EDA 390 - Datakommunikation och Distribuerade System
Distribuerade filsystem

Thomas Hellström
heltom@dtek.chalmers.se

1 Introduktion

Denna rapport är ett projekt i kursen Datakommunikation och Distribuerade System (EDA 390) och tar upp distribuerade filsystem och jämför dom i olika aspekter. Fokus är på säkerhet, skalbarhet och prestanda samt lösningar på vanliga problem som ett distribuerat filsystem brukar stöta på.

1.1 Notis om källor

Information om skalbarhet var synnerligen svårt att komma över, det mesta man kan finna på Internet är pressmeddelande och marketing-avdelnings-text utan någon som helst substans. Alla tre system skall vara extremt skalbara om man ska tro deras utvecklare, men någon konkret information och empiriska tester om hur många användare ett system faktiskt klarar av är omöjligt att finna. Därför är många av slutsatserna angående skalbarhet just slutsatser av författaren baserat på den information som finns tillgänglig. Prestanda-jämförelser lider av samma problem, mycket av den informationen som går att finna på Internet är andrahands-information i nyhetsgrupper och mailinglistor från diverse obskyra linux-tidningar och liknande. Mitt mål från början var att personligen kunna göra ett prestandatest, men på grund av tidsbrist kunde detta inte genomföras. Information om de olika protokollens säkerhetslösningar fanns det däremot i mängder, mycket var dock i kritiska texter där författaren hade en uppenbar förutfattad mening om systemet i fråga.

1.2 Korta definitioner

- Låsa filer - Få ensamrätt till en fil för att undvika en konflikt
- SSH - Secure Shell. Ett mycket säkert protkoll för remote login
- VPN - Virtual Private Network. Ett krypterat nätverk i ett annat nätverk
- Atomärt - Omedelbart. En operation som inte kan avbrytas innan den är klar
- DES - Väl använd krypteringsalgoritm
- RPC - Remote Procedure Call. Används främst för att exekvera kod på en server och få resultatet på en klient
- Kerberos - System för autentisering
- UID - User ID. Unikt ID för en användare
- GID - Group ID. Unikt ID för en grupp

1.3 Andra system

Det är inte helt trivialt att definiera vad som är ett distribuerat filsystem, vad som är distribuerad fillagring och vad som är fildelning. Skillnaderna är mycket diffusa stundtals, och jag har varit strikt i vilka system som rapporten går in i djupare. Ett distribuerat filsystem skall först och främst lagra data på en eller flera noder som är skilda från användarna rent fysiskt. Sedan skall det vara

transparent, dvs intergrera snyggt med de lokala filsystemen, en användare skall inte se skillnad på en lokal fil och en fil lagrad på det distribuerade filsystemet. Detta ställer en del krav på accesstid och liknande, vilket gör att flera system faller bort. Nedan följer några andra system som av olika anledningar inte tas upp i rapporten.

1.3.1 Coda

Coda har utvecklats sen 1987 vid Carnegie Mellon University och är ett derivat av en äldre version av AFS (Andrew File System). Det används inte i nån större utsträckning och är i grund och botten AFS med några nya (men högst användbara) funktioner tillförda.

1.3.2 Google File System

Google File System (GFS) är, som namnet indikerar, utvecklat av Google och är optimerat för deras något speciella ändamål. Systemet är byggt för extremt stora filer som aldrig raderas och gör skillnad på master-servers och chuckservers som innehåller data. Det används i stort sett bara av google, men är snyggt integrerat med linux, men det funkar då enbart med gmail som är googles webmail-system.

1.3.3 BitTorrent

BitTorrent är ett mycket framgångsrikt peer-to-peer-system för att distribuera filer (ofta av semilegalt ursprung). Vissa webbsidor inkluderar bittorrent som ett distribuerat filsystem, men det finns ingen mjukvara för att snyggt och enkelt intergrera en torrent som ett filsystem samt att access till filer kan ta flera timmar om det vill sig illa.

