BACHELOROPPGAVE:

YTELSESTESTING AV VIRTUALISERINGS TEKNOLOGIER

FORFATTER(E):

Kristoffer M. Elde
Øyvind Haugedal

Dato:

19.05.2010

SAMMENDRAG AV BACHELOROPPGAVEN

Tittel:	Ytelsestesting av virtualiserings teknologier	Nr. :
		Dato : 19.05.10
Deltaker(e):	Kristoffer M. Elde	
	Øyvind Haugedal	
Veileder(e):	Erik Hjelmås	
Oppdragsgiver:	IT avdelingen ved HiG	
Kontaktperson:	Jon Langseth	
Stikkord (4 stk)	Virtualisering, Xen Server 5.5, VMware ESXi 4, Ytelsestesting	

Antall sider: 196	Antall bilag: 121	Tilgjengelighet (åpen/konfidensiell):

Kort beskrivelse av bacheloroppgaven:

Dette prosjektet går ut på å ytelsesteste virtualiserings servere av typen VMware ESXi 4 og Xen Server 5.5 med forskjellige server applikasjoner. I tilegg til ren ytelsestesting av applikasjonene kombineres testene med parallel belastning fre et egent oprativsystem kun ment for produsering av cpu last i egendefinerte nivåer.

Selve belastnings testene ble utviklet i perl med bruk av noen ferdig applikasjoner. Siden flere oprativsystemer er involvert i testen ble det også laget scipt i perl for å synkronisere kjøringene til testene via nettverk.

I rapporten presenteres og diskuteres test resultater som gruppen selv har fått for å kjøre testene på egen server. Test oppsettene og scriptene blir også igjennomgått i rapporten.

Forord

Ved vår tid ved høgskolen i Gjøvik har virtualisering vært et viktig verktøy for løsing av fag prosjekter og oppgaver. Med dette har vi fått erfare hvor nyttig verktøy virtualisering er for bespare tid ved bruk av kloning og ikke minst behovet for fysisk hardware. Som studenter med fokus på it drift vekte derfor oppgaven store interesse etter som den fokuserte på rene virtualiserings servere og muligheten for å kunne ytelses teste disse med reelle server applikasjoner. I oppgaven prøver vi å teste i en realistisk settning hvor server tjenestens gjeste operativsystem ikke er alene om tildelingen av hardware ressursene. Dette gjøres med å kjøre bestemte belastninger fra et annet gjeste operativsystem. Ved å regulere belastningene kan vi se hvordan belastningsnivåene påvirker ytelsen en server applikasjon leverer.

Vi vil i denne rapporten presentere hvilke applikasjoner vi har valgt å teste, vår test metode igjennom bruk av script og hvilke resultater vi sitter igjen med etter kjøring av scriptene.

Vi ønsker for øvrig å takke for hjelp og veiledning vi har fått fra vår veileder
Erik Hjelmås

Vi ønsker også å takke vår arbeidsgiver
Jon Langseth
for oppgaven, råd og hjelp med script.

Oversikt over figurer

Oversikt over tabeller

1 INNLEDNING

1.1 INNHOLD I RAPPORTEN:

Kapittel 1 Innledning

Dette kapittelet er en introduksjon til resten av rapporten. Her får leseren en innføring i oppgaven og strukturen rundt prosjektet.

Kapittel 2 Bakgrunn/Virtualisering - Teori

Kapittelet omhandler teorier og prinsipper rundt virtualisering, som er en del av grunnlaget for prosjektet.

Kapittel 3 Testing – Teori

I dette kapittelet finnes det prinsipper og teorier om testing og benchmarking generelt. Det blir her pekt på viktige synspunkter som legger grunnlag for valg som blir gjort videre.

Kapittel 4 Utstyr

Denne delen er en gjennomgang av utstyr(hardware og software) som er benyttet under prosjektet. Det blir lagt vekt på virtualiserings serverne VMware esxi 4.0 og Citrix XenServer 5.5 samt applikasjonene som bruks for å kontrollere disse.

Kapittel 5 Utførelse

Gjennomgang av skript og testsenarioer. Samt beskrivelse av hva som skal testes, hvordan de skal gjennomføres og hvordan testene funger slik at dette kan utføres av oppdragsgiver dersom det er ønskelig.

Kapittel 6 Resultater

I dette kapittelet presenteres resultatene som er generert av test senarioene og skriptene presentert i kapittel 5.

Kapittel 7 Diskusjon av resultater

En diskusjon av resultatene som er representert i kapittel 6.

Kapittel 8 Konklusjon

I dette kapittelet konkluderes det med funnet som er gjort i rapporten. Det ses også på ting som kunne vært gjort annerledes i prosjektet samt hvordan arbeidet har fungert.

Kapittel 9 Litteraturliste

Liste med litteratur og referanser som er benyttet i rapporten.

