HÖGSKOLAN SKÖVDE

Institutionen för kommunikation och information

Examensarbete i datavetenskap 15hp (fristående)
C-nivå
Vårterminen 2008

Virtualisering:

En prestandajämförelse mellan fullständig- och parallell systemvirtualisering

Magnus Lindberg

Virtualisering: en prestandajämförelse mellan fullständig- och parallell systemvirtualisering.

Examensrapport inlämnad av Magnus Lindberg till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Alexander Karlsson.

2008-06-11

Härmed intygas att allt material i denna rapport, vilket inte är mitt eget, har blivit tydligt identifierat och att inget material är inkluderat som tidigare använts för erhållande av annan examen.

Signerat: _____

Virtualisering: en prestandajämförelse mellan fullständig- och parallell systemvirtualisering.

Magnus Lindberg

Sammanfattning

Virtualisering är en abstraktion av underliggande fysisk hårdvara som omvandlas till en förutbestämd struktur av hårdvara via mjukvara. En virtuell maskin kan då vara frånkopplad från hårdvaran. Virtualisering tillåter hårdvara att delas upp som flera separata virtuella hårdvaror vilket kan ske transparent för operativsystem i virtuella maskiner. Virtualisering ökade under 90-talet och det utvecklades två virtualiseringsteknologier: *(i)* den fullständiga systemvirtualisering och *(ii)* parallell systemvirtualisering. Fullständig systemvirtualisering erbjuder abstraktion som utgör en frånkoppling från hårdvara. Operativsystem som använder en virtuell maskin känner då inte till att virtualisering skett med resultatet att alla operativsystem kan användas. Parallell systemvirtualisering använder en delvis abstraktion då operativsystem modifieras för att virtuell maskin skall vara medveten om att virtualisering utförts för att möjliggöra för prestandaförbättringar. Den problemställningen som ställts försöker utröna vilken av dessa två teknologier som kan leverera bästa prestanda över FTP. Experiment har då utförts och visade att det är inga skillnader mellan teknologierna.

Nyckelord: Parallell systemvirtualisering, Fullständig systemvirtualisering, Virtualisering, Xen, VMware, Experiment, Hårdvaruemulator, Binär direktöversättning och direktexekvering, Hypercall.

Innehållsförteckning

1 Introduktion

En virtuell maskin är en mjukvarurepresentation av en fysisk dator. Detta möjliggörs med hjälp av en mjukvara som ligger som ett lager mellan den virtuella maskinen och hårdvaran eller operativsystemet. Denna mjukvara benämns som ”*hypervisor*” eller ”*virtual machine monitor*” och utför den abstraktion som gör det möjligt för de operativsystem och applikationer som körs i en virtuell maskin, att tro att de arbetar mot en fysisk hårdvara fast att den i själva verket är virtuell. Benämningarna ”*hypervisor*” och ”*virtual machine monitor*” kommer från och med nu i rapporten att nämnas som enbart monitor. Användandet av virtuella maskiner ger den egenskapen av att det går att logiskt partitionera (dela upp) en fysisk dator i flera delar, likt hur en fysisk hårddisk delas upp i flera olika logiska systempartitioner eller fysiskt minne som delats upp i delar av virtuellt minne. Partitionering kan sedan användas i något som benämns, ”konsolidering av servrar” (sammanslagning av resurser). Det är ett begrepp som innebär att flera underlastade servrar virtualiseras och placeras i en enda fysisk server. Detta för att ge en så hög belastningsgrad som möjligt på fysiska servrar och minska den tid som servrar befinner sig i outnyttjat (eng. ”*idle*”) läge. Vidare så bidrar virtualisering till att minska bland annat kostnader för el, kylanordningar och server utrymme. Enligt Marshall m.fl. (2006) är dagens servrar enbart 8-12% belastade.

Virtualisering tog sin början på 1970-talet (IBM VM/370), trenden med att använda virtulisering minskade sedan och det var inte förrän på 1990-talet som det ökade på nytt. Dåtidens system var decentraliserade (utspridda), tog upp mycket plats med många servrar som körde ett fåtal tjänster och var oftast underbelastade och svåradministrerade. Lösningen på detta problem ansågs vara att återgå till ett centraliserat synsätt. Enligt Larsson (2008) kommer 15% av världens alla servrar att bestå av virtualiserade maskiner från 2011. Fördelarna med virtualisering är bland annat ökad säkerhet genom den isolering som sker mellan virtuella maskinerna. Detta gör dem lämpliga för testning och felsökning av känsliga program. Populära tekniker som används för virtualisering är: fullständig och parallell systemvirtualisering. Dessa tekniker har både för och nackdelar samt är vanligt förekommande på marknaden och det är skillnaden mellan dessa i prestanda som är intressant att undersöka. Vilken skillnad finns det mellan den parallella och fullständiga systemvirtualiserings teknikerna? Vilken teknik presterar bäst med avseende på filöverföringsprestanda?

Här följer en översikt över rapportens struktur. Kapitel 2 definierar vad virtualisering är och hur den fungerar samt vilka olika sätt som finns för att uppnå virtualisering.

Kapitel 3 beskriver problembeskrivningen för denna rapport. Kapitel 4 tar upp och beskriver de metoder som används för att uppnå de mål som ställts i problembeskrivningen. Kapitel 5 presenterar resultatet och analys av denna. Slutsatsen och diskussion kring resultatet samt framtida arbeten i kapitel 6 avslutar sedan rapporten.

2 Bakgrund

Nedan följer den bakgrund som skall hjälpa till med att redogöra för de begrepp och tekniker som innefattar virtualisering. I sektion 2.1 ges en definition på virtuell maskin. Sektion 2.2 ger en introduktion till virtualisering. Nästa sektion 2.3 beskriver de krav som finns för att uppnå lyckad virtualisering. Efter detta följer sektion 2.4 vilket redogör i detalj för hur virtualisering fungerar. Sedan kommer sektion 2.5 som tar upp två vanligt förekommande virtualiseringstekniker den fullständiga och parallella systemvirtualiseringen samt hur dessa är konstruerade. I slutet av rapporten kommer sektion 2.6 vilket beskriver de fördelar som finns med virtualisering samt sektion 2.7 som tar upp vad FTP är och hur detta protokoll används.

2.1 Definition av en virtuell maskin

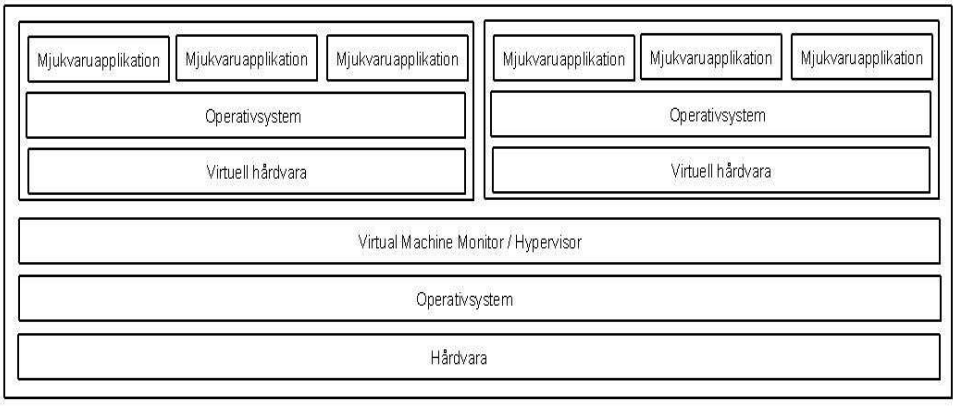
Bullers, Jr (2006) definierar en virtuell maskin som ett datorsystem i vilket de instruktioner som utförs kan skilja sig åt från dem som faktiskt exekverar på hårdvaran för en given arbetsuppgift. Vidare beskrivs en virtuell maskin som en mjukvaruabstraktion av ett datorsystem vilket försöker representera dess hårdvara.

2.2 Introduktion till virtualisering

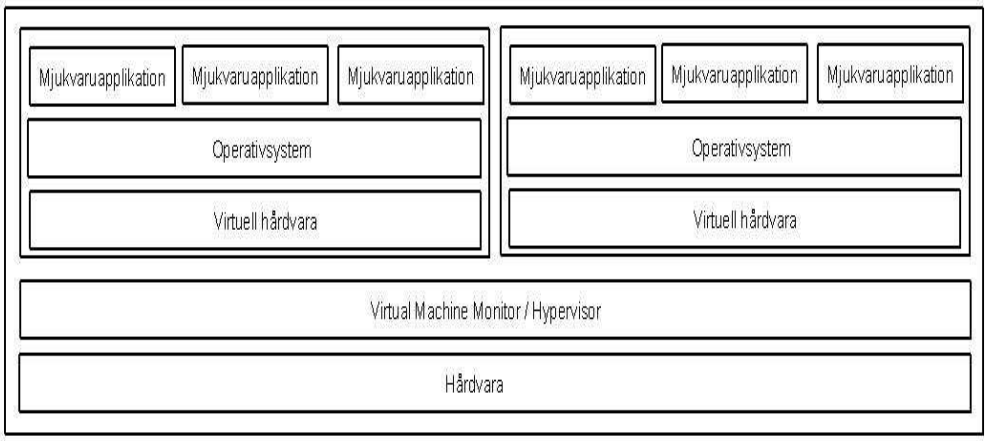
Användandet av virtualisering är ett tillvägagångssätt vars syfte är att en fysisk dator kan agerar som om den vore flera separata datorer. Varje virtualiserad dator består av samma grundläggande arkitektur som det av en generell representation (i form av processor, minne, I/O enheter m.m.) av en fysisk dator. För att lyckas med detta finns olika tekniker med olika för- och nackdelar. Om det skall vara möjligt med att få en fysisk dator att agera som om den består av flera datorer, så måste dess fysiska hårdvara representeras genom mjukvarukonstruktioner. Detta möjliggörs genom ett mjukvarulager som utför något som benämns som abstraktion. Abstraktion används ofta inom många mjukvarumiljöer, ett exempel på användandet av abstraktion inom Microsoft Windows är ”*Windows Hardware Abstraction Layer (HAL)*”. HAL möjliggör ett gemensamt sätt för alla drivrutiner och mjukvara att kommunicera med hårdvara transparent åt användaren och utvecklare. Detta underlättar för utvecklare då de slipper skapa specifik mjukvara och drivrutiner för enskild tillverkare och datortyp, utan då bara drivrutiner för Windows plattformen som sköter resten. När abstraktion nämns i virtualiseringssamanhang, så är det en representation av vanligt förekommande hårdvara som är helt utformad genom mjukvara. Resultatet blir mjukvara som både ser ut och agerar som hårdvara åt operativsystem som befinner sig i virtuella maskiner. Detta möjliggör för installation av ett operativsystem på hårdvara som inte fysiskt existerar. Virtualisering tillåter en fysisk dators resurser att delas eller individuellt tilldelas av flertalet virtualiserade miljöer samtidigt. Miljöerna kan sedan