1.3.4 Freenet

Freenet är ett rent peer-to-peer fillagringssystem. Det är fokuserat på anonymitet snarare än prestanda. Det är dock i gränslandet till att vara ett distribuerat filsystem. Systemet fungerar så att en användare laddar upp en fil till nätverket där den sprids ut på olika noder och får ett unikt ID (med hjälp av en hashalgoritm). Detta unika ID används sedan för att andra användare skall kunna få tillgång till filen. Systemet är näst intill helt anonymt, helt distribuerat men olyckligtvis är det extremt långsamt. Det intergrerar inte smärtfritt med det lokala filsystemet vilket är anledningen till att denna rapport inte tar upp det i djupare detalj. En fil som laddas upp i freenet är mer eller mindre garanterad att finnas på att stort antal datorer, vilket skulle kunna göra det väldigt attraktivt ur ren filsystemsynvinkel men de nuvarande hastighetsbegränsningarna gör det nästan helt obrukbart som ett distribuerat filsystem.

1.4 De tre stora

SMB, NFS och AFS är förmodligen de tre största distribuerade filsystemen som finns tillgänliga. Alla tre har stöd i de flesta stora operativsystem och används i stor utsträckning på alla platformar. De är väl dokumenterade och figurerar i flera test i olika tidningar och rapporeter. Resten av rapporten kommer därför att uteslutande fokusera på dessa tre.

1.5 Gemensamma problem, jämförelser

Systemen kommer att främst jämföras på tre olika punkter: säkerhet, skalbarhet samt prestanda. Alla distribuerade system stöter även på problem med att hålla samma version av en fil åt flera olika klienter. Detta och andra mer intressanta funktioner och lösningar samt problem kommer att tas upp för alla system.

2 Server Message Block

2.1 Historik

SMB var från början utvecklat av IBM för att ge DOS stöd för nätverksfilsystem. Det vidareutvecklades av Microsoft och körde från början över NetBIOS-protkollet istället för UDP/TCP, men detta ändrades med windows 2000. SMB är egentligen inget rent distribuerat filsystem, det inkluderar skrivardelning, IPC (Inter process communication), och annat. Filsystemet kallas för en ”service” som körs över protokollet SMB. Broadcast-systemet när man smidigt kan hitta andra datorer i nätverket som många windowsanvändare är bekanta med ingår alltså inte i SMB, utan är en funktion hos NetBIOS-protokollet. Filsystem-delen av SMB kallas ibland CIFS (Common Internet File System) beroende på vilken version man hänvisar till.[8]

2.2 Säkerhet

Säkerheten i SMB har varit mycket diskuterad på senare år. De flesta stora internetmaskar på senare år har angripit Microsoft Windows-maskiner och då speciellt SMB:s IPC-funktioner, funktioner som är delvis relaterade till filsystemet och är därför värda att ta upp ur en säkerhetsaspekt. Just Microsofts implementation av SMB är utan tvekan den mest dominerande på marknaden och kanske är det just därför deras platform är den mest attackerade. Deras källkod är stängd och den enda officiella specifikationen av CIFS är flera år gammal och i stort sett obrukbar, vilket har gjort att den andra stora implementationen av SMB, Samba, har byggts till stor del genom analys av protkollet för att i efterhand bygga upp en specifikation som kan användas. Allt detta gör att SMB-kommunikation mellan olika platformar är till stor del byggt av gissningar och analys, vilket fungerar förvånansvärt bra. SMB använder antingen en svag variant av DES eller en 40-bitars kryptering som tillhandahålls av NT, båda anses vara osäkra, vilket gör autentisering intressant. Nyare versioner av windows har en påslagen firewall som standard vilket effektivt filtrerar bort all SMB-trafik mot Internet, och gör att situationen är något bättre än tidigare, men grundsystemet, med sin dåliga dokumentation, stängda källkod och hopplöst många RPC-anrop gör SMB till ett mycket osäkert system i sitt grundutförande. Många av dessa problem kan givetvis lösas på samma sätt som för NFS, genom att använda en tredje parts mjukvara för kryptering, autentisering och så vidare. SMB-protkollet har dock stöd för fler krypteringsalgoritmer än de tidigare nämnda, men dessa fungerar enbart med Samba och inte med Microsofts SMB-implementation. Nyare versioner av SMB skall även stödja Kerberos.[5][6]