Kapittel 10 Vedlegg

Vedlegg inneholder Gantt-skjema, skript, installasjons guider, test senarioer, møtereferat, arbeidslogg og terminologiliste.

1.2 OM PROSJEKTET:

1.2.1 OPPDRAGSGIVER

Vår oppdragsgiveren er IT avdelingen[1] ved høgskolen i Gjøvik. Denne avdelingen har ansvar for drifting av om lag 500 klientmaskiner, 80 serverkjørende operativsystemer og for øvrig alle IT tjenester tilgjenglig ved skolen. Av de 80 kjørende server operativsystemene er det kun om lag havparten som kjører alene rett på en maskin. Resterende kjører fordelt over 5 maskiner med virtualiserings plattform[2]. Deres datamaskinpark består av klientmaskiner av Mac[3], Microsoft Windows[4] og Linux[5] basert OS. I tilegg har de ansvar for en rekke klient applikasjoner som epost, kalender og lignende. Avdelingen har allerede virtualiserings plattformen VMware[6] ESX[7] 1 som kjører enkelte av tjenestene avdelingen har ansvar for.

1.2.2 BAKGRUNN FOR OPPGAVEN

It tjenesten ved høgskolen i Gjøvik har per dags dato webbaserte server applikasjoner, database og replikering av e-post servere kjørende virtuelt. Avdelingen ser for seg at de muligens kan virtualisere enkelte filservere i framtiden også. De ønsker nå en oppdatering av dagens kjørende virtualiserings plattformen. De har sett for seg en løsning basert på seneste VMware ESXi[8] eller Citrix XenServer[9], og ønsker da en metode for å teste hvilke av disse plattformene som ville vært mest lønnsom med den forutsetning de har med tanke på hardware og operativsystemer.

1.2.3 OPPGAVEN

IT Tjenesten ved høgskolen i Gjøvik ønsker seg nå en metode for å kunne systematisk stress teste og benchmarke[10] forskjellige server applikasjoner på gjeste operativsystemer, kjørende på en virtualiserings server. Gjeste operativsystemene skal være av både Microsoft Windows og Linux varianter under forskjellige Virtualiserings Plattformer.

It-avdelingen ved hig ønsker å få testet ytelse fra følgende virtualiserings plattformer opp mot hverandre:

- VMware ESXi 4.0
- Citrix XenServer 5.5
- VMware Server 1.x[11]
- Xen 3 (xen.org) on CentOS 5[12,13]

- VMware ESXi 3.5[14]
- Xen 3 on Debian 5[15]

Oppgaven vil være delt i to deler. I den ene delen fokuseres det på metoden for testing som skal utvikles for oppdragsgiver. Resultatet av denne delen vil forklares gjennom hele rapporten.

I den andre delen av oppgaven vil metoden som er utviklet benyttes til å utføre tester på VMware ESXi 4 kontra Citrix XenServer 5.5.

Det skal derfor settes opp og konfigureres en rekke test senarioer som gjør det mulig å utføre målinger av de forskjellige virtualiserings produktene. Her vil gruppen finne målbare verdier som gjør det mulig å sette de forskjellige produktene opp mot hverandre. Ved utvikling av metoden fokuseres det på å etterligne belastninger produktene ville hatt, dersom de hadde vært satt i drift ved it avdelingen ved HiG over lengre tid og hvordan de ville ha taklet oppgaven. På denne måten gi oppdragsgiver mulighet for teste virtualiserings servere med belastning som er relevant for dem.

1. 2.4 AVGRENSINGER

Med begrenset tid vil gruppen ta i bruk ferdig applikasjoner i den grad det er mulig i prosjektet. På de områdene hvor ferdig applikasjonene ikke strekker til hvor prosesser skal automatiseres og data skal hentes ut i riktig format vil det i stor grad bli brukt scripting[16].

Det finnes mange virtualiserings plattformer men gruppen vil kun forholde seg til virtualiserings produktene nevnt i oppgavebeskrivelsen og da spesielt VMware Esxi og Citrix XenServer 5.5. Bruk av gjeste operativsystemer under testing avgrenses til ett Windows basert og ett linux basert opertivsystem.

1.3 MÅL

1.3.1 MÅLGRUPPER

Målgruppen for testmetoden og resultater som gruppen produserer er oppdragsgiveren IT tjenesten ved høgskolen i Gjøvik. Det er ønskelig at resultat og metoden som utvikles kan benytte av andre bedrifter og it avdelinger som befinner seg i samme situasjon som it tjenesten ved høgskolen i Gjøvik.

Rapporten er først og fremst produsert med tanke på veileder og ekstern sensor. Dette valget begrunnes med at det er disse som skal vurdere resultatet. Rapporten skal kunne brukes av oppdragsgiver til å sette seg inn i oppgavens helhet.