användas för att interagera med varandra eller vara totalt isolerade ifrån varandra. En miljö behöver inte vara utvecklad för att vara helt frånkopplad från hårdvara. Sådana virtuella miljöer benämns då som virtuella maskiner. Virtuella maskiner brukar oftast innehålla någon form av ett operativsystem som t.ex. Linux, Windows, m.m. Operativsystem som körs som virtuella maskiner benämns oftast som gästoperativsystem inuti en värd eller värdoperativsystem. Instruktioner ifrån en virtuell maskin skickas oftast direkt till den fysiska hårdvaran på värden när det är möjligt. Detta gör att en virtuell maskin är både snabbare och mer effektiv än användandet av emulering, trots att den måste utföra komplexa instruktioner som måste fångas upp och tolkas korrekt av mjukvara (monitor).

En dator som använder virtualiserade miljöer är annorlunda organiserad jämfört med en typisk dator. Den består av en uppsättning hårdvara vilket ett operativsystem är installerat på t.ex. Windows eller Linux. Ovanpå operativsystemet är sedan en virtualiseringsmjukvara installerad och i den finns en eller flera virtuella maskiner. Där varje virtuell maskin agerar som en separat hårdvara vilket kan innehålla ett eget operativsystem med tillhörande applikationer installerade. Se Figur 1. Den andra typen av virtualiserad miljö är en sådan, där miljön är installerad direkt mot datorns hårdvara s.k. *(”bare metall”).* Detta bidrar då till att flertalet virtuella maskiner kan skapas med egna operativsystem och applikationer med maximal effektivitet, då en speciellt utformad monitor som utnyttjar mindre resurser på ett virtualiserat system används. Se Figur 2. Baserad på Marshall et al. (2006).



Figur 1. Illustration över lager vid fullständig systemvirtualisering, Marshall et al. (2006).



Figur 2. Illustration över lager vid parallell systemvirtualisering, Marshall et al. (2006).

2.3 Krav för virtuella maskiner

Goldberg (1974) beskriver en virtuell maskin som en effektiv och isolerad kopia av en riktig dator. Han beskriver också att det finns tre krav som måste uppfyllas för att nå detta.

2.3.1 Identisk med ursprungsmaskinen

Det första kravet är att en monitor måste tillåta en miljö för program som är i grund och botten identisk med ursprungsmaskinen. Dock med undantag för möjlig påverkan då monitor inte alltid kan garantera att operationer utförts under förutsatt exekveringstid, då den måste hantera eller bearbetar tillgängliga resurser åt en eller flera samkörande virtuella maskiner på samma hårdvara.

2.3.2 Effektivitet

Det andra kravet är att program som kör i denna miljö, inte får påvisa någon större prestandaförsämring. Detta kräver att ett statistiskt sätt stor del av de instruktioner som den virtuella processorn bearbetar exekveras direkt på den riktiga processorn, utan inverkan från monitor. Detta påstående plockar bort emulatorer och simulatorer som kandidater till virtuella maskiner. Då de inte har tillgång till att utföra direktexekvering mot processor.

2.3.3 Kontroll av resurser

Monitor sägs ha total kontroll över resurser (minne, I/O, m.m. men dock inte alltid processor) om:

1. Det skall inte vara möjligt för en miljö skapad under monitor att få tillgång till resurser som inte explicit har tilldelats den.

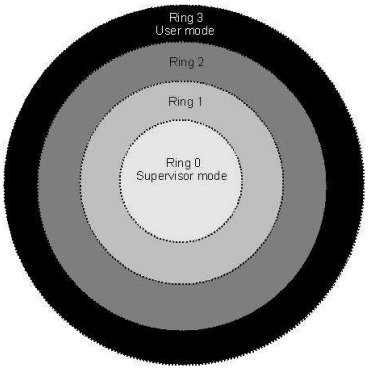
2. Det skall vara möjligt under vissa omständigheter för monitor att återta kontrollen över resurser som redan har allokerats.

2.4 Monitor

Som tidigare nämnts, för att kunna erbjuda virtuella miljöer behövs en monitor. Den har som uppgift att samköra parallellt med virtuella maskiner och genom ett mjukvarulager abstrahera den fysiskt underliggande hårdvaran åt de virtuella maskinerna. Den kontrollerar virtuella resurser som processor, I/O, lagring och minne etc. åt de virtuella maskinerna. En annan viktig del är att erbjuda isolering mellan samkörande gäster, Rose (2004). Enligt Opsahl (2007) är syftet med en monitor att erbjuda isolering, schemaläggning, resurshantering och att se till att virtuella maskiner fungerar på ett korrekt sätt. Menascé (2005) beskriver att en monitor kör i ett s.k. ”*supervisor mode*” (fullständig åtkomst till hårdvara) och kontrollerar all åtkomst till resurser som delas av virtuella maskiner i ”*user mode*” (begränsad åtkomst till hårdvara).

2.4.1 Hur fungerar virtualiseringstekniken?

Monitor lyckas erbjuda isolering mellan virtuella maskiner genom att flytta maskinerna och dess mjukvara närmare den fysiska processorn. Detta genom användandet av olika priviligerade lägen som benämns som ringar. Detta är en av grundstenarna bakom effektiv virtualisering. Den vanligt förekommande x86 kompatibla processorerna på PC-arkitekturen har stöd för ett skyddat läge (uppdelning av minne), vilket används för att adressera minne och multitasking. Detta skyddade läge gjorde det möjligt att dela upp processer så att de inte kunde skriva till minne som befinner sig utanför dess tilldelade addressutrymme och för möjligheten att växla mellan de olika processer som finns. Det skyddade läget använder sig av fyra priviligerade lägen eller ringar från 0 till 3. Det systemminne som finns att tillgå delas upp i olika delar och varje del tilldelas sedan till en av de olika ringarna. Processorn har sedan full kontroll över de olika priviligerade lägena, för att sedan bestämma vad som får ske med kod och data inom en ring (uppdelat addressutrymme). Ring 0 (”*supervisor mode*”) är den innersta med full kontroll över processor och exekvering av priviligerade instruktioner för I/O och minneshantering. Ring 3 *(”user mode”)* det yttersta lagret används för att erbjuda isolering genom att begränsa åtkomst till processor. Skulle ett program i ring 3 försöka få åtkomst till minne som befinner sig utanför ringen så skulle ett anrop ifrån hårdvara stoppa exekveringen. Se Figur 3.



Figur 3. Illustration över de olika priviligerade (ring) lägen på x86 plattformen, Garcia & Williams (2007).

Normalt sett så är det i ring 0 som ett operativsystem befinner sig och applikationer i ring 3. Om ring 0 lider av instabilitet eller annan påverkan så kommer också de efterföljande ringar 1,2 och 3 att påverkas. För att lösa detta så var det nödvändigt att flytta ring 0 närmare de virtuella maskinerna för att isolera de olika gästerna. Påverkan på ring 0 och underliggande lager i en gäst inverkade då inte på en annan gäst som också befann sig i en egen ring 0, utan bara lager inom varje individuell gäst. Monitor placerar sig då i ring 0 och ger gästoperativsystemet illusionen av att de också befinner sig i ring 0, fast de i själva verket är i 1,2 eller 3. Ju längre bort en gäst befinner sig från den riktiga ring 0, desto längre bort från hårdvaran befinner sig även gästen med ökad prestandaförlust som följd. Detta för att monitor skall hantera och kontrollera interaktionen mellan virtuella maskiner och den underliggande hårdvaran (processor, minne och I/O). Det finns två typer av monitor som använder sig av ring 0 representationen *(i)* typ1 monitor och *(ii)* typ2 monitor. Dessa beskrivs mer utförligt i Goldberg (1973). Baserat på Garcia & Williams (2007).

- **Typ1 monitor**, en mjukvara som kör direkt på hårdvaran i det äkta ring 0 läget. Övriga gästoperativsystem kör sedan i ett högre ring lager ovanpå hårdvaran. Detta bidrar till äkta isolering mellan de virtuella maskinerna.
- **Typ2 monitor**, en mjukvara som kör inom ett operativsystem, oftast i ring 3 läget. Då monitor befinner sig i ring 3 så är den så långt bort ifrån hårdvaran som är möjligt på x86 plattformen. Detta bidrar till prestandaförlust då önskad åtkomst till och operationer på hårdvara, måste passera flera lager (ringar).