2.3 Skalbarhet

Precis som NFS så är systemet uppbyggt enligt server-klient-modellen med en server och flera klienter. En fil kan enbart ligga på en disk och en server vilket gör att själva mediumet som filen ligger på begränsar hur många klienter som kan ansluta och använda sig av den samtidigt. CIFS har något som kallas DFS (Distributed File System), vilket låter synnerligen intressant, men är enbart ett sätt att bygga virtuella kataloger som består av flera mindre SMB-utdelningar. Detta löser egentligen inte skalbarhetsproblemet, men är en förbättring jämtemot NFS. [5]

2.4 Diverse algoritmer, lösningar och problem

Då Microsofts implementation av SMB är stängd och synnerligen dåligt dokumenterad så har det uppstått ”dialekter” av protokollet över platformar och det är inte helt konsekvent hur olika anrop inom SMB skall hanteras. All information nedan är baserad på den öppna varianten Samba.

2.4.1 Filattribut och filnamn

CIFS/SMB var från början tänkt att fungera ihop med DOS, vilket har ett väldigt spartanskt sätt att hantera filnamn och filattribut. Filnamn är case-insensitive, dvs de gör ingen skillnad på stora och små bokstäver, filnamn är uppbyggda av 8 plus 3 bokstäver, de har inga grupper eller användare associerade med sig och filattributen är begränsade till, arkivfil, dold fil, systemfil och katalog. UNIX-filsystem, NTFS och FAT32 har alla mer avancerade sätt att hantera namn och attribut och problem uppstår ibland när olika platformar skall kommunicera med varandra. En SMB-server förväntas därmed kunna omvandla sina filnamn på flera olika sätt för att kunna tillfredsställa alla sorters klienter. [6]

2.4.2 Konflikter

SMB har stöd för atomiskt låsa både hela filer och delar av filer. Det är inte stateless på samma sätt som NFS och har därför möjlighet att veta vilken användare som har vilka filer, samt bättre möjlighet att återhämta sig från olika typer av crasher.

2.5 Prestanda

Flertalet tester i olika tidningar (se referenser) kommer fram till samma slutsats; SMB är långsamt. Prestandan verkar variera mellan 75NFS beroende på vilken operation man genomför (läsa, skriva, synkronisera, etc). Windows-implementationen av SMB är mycket sämre än Samba som är något sämre än NFS. Detta är givetvis en stor last till SMB, men det är föga förvånande när man läser protokollspecifikationen. Det är lager på lager av funktioner och till synes onödiga handskakningar och systemanrop vilket gör att varje överförd fil kräver stora mängder (till synes onödig) kontrolldata. [5][7]

3 Network File System

3.1 Historik

NFS var ett av de tidigaste system och utvecklades under 80-talet av Sun Microsystems (andra konkurrerande system var Network Computing System från Apollo och AT&Ts Remote File System).

Fördelen med NFS är främst dess ålder. Det är robust, välbeprövat och väldokumenterat. Det är även utan tvekan det minst komplicerade av de tre protokollen, både för en administratör och en utvecklare.[4]

3.2 Säkerhet

Enkelheten kommer dock inte gratis; standardversionen av NFS 3 lämnar en hel del att önska i termer av säkerhet. RFC1813 specifierar tydligt att DES-kryptering skall gå att använda som autenticering, men detta verkar stödjas av enbart en handfull implementationer. Istället läggs ansvaret på IP-baserad säkerhet och yttre säkerhetssystem. Med IP-baserad säkerhet menas alltså att tillgång till filsystemet bestäms av vilken IP-address en klient har, vilket givetvis är en väldigt svag form av säkerhet. NFS fungerar utmärkt genom VPN-tunnlar och SSH-tunnlar, vilket ger det en ganska god säkerhet med certifikat och allt som hör VPN och SSH till.