1.3.2 EFFEKTMÅL

Effektmålet for oppgaven vil være å utvikle en automatisert metode for stress testing av virtuelle maskinger som kjører på en virtuell server løsning. Som IT tjenesten ved høgskolen i Gjøvik og eventuelt andre aktører kan bruke i sin vurdering av hvilken virtualiserings teknologi som passer deres krav og behov best.

Dette vil i første omgang gi oppdragsgiver test resultater som gjør at de kan ta en avgjørelse på hvilken virtualiserings plattform som gir deres virtuelle maskiner tilfredsstillende ytelse.

1.3.3 RESULTATMÅL

Vårt mål som en gruppe er først å fremst å utvikle og utarbeide en metode for å kunne teste ytelsen til virtuelle servere under forskjellige arbeidsformer og arbeids belastninger. Samt produsere detaljerte resultater som oppdragsgiver og eventuelt andre kan legge til grunn for vurdering av virituelle server plattformer. Test løsningen skal være automatisert i den form at den ikke trenger input fra brukeren under kjøring, men dette forutsetter at test miljøet og konfigurasjon for test er satt opp.

1.3.4 LÆRINGSMÅL

Vi ønsker å legge vekt på å bruke så mye som mulig av den lærdommen vi har tilegnet oss igjennom studiet så langt, og sette det ut i praksis. Og på denne måten tilegne oss ny lærdom om hvordan ting vi kan i teorien, kan brukes i en mer drift praktisk sammenheng. Samt tilegne seg nye teoretiske og praktiske kunnskaper gjennom arbeid med selve oppgaven. I tilegg vil oppgaven gi ansvars erfaring og lærdom med å jobbe systematisk og strukturert.

1.3.5 RAMMER

Prosjektet skal kun forholde seg til virtualiserings plattformer nevnt i oppgavebeskrivelsen av arbeidsgiver. Operativsystemer og programvare for testing vil bli valgt ut fra hva oppdragsgiver bruker virtualisering i dag. Dette gjøres for å sikre relevant belastning. Ytterligere har prosjektet en tidsramme hvor den ferdige rapporten skal være ferdig og levert inn innen 20. mai 2010.

1.4 ROLLER OG ARBEIDSFORM:

1.4.1 ROLLER

Gruppen har som mål at begge skal tilegne seg kunnskap og lærdom om alle deler av prosjektet. Tildelig av spesifiserte roller kan føre til en deling av oppgaven. Der hvert medlem gjør mye arbeid på egenhånd, noe som er i strid med gruppens målsetting. Det velges da å være forsiktig med å gi spesifiserte roller over lang tid en så liten gruppe.

1.4.2 ARBEIDSFORM

Gjennom skolen har gruppen fått lånt gruppe rom. Utstyr lånt av oppdragsgiver til testing er plassert på gruppe rommet og benyttet der. Det er enighet i gruppen om å nedlegge ca 30 timer i uken til arbeid med oppgaven. Arbeidet vil hovedsakelig forgå i tidsrommet 08:00-16:00 mandag til fredag. Gruppen kommer til å arbeide så mye som mulig i fellesskap slik at vi begge kan tilegne seg kunnskap i alle deler av prosjektet.

1.4.3 STATUSMØTER OG BESLUTNINGS PUNKTER

Statusmøter vil bli avholdt sammen med veileder og oppdragsgiver for å gi begge parter en innsikt samt holde en åpen dialog mellom alle involverte parter.

Det er kun to planlagte statusmøter. Dette er for å forsikre at gruppen er på rett spor i starten av prosjektet. Grunnet er kort fysisk avstand mellom gruppens arbeidsrom veileder og oppdragsgiver kan det ved behov enkelt avtales nye statusmøter.

2 BAKGRUNN/TEORI

2.1 HISTORIE:

Virtualiserings teknologien ble utviklet på 60 tallet[18]. Denne teknologien ble da brukt til virtualisering av stormaskiner[29] for å utnytte kapasiteten til maskinene bedre. Virtuelle maskiner ble da opprettet for å speile hovedmaskinene slik at man kunne kjøre flere applikasjoner på samme stormaskin. Etter hvert ble det utviklet "Virtual Machine Monitor" (VMM) [20], dette gjorde det mulig å ha flere virtuelle maskiner på samme fysiske maskin.

På 80 og 90 tallet falt begrepet virtualisering nesten helt bort. Dette hadde med billige x86[21] servere og klint-server tanke gang.