2.4.2 Virtualisering på x86 plattformen

Enligt Williams & Garcia (2007) så har arkitekturen för x86 fyra priviligerade ringar. Vanligtvis så befinner sig ett operativsystem i ring 0 och dess applikationer i ring 3. På detta sätt är operativsystemet i full kontroll över plattformens resurser. I och med användandet av monitor så måste denna befinna sig i ring 0 och gäster i ring 1 eller 3. Problemet med detta är dock att dagens operativsystem inte har utvecklats för att köra

i andra lägen än i ring 0 och x86 plattformen var inte utvecklad för att användas i en delad miljö för virtuella maskiner. Detta bidrog till att det fanns 17 priviligerade instruktioner på x86 plattformen som utgjorde att processorn inte var lämplig för virtualisering (För mer information se Robin & Irvine (2000)). Dessa instruktioner används av de flesta operativsystem och kan leda till konflikter som systemfel eller felaktiga svar. För att lösa detta problem enligt Goldworm & Skamarock (2007) och Reuben (2007) så har både Intel och AMD utvecklat hårdvarustöd för virtualisering på x86 plattformen. Lösningen är att de har introducerat en ytterliggare ring nivå ring-1 (ring minus 1). Monitor får då köra i ring-1 (”*root mode*”) och varje virtuellt gästoperativsystem får sedan köras i ring 0 (”*non-root mode*”) och applikationer i ring 3. Intels teknik går under namnet *”The Intel Virtualization Technology (Intel VT)”* och AMD under *”AMD Virtualization (AMD-V)”*, dessa tekniker dök upp på marknaden under mitten av 2006.

2.5 Olika kategorier av virtualiseringstekniker

De problem som finns för virtualisering på x86 plattformen och Popek och Goldberg´s formella krav har bidragit till utvecklandet av flera olika tillvägagångssätt för virtualisering. Detta kapitel kommer här att beskriva två av dessa den fullständiga och parallella systemvirtualisering.

2.5.1 Fullständig systemvirtualisering

Goldworm & Skamarock (2007) samt Williams & Garcia (2007) beskriver att fullständig systemvirtualisering utförs med hjälp av en virtualiseringsmjukvara som kör ovanpå ett befintligt operativsystem och erbjuder emulering av den underliggande hårdvaran åt virtuella maskiner. Detta bidrar till att all mjukvara som kan exekvera på hårdvaran också kan köras i en virtuell maskin. Enligt Reuben (2007) är denna teknik den mest förekommande och erbjuder störst stöd för olika operativsystem bland virtualiseringstekniker. Tekniken erbjuder total isolering mellan virtuella maskiner och monitor. Fördelen med denna teknik är att operativsystem inte behöver modifieras för att vara virtualiseringsbar. Tekniken erbjuder prestanda som är nära den av hårdvara men kan lida av prestandaförluster på x86 plattformen. Detta då priviligerade instruktioner måste passera flera virtualiseringslager (först i det virtualiserade operativsystemet sedan genom monitor översättas till data som värdoperativsystemet kan tolka och hantera samt kommunicera med hårdvaran genom drivrutiner i värdoperativsystemet eller monitor).

Enligt Rose (2004) samt Qian Huang (2006) är VMware en applikation som använder sig av fullständig systemvirtualisering och bygger sitt tillvägagångssätt på något som

benämns ”*Hosted virtual machine architecture*”. Detta innebär att monitor kör ovanpå ett existerande värdoperativsystem likt en applikation. Detta möjliggör att VMware kan tillåta värdoperativsystemet att förse mjukvaran med ett brett stöd av drivrutiner för olika hårdvarukomponenter. På så sätt slipper VMware integrera ett stort drivrutinsstöd i mjukvaran. Själva monitor består av en drivrutin vid namn ”*VMDriver*” som installeras på värdoperativsystem och hanterar åtkomst till resurser för de olika virtuell maskinerna. Denna är speciellt anpassad för att ge snabb åtkomst till enheter och hårdvara via värdoperativsystemet. VMware förser de virtuella maskinerna med en uppsättning av standard drivrutiner. Att möjliggöra för operativsystemet att hantera drivrutiner och ge åtkomst (abstrahera) drivrutiner, minskar en del av den komplexitet som mjukvaran redan utgör. Enligt Qian Huang (2006) använder VMware samma gränssnitt ISA (”*Instruction Set Architecture*”) för gästoperativsystem i de virtuella maskinerna som för värdoperativsystemet vid kommunikation med x86 hårdvaran. ISA är den del av processor som är synlig för programmerare och kompilatorer för åtkomst till ej priviligerat användar (operativsystem och applikationer) och priviligerade system (operativsystem) instruktioner samt används för att förenkla för programmerare då de inte behöver hålla reda på detaljer kring en hårdvara vid utvecklandet. Exempel på mjukvara som använder fullständig systemvirtualisering är: Microsoft Virtual Server och VMware Server (tidigare VMware GSX).

2.5.2 Binär direktöversättning och direktexekvering

Enligt Rose (2004) samt VMware (2007), installeras VMware i värdoperativsystemet likt en applikation. När denna applikation körs använder den en drivrutin vid namn ”*VMDriver*” som laddas in i värdoperativsystemets kärna och bildar en priviligerad monitor. Denna komponent agerar sedan som ett gränssnitt mellan virtuella gästoperativsystem och kör samtidigt ovanpå hårdvaran med värdoperativsystemet (ring 0). Komponenten har som uppgift att översätta kod från värdoperativsystemets kärna mot kod som är virtualiseringsbar för de virtuella maskinerna. Den fysiska processorn blir nu tvungen att växla mellan de körande processer som utgör värdoperativsystemet och gästoperativsystemen i de virtuella maskinerna. Gästoperativsystemen och deras applikationer tillåts köra i det mindre priviligerade och restriktiva användarläget (ring 1 resp. ring 3). Det är här som binär direktöversättning och direktexekvering kommer in. För de icke priviligerade instruktioner som utförs av gästoperativsystem kan och tillåts direkt åtkomst till den fysiska hårdvaran (direktexekvering), medan de priviligerade instruktionerna som utförs översätts direkt under körning (binär direktöversättning) för att det skall vara möjligt att introducera brytpunkter i känsliga instruktionerna, Chen & King (2002). Dessa brytpunkter utlöses sedan för att instruktionerna skall fångas upp av monitor

under körning. Detta bidrar sedan till en emulering av dessa känsliga instruktioner i mjukvara (monitor) utan att behöva nå hårdvaran. Emuleringen skall efterlikna vad som sker på hårdvara under exekvering av priviligerade instruktioner, helt utan gästoperativsystemets vetskap. Binär direktöversättning ger en ökad komplexitet vid utvecklandet av mjukvaran men kan då erbjuda en fullständig uppsättning av x86 instruktioner (x86 ISA) åt de virtuella maskinerna. Detta bidrar till att alla operativsystem och applikationer för x86 plattformen är kompatibla för att fungera i virtuella maskiner utan att behöva modifieras. Denna teknik är således snabbare än emulering då denna enbart tolkar instruktioner för en ISA åt en annan ISA, Qian Huang (2006). Samt att fullständig systemvirtualisering använder sig av både direktexekvering (hårdvaruåtkomst) och direktöversättning, med möjlighet till cacheing av instruktioner. Dock är de virtuella maskinerna och gästoperativsystemen helt påverkningsbara av värdoperativsystemets schemaläggning. Då de virtuella maskinerna kör som applikationer i värdoperativsystemet.

2.5.3 Parallell systemvirtualisering

Parallell systemvirtualisering är en teknik som tar vara på de fördelar virtualisering ger, men utvecklats för att lösa de problem med prestandan som finns med tekniken, Ogden (2006). Erbjuder delvis emulering av den underliggande hårdvaran och använder sig av en optimerad monitor (mindre kod, 50 000 rader) som bidrar med prestanda som är väldigt nära den som hårdvara ger, Reuben (2007) samt A Performance Comparison of Commercial Hypervisors (2007). Enligt Williams & Garcia (2007) samt Goldworm & Skamarock (2007) är skillnaden mot fullständig systemvirtualisering att gästoperativsystemen är medvetna om att de befinner sig i ett virtualiserat system och att monitor befinner sig direkt ovanpå hårdvaran (”*bare metall*”). De gästoperativsystem som önskas köras på parallell systemvirtualisering måste då modifieras för att tillåtas fungera. Detta kan då vara ett problem då vissa operativsystem och drivrutiner är av sluten källkod och kan då inte portas. Detta gör att tekniken inte är så bakåtkompatibel, dock med hårdvarustöd för virtualisering så behöver operativsystem inte modifieras. Gästoperativsystem på parallell systemvirtualisering är medvetna om att de är virtualiserade och använder sig då av en speciell API (ett mjukvarugränssnitt) för att kommunicera med monitor och hårdvara mer effektivt med ökad prestanda som följd. Exempel på denna teknik är: VMware ESX, Xen 3.0, Microsoft Windows Server 2008 Hyper-V och avarter av IBM VM/370 för stordatorer.