3.3 Skalbarhet

Eftersom NFS är designat för att vara en server, flera klienter så skalar den tämligen urselt. Enklare scenarion, exempelvis en gemensam punkt för alla hemkonton för användare och liknande duger NFS utmärkt till dock.

3.4 Diverse algoritmer, lösningar och problem

3.4.1 Konflikter

NFS 2 och 3 stödjer något förvånande inte låsning av filer i sitt grundutförande, utan litar ett extra protokoll som heter NLM (Network Lock Manager). NLM sköter alla fillåsningar och har även stöd för att återhämta sig vid olika typer av crasher (både server och klient).

3.4.2 Tillstånd

NFS är stateless (utan tillstånd) och har därför ingen möjlighet att veta vilken användare som har öppnat vilka filer. Detta är även relaterat till dess undermåliga sätt att låsa filer. Att ha låsta filer är att ha ett tillstånd och därför behövs det annan mjukvara för att lösa dessa problem.

3.4.3 Tidsproblem

NFS har ingen synkronisering av tid mellan server och klient, vilket kan ge intressanta bieffekter med filer som skapas i framtiden och liknande. Även detta brukar man generellt kringgå genom att

nitiskt synkronisera klockan på klient och server mot en tredje part, förslagsvis via NTP (Network Time Protocol).

3.5 Prestanda

Enligt ett något ovetenskapligt test så är NFS något bättre än SMB när det gäller rå hastighet. Man testade genom att skapa filer av olika storlekar och gjorde överföringar mellan olika platformar och NFS presterade bättre än SMB med ungefär 25%, en siffra som nog skall tas med en nypa salt. [2][3][1]

4 Andrew File System

4.1 Historik

Andrew File System (AFS) utvecklas vid Carnegie Mellon University som en del av det större Andrew-projektet, som är ett samarbete mellan universitet och IBM. Det påbörjades redan 1985 men AFS flyttades över till företaget Transarc 1988. Idag finns det flera fristående implementationer av protkollet, bland annat Arla och OpenAFS. AFS var och är från början tänkt som en del i ett distribuerat nätverk och har därmed en hel del bra lösningar på de olika problemen som uppstår.[11]

4.2 Säkerhet

AFS förlitar sig till Kerberos för autentisering och certifikat, vilket gör det mycket säkert. Det använder sig även av en mer raffinerad variant än det vanliga UNIX-systemet för filattribut, nämligen sk Access Control Lists (ACL). ACL ger en finare upplösning av rättigheter på katalogstrukturer och filer än standard-unix, varje fil kan ha olika rättigheter för olika användare exempelvis istället än bara ett UID och GID som vanligt.

4.2.1 Kerberos

Då Kerberos är en sån fundamental del av AFS-systemet så bör det gås igenom i lite djupare detalj. Kerberos utvecklades vid MIT och är ett protokol främst för säker autentisering och har ett biljettsystem (eng. tickets) för att ge rättigheter till filer och tjänster inom ett nätverk. Fördelen med detta systemet är att all kryptering sker med hjälp av en dedikerad tredje part istället för att specifieras av själva filsystemet, vilket gör att en uppgradering av Kerberos inte behöver innebära en uppgradering av AFS. Kerberos gör även att lösenord inte skickas varken krypterade eller i klartext över nätverket och anses vara ett av de säkraste systemen som finns för autentisering.

