -s /cluster_metadata/var/lib/nfs/ /var/lib/nfs
mv /etc/exports /cluster_metadata/etc/
ln -s /cluster_metadata/etc/exports /etc/exports

6. Utför sedan dessa kommandon hos node2:
rm -rf /etc/samba/
ln -s /cluster_metadata/etc/samba/ /etc/samba

rm -rf /var/spool/samba/

ln -s /cluster_metadata/var/spool/samba/ /var/spool/samba

rm -rf /var/lib/nfs/

ln -s /cluster_metadata/var/lib/nfs/ /var/lib/nfs

rm -rf /etc/exports

ln -s /cluster_metadata/etc/exports /etc/exports

7. Till sist startar man tjänsterna igen hos båda noderna: mount -t rpc_pipefs sunrpc /var/lib/rpc_pipefs

service rpcidmapd start

Konfigurering av iSCSI

För att fixa så att iSCSI fungerar efter en failover så behövs ytterligare utföranden av kommandon. Det är filer som behöver flyttas till /cluster_metadata samt skapa en symbolisk länk tillbaka.

1. Först utför man dessa kommandon hos node1: mv /etc/ietd.conf /cluster_metadata/etc/
ln -s /cluster_metadata/etc/ietd.conf /etc/ietd.conf
mv /etc/initiators.allow /cluster_metadata/etc/
ln -s /cluster_metadata/etc/initiators.allow /etc/initiators.allow
mv /etc/initiators.deny /cluster_metadata/etc/
ln -s /cluster_metadata/etc/initiators.deny /etc/initiators.deny

2. Sedan dessa hos node2: rm /etc/ietd.conf
ln -s /cluster_metadata/etc/ietd.conf /etc/ietd.conf
rm /etc/initiators.allow
ln -s /cluster_metadata/etc/initiators.allow /etc/initiators.allow
rm /etc/initiators.deny
ln -s /cluster_metadata/etc/initiators.deny /etc/initiators.deny

Sista konfigureringen

Nu är det snart färdigt. Det är några småsaker kvar som ska göras innan man startar om noderna.

1. Det behövs en volymgrupp för /dev/drbd1: vgcreate vg0drbd /dev/drbd1. Här gäller

det att utesluta ”_” i namnet om man har tänkt använda windows för att komma åt iSCSI-enheter.
2. Utför dessa kommandon hos node1 (om man kör 32-bits så ändra lib64 till lib): rm /opt/openfiler/etc/httpd/modules
ln -s /usr/lib64/httpd/modules /opt/openfiler/etc/httpd/modules

3. Det behövs en volym för att allt ska fungera. Denna volym kan man sedan ta bort, men ni måste då skapa en ny, annars kan det sluta fungera. lvcreate -L 400M -n filer vg0drbd

4. Nu när allt är konfigurerat så startar ni om tjänsten Openfiler hos node1: service openfiler restart

5. När detta är gjort så ska ni tvinga filen haresources att skrivas. Detta görs via webbgränssnittet (hos node1) genom att ni aktiverar en tjänst. Gå in under tabben Services och aktivera tjänsten NFS och iSCSI.

6. När filen haresources är skapad så skickar ni filen över till node2: scp /etc/ha.d/haresources root@node2:/etc/ha.d/haresources.

7. Nu ska allt vara färdigt. Starta först om node1, efter det node2. Om allt går rätt till så ska noderna hitta varandra under uppstart. Sedan så ska man kunna komma åt webbgränssnittet via den HA-adress ni valt.

Problem

I denna sektion kommer lite problem som uppkom och hur vi löste dessa.

Första uppstarten

När vi startade upp båda noderna efter vi hade konfigurerat klart så blev den andra nodens tillstånd diskless. Detta var endast på LVM-partitionen. Vi löste det med att starta om node2 först och sedan node1 efter en minut, så att node2 fick stå och vänta på node1. Ett problem som vi inte riktigt vet varför det uppstår, men det löser sig om vi startar om node2 före node1.

Blank sida när man gick in på Volumes-tabben

Ett annat problem som vi fick i början var att när vi gick in på tabben Volumes så hände det inget. Det kom bara en blank sida. Detta visade sig att det var diskettstationen som spökade. Vi gick in i BIOS och avaktiverade diskettstationen. Efter det så fungerade det.

Problem med att lägga till nya iSCSI- och NFS-volymer

När man lagt till nya volymer, och en failover sker, så kommer inte node2 att hitta de nya volymer. Anledningen är vi inte riktigt säker på men vi misstänker att filen haresources inte skrivits med de nya informationen. När man väl är i node2, och tvingar att skriva filen haresources (genom att avaktivera och aktivera NFS- eller iSCSI-tjänsten), så verkar det funka. Så fort man lägger till en ny volym hos node1, så bör man kopiera över filen /etc/ha.d/haresources till node2, men det är egentligen inget vi rekommenderar.

RPC: fragment too large

Ett annat problem uppstod när en failover hade skett och den sekundära servern hade tagit över medan vi överförde filer till en NFS-enhet. Då kom detta felmeddelande upp:

RPC: fragment too large

Vad detta berodde på är svårt att säga eftersom någon bekräftad lösning aldrig hittades. Detta hade säkerligen med klienten att göra eftersom när man bytte klient så fungerade det. Samma fel uppstod senare också på den nya klienten vilket var mycket märkligt.

Split-Bain

När något blir fel i anslutningen mellan den sekundära servern och den primära servern kan en så kallad split-brain uppstå. Detta betyder att båda servrarna tror att den andra är död och båda servrarna tar över resurser som de egentligen inte ska göra och blir primära. När ett sådant fel uppstår får man detta meddelande:

Split-Brain detected, dropping connection!