2.5.4 Hypercall

Parallell systemvirtualisering är ett sätt att utföra virtualisering genom att erbjuda en abstraktion av en underliggande hårdvara till virtuella maskiner. Dock skiljer den sig

åt från fullständig systemvirtualisering, då den erbjuder en abstraktion som inte är identisk med hårdvaran, Qian Huang (2006). Detta för att virtuella maskiner skall kunna använda både fysikt existerande och virtuella resurser t.ex. gästoperativsystem kan hantera tidskänsliga tekniker för TCP effektivare, om gästoperativsystemet tillåts åtkomst till både virtuell (tid gästoperativsystem upptar processor) och verklig tid (räknare för processor cykler) för hårdvaran, Dragovic et al. (2003) samt Qian Huang (2006). Den hårdvara som abstraheras emuleras i kombination med att operativsystem i de virtuella maskinerna har ett nära samarbete med monitor. Detta uppnås genom att värdoperativsystemen modifieras för att dess kärna skall ersätta icke virtualiseringsbara instruktioner med hjälp av ”*hypercalls*” till monitor. Ett ”*hypercall*” är jämförbart med ett systemanrop i ett vanligt operativsystem, Dragovic et al. (2003) samt Rose (2004). Denna tillåter då värdoperativsystemen att utlösa anrop som fångas upp av monitor när priviligerade instruktioner utförs. Icke priviligerade instruktioner hanteras av gästoperativsystemen utan inverkan ifrån monitor. Vidare så används ett händelsebaserat system för att övervaka alla händelser som sker för varje domän t.ex. uppdaterad sidtabell eller DMA operationer. Detta händelsebaserade systemet används för att ersätta traditionella hårdvaruanrop och kan då användas för att hantera och optimera händelser (bestäms av monitor) mellan monitor och de olika domänerna t.ex. virtuell diskåtkomst klar eller data mottaget från nätverket, Dragovic et al. (2003). Vidare är det möjligt att helt använda omodiferade applikationer i de gästoperativsystem som körs i de virtuella maskinerna. Modifiering av gästoperativsystem ger en ökad prestanda och möjlighet för bättre hantering för tidskänsliga operationer, Qian Huang (2006). Enligt Williams & Garcia (2007) så har arkitekturen för x86 har fyra olika priviligerade lägen, 0-3. Där ring 0 har direkt åtkomst till hårdvara och det är här som värdoperativsystemet befinner sig. För att erbjuda isolering och korrekt funktionalitet så måste monitor köra i en ring som har högre priviligerat läge än gästoperativsystemen. Därmed befinner sig monitor i ring 0 och gästoperativsystem i ring 1. Detta möjliggör att priviligerade instruktioner ifrån gästoperativsystem fångas upp av monitor i ring 0, vilket sedan utför instruktionerna där det är möjligt. Om inga priviligerade instruktioner utförs vilket är vanligt för den största delen av tiden, så behöver inte den belastning som medföljer vid att anropa och undersöka instruktioner (monitor) skapas, Qian Huang (2006). Skulle gästoperativsystem försöka utföra priviligerade instruktioner, skulle processor ignorera eller rapportera fel då enbart monitor har rättigheter för detta.

Xen är en applikation som är baserad på parallell systemvirtualisering. Xen har delat upp värdoperativsystem och gästoperativsystem i något som de benämner domän 0 (värd) och domän U (gäst). Båda dessa är representationer för virtuella maskiner. Domän 0 är en högre priviligerad virtuell maskin som har tillgång till den fysisk hårdvaran och gäster via api. Den kontrollerar och hanterar process schemaläggning,

minnesallokering samt I/0 och disk åtkomst för de olika gästerna. Den utför abstraktion av den fysiska hårdvaran till enklare representationer av mjukvaruhårdvara för gäster t.ex. block I/0, Ethernet, frame buffer o.s.v. Domän 0 agerar utanför monitor i värdoperativsystems kärnan, detta minskar påverkan på monitor vid eventuella felaktigheter i domän 0, A Performance Comparison of Commercial Hypervisors (2007).

Summering är att gästoperativsystem vet att de är virtualiserade. Tekniken har en lägre belastning än fullständig systemvirtualisering. Det går inte att använda omodifierade operativsystem, detta kräver då en högre kompetens av utvecklare vid modifiering av operativsystemskärnor. Det är mer komplext att utföra och programmera binär direktöversättning, än att lyckas med parallell systemvirtualisering, Understanding Full Virtualization, Para virtualization, and Hardware Assist (2007).

2.6 Fördelar med virtualisering

Virtualisering har många fördelar, under 1960-70 talet ansågs det vara ett bra alternativ då stordatorer och hårdvara var mycket dyr och tog upp stora utrymmen, Menascé (2005). Fördelen då var att det gick att logiskt partitionera en fysisk stordator till flera virtuella representationer av hårdvaran. Det upptäcktes då att det var möjligt att köra flertalet processer och applikationer samtidigt. Detta bidrog då till ökad effektivitet av hårdvaran och minskat underhåll av dyr hårdvara, Reuben (2007). Men de mest grundläggande fördelarna med virtualisering är möjligheten för resursdelning av hårdvara mellan virtuella maskiner och isolering mellan de olika virtuella miljöerna. Sedan var det inte för än under 1980-talet tack vare den tekniska utvecklingen till följd av Moore´s lag (antalet transistorer fördubblas under en period av två år), som kostnaden för hårdvara sjönk och introduktionen av minidatorer bidrog till att stordatorer och virtualisering övergavs. Vad som nu skedde var att det övergick ifrån ett centraliserat synsätt till decentraliserat, där varje kontor och person hade tillgång till en egen dator. Denna ökning fortsatte under 1990-talet och bidrog till en hög komplexitet, då utspridda system blev allt svårare att administrera. Andra skäl var att enskilda servrar körde en applikation per server, vilket resulterade i att servrarna fick en låg belastningsgrad så lågt som 5%. Detta resursslösande i samband med att flertalet servrar tog upp mycket utrymme, extra kylning och ökade el- samt administrations kostnader. Var utgångspunkten för att återgå till ett centraliserat synsätt och till server konsolidering, Goldworm & Skamarock (2007).

2.7 File Transfer Protocol (FTP)

FTP är ett vanligt förekommande filöverförings protokoll över Internet. Protokollet finns integrerat i de flesta webbläsare och stöds av de många operativsystem. FTP möjliggör filöverföringar mellan två nätverksanslutna datorer över TCP/IP. Vanligtvis vid FTP så används en applikation som benämns som en FTP-klient på en dator och ansluter till en FTP-server på en annan avlägsen dator över FTP-protokollet. FTP använder sig av TCP för att erbjuda en pålitlig uppkopplingsorienterad anslutning mellan klient och server. Detta innebär förenklat bland annat att paket som försvinner eller blir korrupta från klient till server återskickas och bildar en pålitlig anslutning med hjälp av tekniker som återfinns i TCP. FTP-servern lyssnar efter klient anslutningar på port 21 och när en klient ansluter ges oftast en möjlighet att identifiera sig med användarnamn och lösenord. Det finns både kommando- och grafiskt baserade FTP-applikationer. Vid användandet av FTP kan användaren bland annat ladda upp och ner filer samt lista och ändra filrättigheter. Baserad på Casad (2003).

3 Problembeskrivning

Det finns många användningsområden för virtualisering de kan användas för bland annat testning och felsökning av applikationer, operativsystem samt erbjuda resursaggregering (sammanslagning av resurser). Virtualisering har ökat sedan mitten av 1990-talet då datorer varit utspridda och användes för ett mindre antal tjänster som mer eller mindre lämnade kvar outnyttjad kapacitet på många av hårdvarorna. Ökad kunskap och skillnader mellan virtualiseringstekniker kan innebära fördelar för organisationer och privatpersoner vid val av teknik. Virtualisering kan erbjuda kostnadsbesparningar av bland annat hårdvara och administration. Vidare kan virtualisering vara fördelsaktigt för miljön då teknikerna kan minska elkonsumtion genom att minimera outnyttjade resurser som hårdvara och utrustning (färre UPS, kylning och servrar samt el för lokaler m.m.), samtidigt kan den kräva mer av hårdvara och nätverk samt ökad kompetens kring hantering och utveckling av teknikerna. Virtualisering har visat sig ge fördelar, det kan då vara intressant att utföra någon form av experiment för de olika teknikerna. Detta för att ta reda på vilken teknik som kan vara lämplig att införskaffa. Denna studie kommer att inrikta sig på två av de vanligaste virtualiseringstekniker på marknaden, den fullständiga och parallella systemvirtualiseringen. Området valdes därför att kunskapen kring de olika teknikerna kan vara svår att förstå, speciellt över vilka skillnader som finns mellan dessa. Detta är viktigt vid ett eventuellt införande av virtualisering i en organisation. Vilken teknik presterar bäst? Vid ett införande av virtualisering privat eller i organisation, finns ett flertal olika alternativ. Två av dessa är idag mest förekommande på marknaden, nämligen den fullständiga (VMware) och parallella (Xen) systemvirtualiseringen. Ogden (2006), Chen & King (2002) samt Soltesz et al. (2007). Vad som är intressant här är att mäta den prestanda som de båda teknikerna kan leverera över FTP-protokollet. Ett webbhotell drivs av ett företag och erbjuder kunder att lagra webbsidor och data vilket blir tillgängliga över Internet. En del webbhotell tillåter då kunder att även hyra privata servrar eller virtuella maskiner. Det är då inte ovanligt att webbhotell tillåter ett flertaligt olika filöverföringsprotokoll för överföring av data. Ett sådant vanligt förekommande och etablerat protokoll är FTP. Detta protokoll används bland annat för rättighetsredigering, upp och nedladdning av webbinnehåll samt data. Protokollet har ett stort stöd i många applikationer, framförallt i webbredigeringsmjukvaror. Det är en konstant utveckling mellan dessa två virtualiseringsteknologier, det är då intressant att veta vilken av dessa som presterar bäst i nuvarande versioner. I fallet med webbhotell kan det ju vara så att kunder väljer den leverantör som erbjuder bästa prestanda. Andra experiment som

gjorts inom området t.ex. Deshane et al (2006) inriktade sig på att mäta och analysera de isolerande egenskaperna som finns mellan bland annat Xen och VMware under belastning. Soltesz et al. (2007), är ett annat exempel på experiment som utförts. Detta experiment fokuserade på att mäta och analysera hur effektiva de olika teknikerna är på bland annat hantering av http-trafik samt processorbelastning. Vidare så utför även de experiment för att få reda vilka isolerande egenskaper som finns mellan olika virtualiseringstekniker. Till skillnad från andra experiment kommer en prestandajämförelse att utföras mellan VMware och Xen med avseende på filöverföring via FTP-protokollet. Detta bör ge ett experiment med en realistisk bild då FTP är ett vanligt förekommande verktyg vid t.ex. webbhotell. Vidare så använder sig både Soltesz el al. (2007) och Deshane et al. (2006) av äldre versioner av Xen och VMware i sina experiment och resultatet ifrån dessa är då inte aktuell längre.