4.3 Skalbarhet

Till skillnad från NFS och SMB kan AFS spara samma fil på flera olika noder (med vissa begränsningar), vilket gör att filsystemet är ytterst skalbart och kan hantera över 50 000 klienter. Det har även stöd för virtuella filsystem, eller ”celler” där flera utdelade filer och kataloger ger intrycket av att vara en enda koncis enhet, på samma sätt som SMB. OpenAFS rekommenderar max 200 klienter per server, vilket är bra mycket mer än vad SMB och NFS anses klara med normal hårdvara.

4.4 Diverse algoritmer, lösningar och problem

4.4.1 Kvotering

AFS har inbyggt kvotering (eng. quota), vilket innebär att klienter kan begränas till att bara använda en viss mängd utrymme på filsystemet, en funktion som NFS och SMB måste förlita sig till operativsystem att hantera.

4.4.2 Caching

En del av AFS prestanda kan förklaras av dess komplicerade algoritm för caching. Klienten begär att få öppna en fil av servern, vilket resulterar i att hela filen låses och förs över till klienten. Alla efterföljande läs och skrivoperationer görs helt och hållet i den lokala cachen. När filen sedan stängs förs alla ändringar över till servern. Servern håller reda på vilka klienter som i given stund arbetar med en fil och informerar dessa när ändring har skett i den lokala cachen på andra klienter. Uppdateringar av de lokala cachesystemen sker automatiskt när ändringar sker. Detta systemet har en nackdel; det går inte att låsa delar av en fil likt SMB. Detta är medvetet gjort, då det passade de något speciella kraven för Andrew Projeketet med många små filer snarare än få stora.

4.4.3 Redundans

Eftersom AFS kan lagra samma fil på flera olika enheter så kan primärservern crasha utan att någon av klienterna märker av det. En av sekundärservarna tar då över och delar ut filen på precis samma sätt som tidigare. De exakta detaljerna för hur cachen hanteras vid en crash är implementationsbaserat. [9][11]

4.5 Prestanda

I ett test där NFS 3, NFS 4 och AFS jämfördes [12] så såg man tydligt spikar i NFS prestanda på vissa operationer och AFS låg stabilt på ungefär samma resultat oavsett vilken filoperation man genonförde. NFS 3 presterade i snitt bättre än AFS, men marginalerna är små.

5 Diskussion och slutsatser

Efter att ha jämfört de olika systemen ur säkerhet, prestanda och skalbarhets-perspektiv samt gjort en kortare genomgång av deras olika metoderna de använder för att lösa de gemensamma problemen så skall jag försöka dra någon form av slutsats av vilket system jag anser vara överlägset. Detta är ingen enkel sak att göra utan att först ta hänsyn till vilken situation man skall använda systemet i. Av de tre systemen är AFS definitivt det mest väldesignade och väl anpassat till en distribuerad miljö, detta är föga förvånande då det var del av ett projekt för distribuerade system. NFS stora fördelar är dess råa prestanda och enkelhet, för mindre mijöer med få noder och ett privat nätverk så är detta nog att föredra. SMB är det mest svårplacerade av de tre. Det fungerar synnerligen bra i enkla blandade miljöer där säkerhet inte är av nån större vikt och det ingår flera windowsplatformar. Annars är det och kommer förmodligen att förbli ett stängt protokoll med odokumenterade funktioner och

problem med dialekter. Finns det möjlighet att ha en Kerberos-server i sitt nätverk och man vill ha en tekniskt kompetent lösning och möjlighet att expandera sitt nätverk så är AFS det givna valet.

6 Ett högst subjektivt försök

Rapporten har förhoppningsvis gått genom de tre stora distribuerade på ett tillfredsställande sätt, men det är en aspekt av dem som jag inte har inkluderat då det är helt och hållet en subjektiv fråga, nämligen användbarvänlighet. Som en avslutning på projektet tänkte jag därför jämföra de tre olika systemen ur ett administratörperspektiv. Försöket gick ut på att dela ut en katalog med en musiksamling. Servern var en AMD Athlon(tm) 64 Processor 3000+ som körde Kubuntu med linuxkärnan 2.6.12. Klientdatorerna var en Windows XP (64-bitars) samt en linuxdator (gentoo) med 2.6.14-kärnan.