Felet löses manuellt genom att skriva:

drbdadm secondary resource drbdadm – –discard-my-data connect resource

på den nod som är sekundär. På den nod som är primär skrivs:

drbdadm connect resource

Lång tid för noderna att hitta varandra

Ett annat konstigt problem som uppkom var att båda servrarna hade svårt att finna varandra och synkningen tog lång tid efter en failover. Detta kan möjligen bero på fel i hårdvaran. Efter ett byte av nätverksport för den korsade nätverkskabeln, som binder ihop de två servrarna, så fungerade allt bra igen.

Trasigt nätverkskort

Andra hårdvarufel uppstod. Ett nätverkskort klagade på MAC-adressen och vi kunde inte få någon kontakt med den. Efter ett byte av nätverkskortet så löstes detta problem.

B Appendix B - drbd.conf

I denna del finns den stora konfigurationsfilen som DRBD använder sig av. Allt i denna fil som är i fet stil är värden som är specifikt för vår testmiljö och kan skilja sig från eran miljö. Ändra dessa värden så att de stämmer överens med eran miljö.

```
global {

# minor-count 64;
# dialog-refresh 5;
# disable-ip-verification;
usage-count ask;
}

common {
syncer { rate 100M; }
}

resource cluster_metadata {

protocol C;
handlers {

pri-on-incon-degr "echo O > /proc/sysrq-trigger ; halt -f";
pri-lost-after-sb "echo O > /proc/sysrq-trigger ; halt -f";
local-io-error "echo O > /proc/sysrq-trigger ; halt -f";
# outdate-peer "/usr/sbin/drbd-peer-outdater";
```

}

startup {
# wfc-timeout 0;
degr-wfc-timeout 120;
}

disk {
on-io-error detach;

}

net {
after-sb-0pri disconnect;
after-sb-1pri disconnect;
after-sb-2pri disconnect;
rr-conflict disconnect;
}
syncer {
# rate 10M;
# after "r2";
al-extents 257;
}

on **node1** {
device /dev/drbd0;
disk /dev/**sda3**; #cluster_metadata hos den primärnoden

address **192.168.10.1:7788**; #Primära nodens IP-adress
meta-disk internal;
}

on **node2** {
device /dev/drbd0;
disk /dev/**sda3**; #cluster_metadata hos den sekundärnoden
address **192.168.10.2:7788**; #Sekundärnodens IP-adress
meta-disk internal;
}
}

resource vg0drbd {

protocol C;

startup {
wfc-timeout 0;
degr-wfc-timeout 120;
}

disk {
on-io-error detach;
}

net {
# timeout 60;

# connect-int 10;
# ping-int 10;
# max-buffers 2048;
# max-epoch-size 2048;
}

syncer {
after "cluster_metadata";
}

on **node1** {
device /dev/drbd1;
disk /dev/**sda4**; #vg0drbd (LVM)
address **192.168.10.2:7789**; #Primärnodens IP-adress
meta-disk internal;
}

on **node2** {
device /dev/drbd1;
disk /dev/**sda4**; #vg0drbd (LVM)
address **192.168.10.2:7789**; #Sekundärnodens IP-adress
meta-disk internal;
}

}

C Appendix C - ha.cf

Det som står i fet stil är specifikt för vår testmiljö och kan skilja sig från eran miljö. Ändra dessa värden så att de stämmer överens med den miljö ni har.

debugfile /var/log/ha-debug
logfile /var/log/ha-log
logfacility local0
bcast **eth1**
keepalive 5
warntime 10
deadtime 120
initdead 120
udpport 694
auto_failback off
node **node1**
node **node1**

D Appendix D - rsync.xml

Det som står i fet stil är specifikt för vår testmiljö och kan skilja sig från eran miljö. Ändra dessa värden så att de stämmer överens med den miljö ni har.

Primärnodens rsync.xml

<?xml version="1.0" ?>

<rsync>

<remote hostname="**192.168.10.2**" /> ## IP-adressen till sekundärnoden

<item path="/etc/ha.d/haresources" />

<item path="/etc/ha.d/ha.cf" />

<item path="/etc/ldap.conf" />

<item path="/etc/openldap/ldap.conf" />

<item path="/etc/ldap.secret" />

<item path="/etc/nsswitch.conf" />

<item path="/etc/krb5.conf" />

</rsync>

Sekundärnodens rsync.xml

<?xml version="1.0" ?>

<rsync>

<remote hostname="**192.168.10.1**" /> ## IP-adressen till primärnoden

<item path="/etc/ha.d/haresources" />

<item path="/etc/ha.d/ha.cf" />

<item path="/etc/ldap.conf" />

<item path="/etc/openldap/ldap.conf" />

<item path="/etc/ldap.secret" />

<item path="/etc/nsswitch.conf" />
<item path="/etc/krb5.conf" />
</rsync>

E Appendix E - cluster.xml

Det som står i fet stil är specifikt för vår testmiljö och kan skilja sig från eran miljö. Ändra dessa värden så att de stämmer överens med den miljö ni har.

<?xml version=”1.0” ?>

<cluster>

<clustering state=”on” />

<nodename value=”**node1**” /> ##Den primära nodens värdnman

<resource value=”MailTo::it@company.com::ClusterFailover” />

<resource value=”IPaddr::**192.168.0.100/24**” /> ##HA-adressen

<resource value=”drbddisk::” />

<resource value=”LVM::vg0drbd” />

<resource value=”Filesystem::/dev/drbd0::/cluster_metadata::ext3::defaults,noatime” />

<resource value=”MakeMounts” />

</cluster>