3.1 Problemprecisering

- Vilken av fullständig- och parallell systemvirtualisering presterar bäst över FTP-protokollet?
- Vilken påverkan har olika filstorlekar på prestanda?
- Är 128MB minne tillräckligt för en virtuell maskin?

4 Metod

I detta kapitel beskrivs de tillvägagångssätt som kommer att utföras för att undersöka problemet. Experiment som innefattar FTP-filöverföring kommer att göras och på insamlad data beräknas konfidensintervall för att skatta riktiga värden. Vidare kommer även graf som ”*boxplot*” att representera spridning av data. I sektion 4.1 förs en diskussion kring metodalternativ. Sektion 4.2 beskriver den valda metoden. Den sista sektionen 4.3 tar upp den hårdvara som kommer att användas för experimenten.

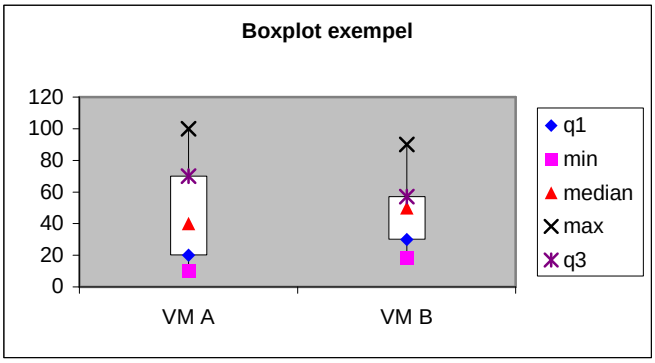
4.1 Metodalternativ

För att undersöka och svara på problemställningen ”Vilken av fullständig- och parallell systemvirtualisering presterar bäst över FTP-protokollet?”, kommer experiment att göras för att utföra prestandamätning vilket sedan kan analyseras med hjälp av statistiska metoder. Några alternativa metoder skulle kunna vara att utföra en litteraturstudie och analysera tidigare data från andra experiment inom området. Men avsaknaden av experiment som utförts mellan teknikerna i nuvarande versioner och över det protokoll som skall analyseras gör detta omöjligt.

4.2 Val av metod

För att insamla den data som krävs för analys kommer metod för experiment att användas. Data som inhämtas kommer bestå av mätningar i tid för överföring av fil från server till klient. Dessa mätdata är det som rapporteras av den integrerade FTP-klienten i Linux. Alternativt är att mäta kB/s, men valet föll på att använda tid. Varje delexperiment utför 100 filöverföringar och för varje experiment beräknas dess medelvärde. Motiveringen till att börja på 100 är att detta kan ändras till högre eller lägre värde om så önskas med hjälp av det skript som används för experimentet. Vidare kommer spänning att avlägsnas mellan hårdvara och nätverksutrustning i 60 sekunder mellan varje delexperiment. Detta för att ta hänsyn till eventuella missvisande data som presenteras till följd av okontrollerade orsaker eller händelser som t.ex. fel i mjukvara, hårdvara, nätverksbelastning samt olika typer av cachning i minne och processorregister. För att erhålla ett kontrollerat slutresultat kommer medelvärde beräknas enligt konfidensintervall på 95% för normalfördelad distribution vid avsaknad av standardavvikelse. Anledning till konfidensintervall är att den utgör ett mått på spridning. Denna beskriver ett intervall inom vilket säkerheten för var det riktiga värdet befinner sig är skattad med 95%. Normalfördelning beskriver den distribution som insamlad data utgör inom statistik för t.ex. medelvärden, Moore (2001). Beräkning för konfidensintervall kommer att ske enligt ($\mu = \bar{x} \pm k_1 \cdot s$), Lennart & Bertil (1988). I denna formel beskrivs μ som konfidensvärde, $\bar{x}$ är medelvärde och k_1 är värde från en matematiskt uträknad tabell för 95%

konfidensnivå vid 100 observationer samt att *s* representerar en skattning av standardavvikelsen. Graf som är utformad enligt ”*boxplot*” kommer att användas för att visa spridning på insamlad data för varje experiment. ”*Boxplot*” kan användas för att undersöka om en distribution är normalfördelad, är detta fallet så är det möjligt att använda medelvärde och varians för att jämföra resultat. Denna metod baseras på beräkning av den minsta (min) observationen, första kvartil q1 (25%), median (50%), tredje kvartil q3 (75%) samt den högsta (max) observationen av data uppställd i stigande eller fallande ordning, Moore (2001). Grafer för ”*boxplot*” kommer att skapas enligt metod utan ”*outliner*” som anges av Hunt (2007). Se Figur 4.



Figur 4. Exempel på ”*boxplot*” och dess beståndsdelar, Hunt (2007).

Experimentet kommer att använda sig av FTP för filöverföring detta ger en realistisk bild av ett verktyg som används dagligen, vid t.ex. webbhotell. Som representant för fullständig systemvirtualisering kommer VMware Server 1.0.5 att användas. Vid skrivandet av denna rapport så är det version 1.0.5 som är den senaste versionen. Valet till att använda VMware är att de har en något längre erfarenhet av virtualisering, då de var bland de första att leverera virtualiseringsprodukter för både desktop och server marknaden (1999). Samt att VMware Server är väl etablerad på marknaden som en god lösning på fullständig systemvirtualisering, King et al. (2002). Representant för parallell systemvirtualisering är Xen. När denna rapport skrivs är Xen 3.2 den senaste stabila utgåvan och kommer att användas i experimentet. Motiveringen till att använda Xen är att den är vanligt förekommande på server marknaden för parallell systemvirtualisering, Ogden (2006). Övrigt så finns fri support med ett stort användarsamfund inom Xen. Xen är av s.k. öppen källkod och släpps under ”*GNU General Public License*” (GPL) detta innebär att mjukvaran är fri att använda och ladda ner. VMware Server är fri att använda men kommer som proprietär kod d.v.s. sluten källkod. Operativsystem som kommer att användas för experimenten kommer att bestå av GNU/Linux Ubuntu 6.10 LTS. Ubuntu är fri att ladda ner och använda och är utvecklad med säkerhet i åtanke och levererar gratis säkerhetsuppdateringar i minst 18 månader. Värdoperativsystem och gästoperativsystem i experimentet kommer att bestå av Ubuntu 6.10 LTS Server

Edition. Denna har minimalt med mjukvaror och tjänster installerade från basis och då också lägre resursnyttjande av hårdvaran, detta bidrar också till ett ökat säkerhetstänkande. Vidare så är releasen Server Edition av LTS (Long Term Support), vilket gör den intressant för organisationer då den erbjuder utökad support och säkerhetsuppdateringar till 2011. Användandet av Server Edition återspeglar ett scenario där det just är resurssnåla serveroperativsystem som får agera virtualiseringsservrar i t.ex. webbhotell. Den klient som kommer att användas i experimentet kommer att bestå av en lokal installation av Ubuntu 6.10 LTS Desktop Edition. Motiveringen till denna är att simulera en normal desktop klient som t.ex. ansluter och laddar ner filer från ett webbhotell via FTP-protokollet. Den FTP-servermjukvara som kommer att användas kommer att bestå av Vsftpd 2.0.6. Då den är en av de FTP-servrar som är stabil, snabb och säker. Mjukvaran är fri och går under GPL licensen. Vid skrivandet av denna rapport är det Vsftpd 2.0.6 som är senaste stabila versionen. Denna mjukvara är en representation för en FTP-applikation som används inom en organisation vilket ofta kräver en högre grad av säkerhet.

4.3 Hårdvara

4.3.1 Server

- Processor: E6750 Intel Core2 Duo 2.66GHz, socket 775.
- Moderkort: Asus P5N-E SLI, chipset Nvidia nForce 650i SLI.
- Minne: 2x 1GB 800MHz Corsair XMS 6400 DDR2, Dual Channel.
- Hårddisk: 2x Western Digital Raptor 74GB 10 000rpm 16MB cache SATA.
- Ljudkort: Creative X-Fi 64MB PCI.
- Grafikkort: Nvidia 8800GT 512MB PCIe.

Experimentet kommer att utföras utan användandet av någon form av RAID (”*Redundant Array of Independent Disks*”). I BIOS (”*Basic Input Output System*”) under processor konfiguration avaktiverades ”*Virtualization Technology*” d.v.s. hårdvarustödet för virtualisering och ”*Enhanced Intel SpeedStep Technology*”, då detta är en teknik som ändrar processor hastigheten beroende på belastning. Och dessa två teknologier skulle eventuellt påverka experimentet.

4.3.2 Klient

- Processor: AMD Athlon 64 3800+, socket 939.
- Moderkort: Asus A8N-E Nvidia nForce 4 Ultra.
- Hårddisk: Samsung SP0812N 7200rpm 8MB cache 80GB PATA 133.
- Grafikkort: 3Dfx Voodoo3 2000 16MB PCI.
- Minne: 2x 512MB RAMBOO PC3200 DDR400, Dual Channel.