6.1 NFS

Efter att ha tankat hem och kompilerat källkoden så var det bara att editera filen /etc/exports och mata in rätt parametrar. Ett mindre hinder var att manualsidan till exports inte verkade vara helt uppdaterat och ett syntaxfel uppstod som löstes efter en stunds googlande. Enheten kunde sedan enkelt mountas av gentoo-maskinen med ett enda kommando. Windows-sidan krävde lite mer arbete då det enda sättet att använda NFS i Windows verkar vara Microsoft Services for UNIX, ett program som inte är gratis för vanliga användare, men som går att få gratis via Chalmers. Installationen var enkel och datorn kunde ansluta till servern på första försöket. Ett irriritationsmoment var de långa handskakningarna, lista filer och initiera en läsning tog nästan en minut, vad det beror på lämnas osagt.

6.2 SMB

Senaste versionen av samba laddades ner och kompilerades, och modulerna i linuxkärnan byggdes utan problem. Med en Samba-HOWTO i ena fönstret och en editor i andra så konfigurerades systemet på ungefär 15 minuter. Windows-datorn hittade inte servern med hjälp av NetBIOS-protkollet, utan IP-numret var tvunget att matas in manuellt. Linuxdatorn mountade enheten utan några som helst problem.

6.3 AFS

Servermjukvaran tankades hem, kompilerades och kernelmodulerna kompilerade även dem utan några problem, men lyckan blev kort. Efter att ha läst både manualsidorna, en how-to och den officiella dokumentationen (vilket tog mycket längre tid än SMB och NFS tillsammans) så lyckades jag inte få servern att ens starta utan felmeddelande. Nånstans på vägen mellan Kerberos-servern och AFS-servern skedde det något högst oväntat och efter mycket googlande lyckades åminstonde servern starta utan klagomål. Tyvärr lyckades varken windows-klienten eller linux-klienten att förstå administratörens koncept av celler och volymer och vägrade all form av kommunikation. Detta är ungefär samma slutsats som tas i den utmärka jämförelsen mellan NFS och AFS som anges som referens. Det är fullt möjligt att AFS skulle fungera om jag hade lagt mer tid på det, men efter

flera timmars läsande och felsökande så ansåg jag det vara onödigt slöseri med tid. AFS tekniskt avancerade system kom i min mening med ett ganska obehagligt pris, en erfaren linuxanvändare kan inte sätta upp, än mindre administrera systemet inom en rimlig tid.

References

[1] “Linux Journal”, http://www.linuxjournal.com

[2] “NFS Performance Tests”, http://www.cisl.ucar.edu/hps/TECH/LINUX/linux.html

[3] “NFS Spec Test”, http://www.spec.org/sfs97r1/docs/sfs-3.0v11.html

[4] “Wikipedia - NFS Article”, http://en.wikipedia.org/wiki/Network_file_system

[5] “Just what is SMB?”, Richard Sharpe, http://samba.anu.edu.au/cifs/docs/what-is-smb.html

[6] “Implementing CIFS”, Chris Hertel, http://ubiqx.org/cifs/index.html

[7] “Network Computing magazine’s web site”, Jeff Ballard, http://www.networkcomputing.com/

[8] “Wikipedia - SMB Article”, http://en.wikipedia.org/wiki/Server_Message_Block

[9] “Grand Central Website”, http://http://grand.central.org

[10] “OpenAFS Website”, http://www.openafs.org/

[11] “Wikipedia - AFS Article”, http://en.wikipedia.org/wiki/Andrew_file_system

[12] “NFS&AFS File Systems”, http://web.math.jjay.cuny.edu/reports/NFS&AFS%20File%20Systems.pdf