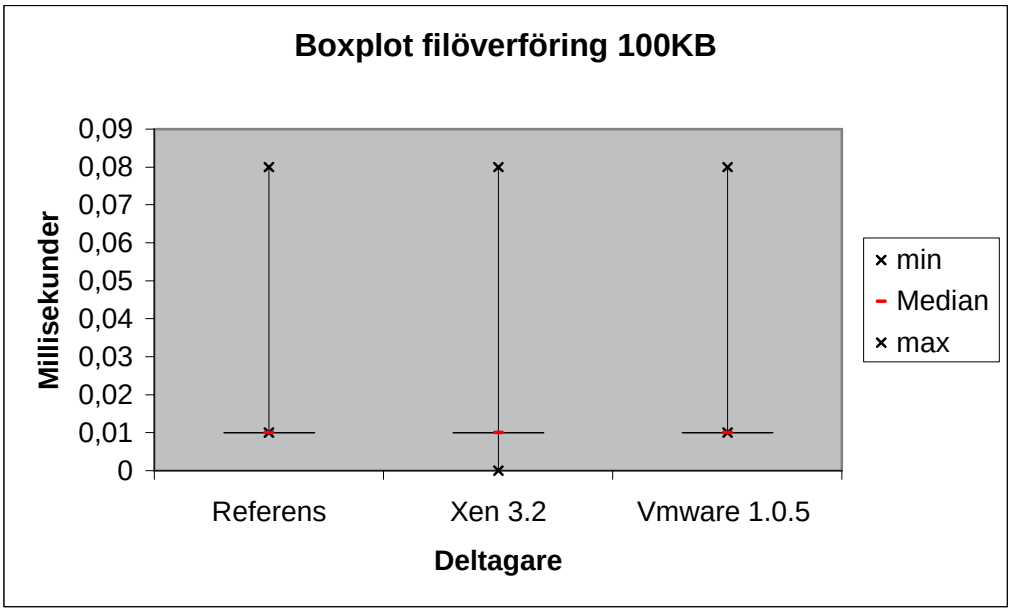
4.3.3 Nätverk

- Asus Multi-Function Wireless Router WL-500g Premium.
- 2x kategori 5 UTP.

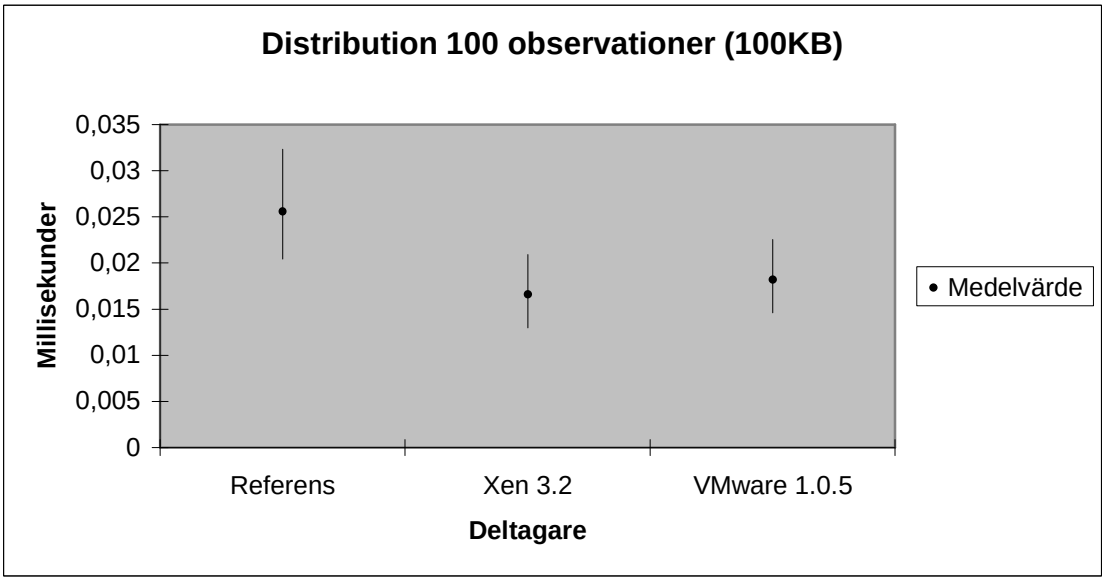
5 Resultat

Sektion 5.1 presenterar resultat för de data som insamlats. Dessa kommer att representeras via grafer som t.ex. ”*boxplot*”. Vidare i sektion 5.2 så kommer en analys av data att utföras för att besvara de frågor som ställdes i problemställningen (kapitel 3, sektion 3.1). I kapitel 6 ges slutsatsen för experimentet och sektion 6.1 diskuterar framtida arbeten.

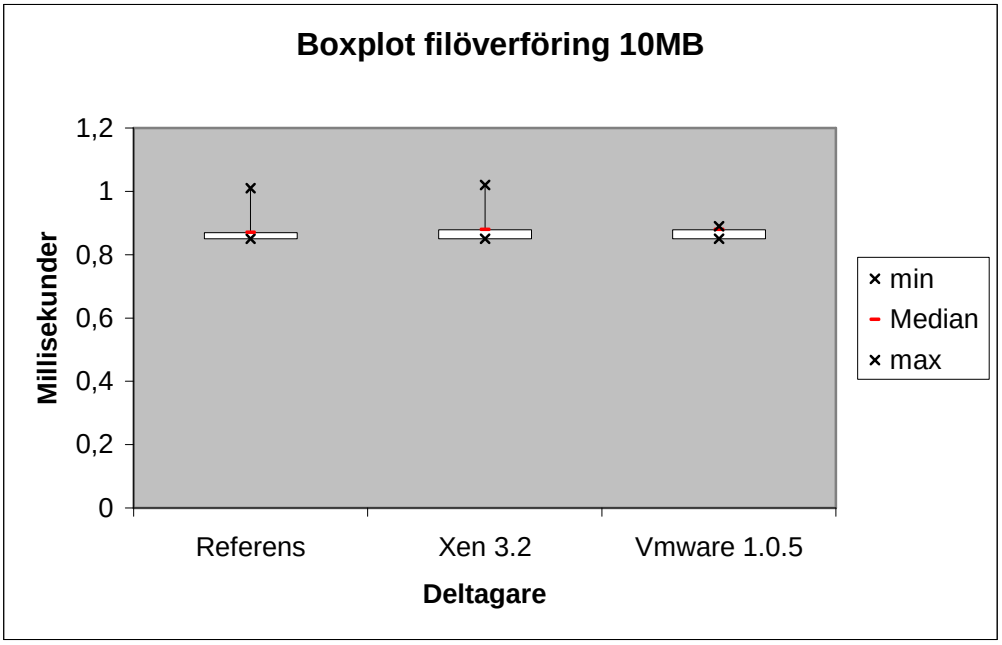
5.1 Experiment



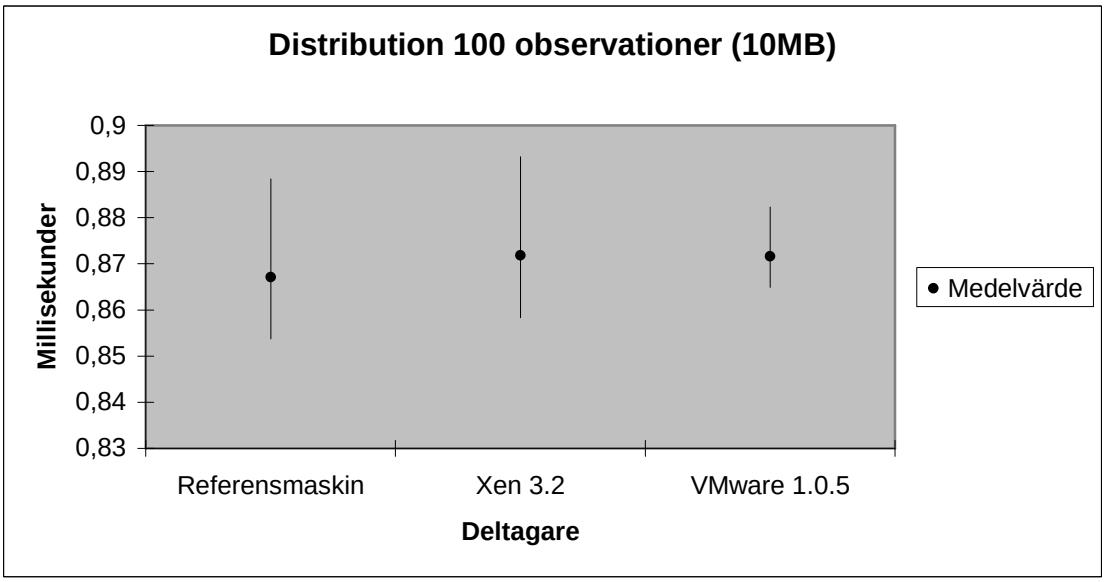
Figur 5. Boxplotdiagram spridning vid 100KB.



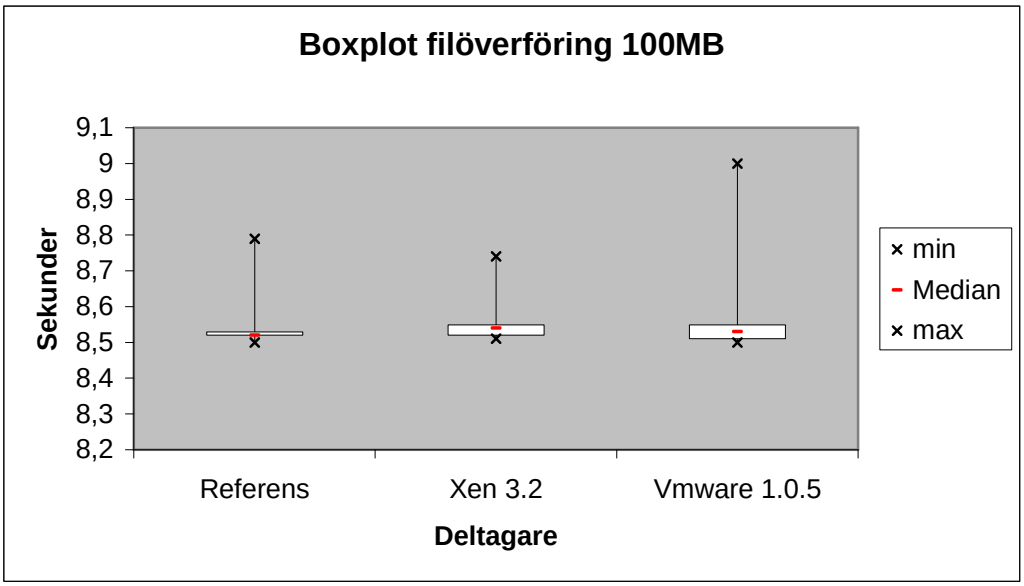
Figur 6. Medelvärde med 95% konfidensintervall för 100KB.



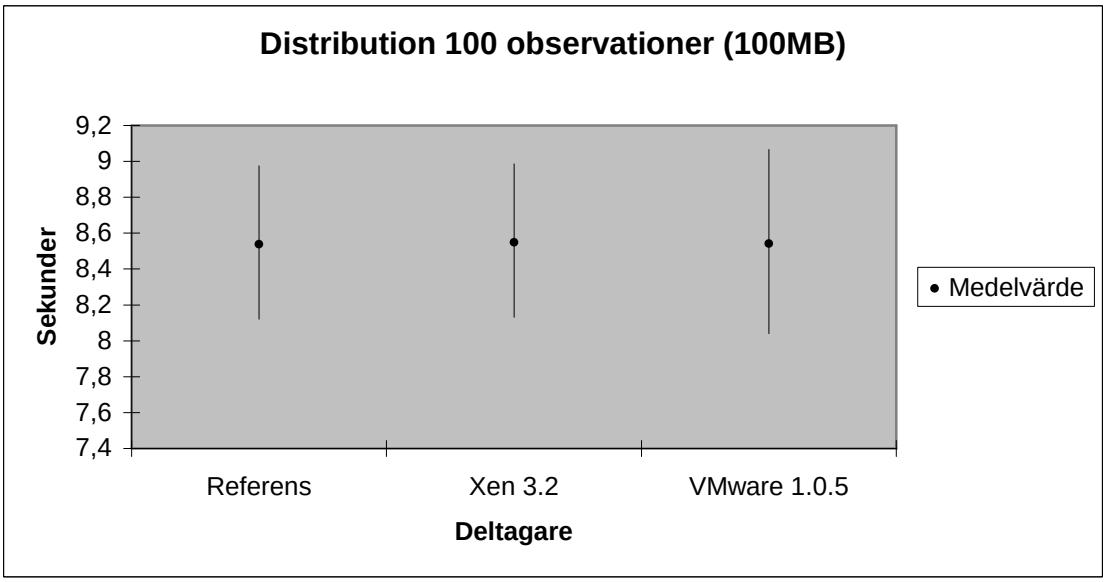
Figur 7. Boxplotdiagram spridning vid 10MB.



Figur 8. Medelvärde med 95% konfidensintervall för 10MB.



Figur 9. Boxplotdiagram spridning vid 100MB.



Figur 10. Medelvärde med 95% konfidensintervall för 100MB.

5.2 Analys

Resultaten som presenterats ger svar på de frågor som ställts i problemställningen i kapitel 3, sektion 3.1. Vilken av fullständig och parallell systemvirtualisering presterar bäst över FTP-protokollet? För de olika filöverföringar som utförts i detta experiment kan slutsatsen dras att skillnader mellan de båda teknologierna är ej signifikant. Varför det är så beror förmodligen på att FTP-filöverföring för de virtuella maskinerna som används i experimentet påvisar en alldeles för låg belastning på bland annat minne och processor. Problemet är att överföring av fil från hårddisk är markant långsammare än minne och detta ger då inte den önskade belastning som krävs för att fylla upp minne med data som sedan leder till att processer måste växlas in och ut vid överbelastning. För att lyckas med detta måste flera virtuella maskiner på en virtualiseringsserver belastas och använda de resurser som finns till max, för att de virtuella maskinerna sedan skall kunna kämpa om tillgängliga resurser. Även om det då finns en minimalt mätbar skillnad mellan de båda teknologierna i experimentet så är den så liten att de upplevs erbjuda likvärdig prestanda. En ytterliggare slutsats är att de båda teknologierna vid experimentet har exklusiv tillgång till hårdvaran och behöver då inte konkurrera med andra virtuella maskiner som samkör parallellt. Detta gör att de har möjligheten att erbjuda bättre prestanda än vad som är normalt under drift av flera virtuella maskiner på server. En ökad belastning på hårdvara (processor och minne) vid användning av flera virtuella maskiner kan ge ett annat resultat än det som experimentet i denna rapport visat. Men det kan konstateras att i nuvarande versioner enligt experimentet är båda representanterna likvärdiga och kan tas i åtanke vid ett potentiellt införskaffande. Detta kan ses i graf då konfidensintervall visar att data är överlappande. Vilken påverkan har olika filstorlekar för prestandan? Skillnaden mellan de filstorlekar som används i experimentet visar att det är en minimal skillnad. Större filer visar en ökad filöveföringstid p.g.a. det tar längre tid för fil att överföras över nätverket. Alternativt kan det vara så att implementationen av de båda teknologierna är så pass lika att en skillnad vid filöverföring inte är möjlig för de båda. Är 128MB minne tillräckligt för en virtuell maskin? För de experiment som utförts kan det konstateras att denna mängd minne är fullt tillräcklig. Skulle mängden minne visat sig vara otillräcklig skulle detta ha visat en kraftig påverkan på de resultat som presenterats, detta då en mängd ”*swapping*” och ”*trashing*” skulle uppstå med fördröjd filöverföring som resultat. Om FTP-filöverföring skulle ske med filer på storlekar över t.ex. 1GB eller flera FTP-klienter som ansluter parallellt till server så skulle kanske detta vara en otillräcklig mängd minne med fördröjd tid för filöverföringar att slutföras.

6 Slutsats

Denna rapport har undersökt skillnaden för FTP-filöverföringsprestanda mellan fullständig och parallell systemvirtualisering. Experiment har utformats för att mäta skillnad för överföring av filer på, 100KB, 10MB och 100MB. Experimentet har använt operativsystemet GNU/Linux Ubuntu för server och klient. På server har FTP-mjukvaran Vsftpd används. Klient har sedan via ett automatiserat skript laddat ner filer i 100 till antal för varje experiment. Ytterliggare ett skript användes för att beräkna medelvärde. Metoder som ”*boxplot*” och konfidensintervall har används för att representera spridning av data med en skattad säkerhet inom 95% för resultaten. Resultat som presenterats har visat att skillnader mellan de båda teknologierna är näst intill omärkbar och de båda är bra lösningar på virtualisering i kombination med FTP-filöverföring för filer på 100KB-100MB. Författaren anser att experimentet uppvisade inga större skillnader och det går då inte att besvara problemställningen, de båda produkterna leverera likvärdig prestanda. Vid ett eventuellt val mellan dessa så är detta upp till användaren om det önskas använda mjukvara som är av sluten eller öppen källkod. Vidare finns andra åtaganden att ta i betänkning så som tillgång till kompetent support och administration av dessa teknologier, men detta tas inte upp av denna rapport. Detta skulle kunna vara ett framtida arbete på en jämförelse vilket kan bidra till att underlätta beslutsprocess vid införskaffande av virtualiseringsprodukter. Det är en snabb utveckling mellan Xen och VMware och nya versioner släpps regelbundet det är då möjligt att större prestanda skillnader mellan de båda kan komma att ske inom en överskådlig framtid. Därför är det viktigt att experiment som denna förbättras och upprepas för att hjälpa till att fastställa vilken teknologi som levererar bästa prestanda och på så vis förbättra och förmedla kunskapen vidare till alla som kan tänkas dra nytta av virtualisering.

6.1 Framtida arbeten

Resultatet från rapporten och experimentet skulle kunna utökas med experiment som undersöker prestanda för FTP-filöverföring mellan Xen och VMware under belastning av flera samkörande virtuella maskiner eller klienter som parallellt ansluter och påbörjar filöverföringar. Detta för att skapa en högre belastningsgrad vilket kan ge tydligare resultat än de som experiment i denna rapport visade. Alternativt skulle experiment kunna utföras som jämför belastningspåverkan vid FTP-filöverföring i virtuella maskiner vid användandet av SSL/TLS kryptering. En litteraturstudie kring virtualisering kan också vara en möjlighet för att utröna t.ex. vilken teknik är mest kostnadseffektiv med avseende på administration, support och utbildning? eller vilken påverkan på miljön som virtualisering kan tänkas ha?

7 Referenser

Chen, Peter M,. King, Samuel T. (2002). Operating System Extensions to Support Host Based Virtual Machines. Department of Electrical Engineering and Computer Science. Technical Report CSE-TR-465-02.

Dragovic, Boris., Hand, Steven., Harris, Tim., Ho, Alex., Fraser, Keir., Neugebaurer, Rolf., Pratt, Ian., Warfield, Andrew. (2003). Xen and the Art of Virtualization. Microsoft Research and Intel Research, *ACM SOSP Symposium, October 2003*.

Rose, Robert. (2004). Survey of System Virtualization Techniques. [elektronisk]. CiteSeer. Tillgänglig: http://www.cs.ucsb.edu/~ckrintz/papers/systemvm-paper-rose04.pdf. [2008-05-07].

Understanding Full Virtualization, Paravirtualization, and Hardware Assist. (2007). [elektronisk]. VMware. Tillgänglig: http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf. [2008-04-30].

A Performance Comparison of Commercial Hypervisors. (2007). [elektronisk]. Citrixxenserver. Tillgänglig: http://www.citrixxenserver.com/Documents/hypervisor_performance_comparison_1_0_5_with_esx-data.pdf. [2008-04-30].

Huang, Qian. (2006). An Introduction to Virtual Machines Implementation and Applications. [elektronisk]. The University of British Columbia. Tillgänglig: http://www.cs.ubc.ca/grads/resources/thesis/May07/Huang_Qiang.pdf. [2008-04-30].

ComputerSweden [elektronisk]. Tillgänglig: http://www.idg.se/2.1085/1.147732. [2008-03-05].

Hunt, Neville. (2007). Boxplots in Excel. [elektronisk]. Coventry University. Tillgänglig: http://www.coventry.ac.uk/ec/~nhunt/boxplot.htm. [2008-06-11].

Bullers Jr, William l & Burd, Stephen & Seazzu, Alessandro F. (2006). Virtual Machines – An Idea Whose Time Has Returned: Application to Network, Security, and Database Courses. *ACM SIGCSE Bulletin, Volume 38, Issue1, (March 2006)*. ISSN:0097-8418.

Wolf, C & M. Halter, E. (2005). *Virtualization From the Desktop to the Enterprise.* Apress. ISBN 1-59059-495-9.

Robin, J & Irvine, C. (2000). Analysis of the Intel Pentium´s Ability to Support a Secure Virtual Machine Monitor. *Proceedings of the 9th USENIX Security Symposium, volym(9), 129-144.*

Soltesz, S & Pötzl, H & Fiuczynski, M & Bavier, A & Peterson, L. (2007). Container-based Operating System Virtualization: A Scalable, High-performance Alternative to Hypervisors. *Proceedings of the ACM SIGOPS/EuroSys European Conference on Computer Systems 2007.*

Menascé, D. (2005). Virtualization: Concepts, Applications, And Performance Modeling. George Mason University Fairfax. Int. CMG Conference, 2005, 407-414.

Marshall, D & Reynolds, W & McCrory, D. (2006*). ADVANCED SERVER VIRTUALIZATION VMware and Microsoft Platforms in the Virtual Data Center.* Auerbach Publications. ISBN 0-8493-3931-6.

Williams, D & Garcia, J. (2007). *Virtualization with Xen Including XenEnterprise, XenServer and XenExpress.* Syngress Publishing, Inc. ISBN 1597491675.

Reuben, J. (2007). A Survey on Virtual Machine Security. Helsinki University of Technology. Tillgänglig: www.tml.tkk.fi/Publications/C/25/papers/Reuben_final.pdf. [2008-04-30].

Goldworm, B & Skamarock, A. (2007). *Blade Servers And Virtualization.*Wiley Publishing Inc. ISBN 0471783951.

Fisher-Ogden, J. (2006). Hardware Support for Efficient Virtualization. UCSD CSE Graduate Research Exam. Tillgänglig: http://www.cse.ucsd.edu/~jfisherogden/hardwareVirt.pdf. [2008-04-30].

Joe, C. (2003). *Sams Teach Yourself TCP/IP in 24 Hours.* Sams. ISBN 0672325659.

Opsahl, C. (2007). A Comparison of Management of Virtual Machines with z/VM and ESX Server Master thesis. University of Oslo Department of Informantics. URN:NBN:no-15037.

Goldberg, R.P. (1973). Architectural principles for virtual computer systems Thesis. Harvard University. ESD-TR-73.105.

Popek, G.J & Goldberg, R.P. (1974). Formal Requirements for Virtualizable Third Generation Architectures. *Communications of the ACM, 17, 7, 412-421.*

Wlodarz, J.J. (2007). Virtualization: A double-edged sword. Faculty of Mathematics, Physics and Chemistry, Silesian University. arXiv:0705.2786v1.

Moore, David S. (2001). STATISTICS Concepts and Controversies. 5 uppl. Purdue University: Litteratur. ISBN: 0-7167-4008-7.

Råde Lennart & Westergren Bertil. (1988). BETA Mathematics Handbook. Studentlitteratur. ISBN: 91-44-25051-7.

Bilagor

Tabell 1. Representation över den konfiguration som används för Xen i experimentet.

Konfiguration Xen 3.2		
	Värdmaskin (Domain 0)	Virtuell maskin (Domain U)
Operativsystem	GNU/Linux Ubuntu Dapper 6.06 LTS Server Edition.	GNU/Linux Ubuntu Dapper 6.06 LTS Server Edition.
Kärna	2.6.18.8	2.6.18.8
Minne	2GB	128MB
Root Partition ”/”	10GB EXT3	2GB EXT3
Swap Partition	3GB	512MB

Tabell 2. Representation över den konfiguration som används för VMware i experimentet.

Konfiguration VMware 1.0.5		
	Värdmaskin	Virtuell maskin
Operativsystem	GNU/Linux Ubuntu Dapper 6.06 LTS Server Edition.	GNU/Linux Ubuntu Dapper 6.06 LTS Server Edition.
Kärna	2.6.18.8	2.6.18.8
Minne	2GB	128MB
Root Partition ”/”	10GB EXT3	2GB EXT3
Swap Partition	3GB	512MB

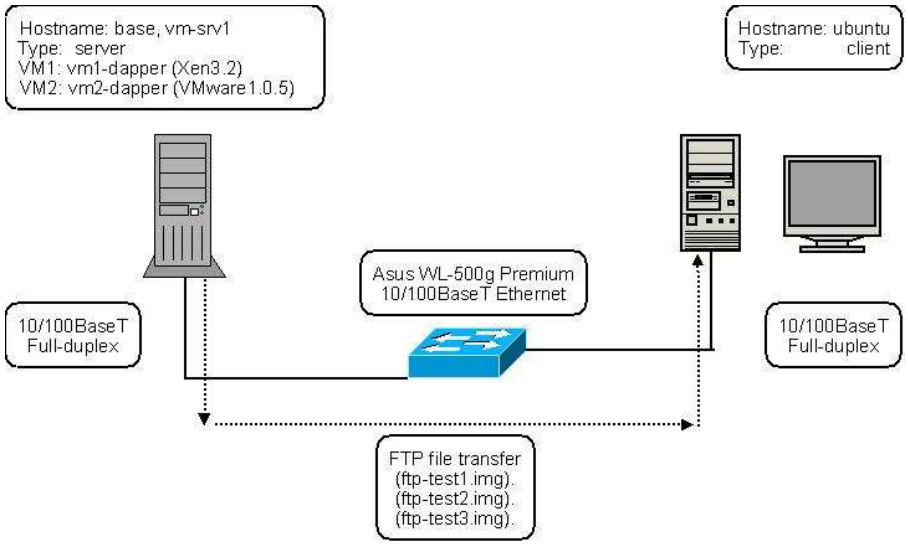
Tabell 3. Representation över den konfiguration som används för grundtest. För att erhålla basdata som sedan kan jämföras mot Xen och VMware.

Konfiguration Grundtest	
	Värdmaskin
Operativsystem	GNU/Linux Ubuntu Dapper 6.06 LTS Server Edition.
Kärna	2.6.18.8
Minne	2GB

Root Partition ”/”	2GB EXT3
Swap Partition	512MB

Tabell 4. Konfiguration för klient som används i experimentet för att ladda ner FTP-data från server bestående av Xen och VMware.

Konfiguration Klient Värdmaskin	
Operativsystem	GNU/Linux Ubuntu Dapper 6.06 LTS Desktop Edition.
Kärna	2.6.15-26
Minne	1GB
Root Partition ”/”	Automatiskt (allt tillgängligt utrymme).
Swap Partition	Automatiskt (allt tillgängligt utrymme).



Figur 11. Illustration över topologi vid experiment.

```
#!/bin/sh

TIMES='100'
SERVER='192.168.1.3'
LOG='/tmp/FTP.log'
RESULT='/tmp/RESULT.log'

echo "Run test: 1 = Baseline, 2 = Xen, 3 = VMware" & read -p "Choose: " opt

if [ "$opt" == "1" ] ; then
      echo Running: & r="Baseline"
fi
if [ "$opt" == "2" ] ; then
        echo Running: & r="Xen"
fi
if [ "$opt" == "3" ] ; then
        echo Running: & r="VMware"
fi

i=1
while [ $i -le $TIMES ];do
echo "FTP-TEST" $i $r & echo "FTP-TEST" $i $r >> $LOG
ftp -n -i -v "$SERVER"<<"EOF" >> $LOG
quote USER magnus
quote PASS magnus
get ftp-test1.img
EOF
mv ftp-test1.img ftp-test1-$i.img
let i++
done

clear & echo "Result:"
sed -n '/FTP-TEST/p;/secs/'p /tmp/FTP.log & sed -n '/FTP-TEST/p;/secs/'p /tmp/FTP.log > $RESULT
```

Figur 12. Skript för FTP-filöverföring från server till klient.

```
#! /bin/sh
#
# Skript som läser igenom en resultatfil, summerar och räknar ut medelvärdet.
#
file=/home/magnus/RESULT.log

#Skriv ut resultatfil, ta bort blankrader och skicka till tmp.log
cat $file | awk '{print $5}' | sed /^$/d  > tmp.log

#Skriv ut tmp.log och summera siffror på kolumn 1 i fil, räkna även ut medelvärdet.
cat tmp.log | awk '{s += $1 } END { print "Sum: ",s, "Average: ", s/NR }'

#Skriv ut och räkna ut hur många rader som filen innehåller.
#cat tmp.log | awk 'END {print NR}'
```

Figur 13. Skript för beräkning av medelvärde.