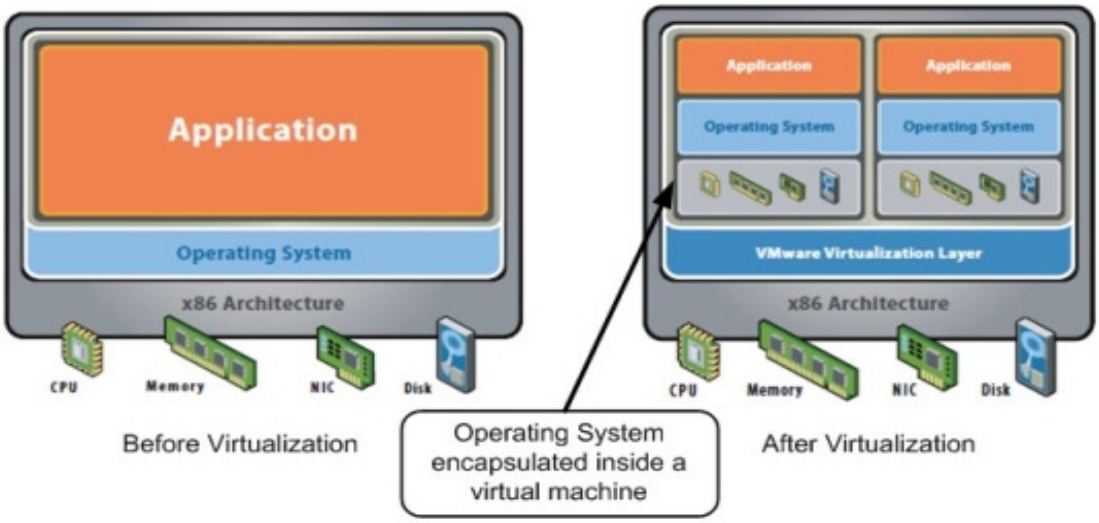
I dag er situasjonen en annen, servere har blitt så kraftige at man ikke får utnyttet alle ressursene. I tillegg er det også ett større fokus på strømforbruk og forurensning. Dette er ting som gjør virtualisering mer aktuelt i dag en det var på 80 og 90 tallet.

I 1999 introduserte selvskapt VMware virtualisering til x86 arkitekturen og har hatt stor suksess med dette. Vmware har lenge vært dominerende på virtualiserings markedet men det er nå kommet inn andre aktører som Citrix.

2.2 HVA ER VIRTUALISERING:

Til vanlig er man vant med at det er ett operativsystem som kjører og kontrollerer alt av maskinens ressurser. Maskiner i dag er mye kraftigere enn de var tidligere, så ved kjøring av kun ett operativsystem får man ikke alltid utnyttet alle ressursene som er tilgjenglige i den fysiske maskinen. Ved og virtualisere kan man bedre dra full nytte av maskinens ressurser.

Virtualiseringen forgår ved at det er en programvare som kontrollerer de fysiske ressursene til maskinen. Denne programvaren gir en logisk presentasjon av den faktiske hardwaren som befinner seg på den fysiske maskinen. Slik at ressurser kan bli kontrollert og behandlet på en slik måte at det er bedre tilpasset hver enkelt maskin som ligger oppe på virtualiserings programvaren. Dette kan for eksempel gjøre det enkelt å tildele maskiner mer ressurser etter behov, på en slik måte at ingen ressurs blir overflødig[22].



Figur 1 – Maskin før og etter virtulaisering

Figur 1 viser forskjellen på en vanlig maskin og en maskin som virtualiserer flere operativsystemer.

Den store forskjellen på disse er at det er ett virtualiserings lag på maskinen som virtualiserer. Dette laget eller programvaren ble nevnt tidligere, det blir også kalt Virtual Machine Monitor(VMM).

En virtuell maskin(VM) er ett miljø som er lagt av en VMM. Den ser hardwaren som er emulert av VMM som ekte og er ikke klar over at den selv er virtuell. Dette er fordi VMM gir VM full tilgang til alle deler av den fysiske hardware. VM lever da i ett fullstendig isolert miljø, som fører til at den oppfører seg helt likt en vanlig fysisk maskin.

Forskjellen ligger som nevnt i at det er VMM som emulerer logisk hardware til VM. Slik at den ikke har tilgang til mer resurser enn det VMM har gitt den.

I 1974 ble artikkelen "Formal Requirements for Virtualizable Third Generation Architectures"[23] lansert av Gerald J. Popek og Robert P. Goldberg. Selv om det er mer enn 35 år siden denne ble utgitt, er ikke denne artikkelen noe mindre riktig i dag.

I artikkelen definerer de en virtuell maskin som

> *"A virtual machine is taken to be an efficient, isolated duplicate of the real machine"*

De har også beskrevet en VMM med følgende beskrivelse

> *" First, the VMM provides an environment for the programs which is essentially identical with the original machine; second, the programs run in this environment show at worst only minor decrease in speed; and last, the VMM is in complete control of system resources"*

De beskriver også tre viktige egenskaper når de analyserer ett miljø som er laget av en VMM:

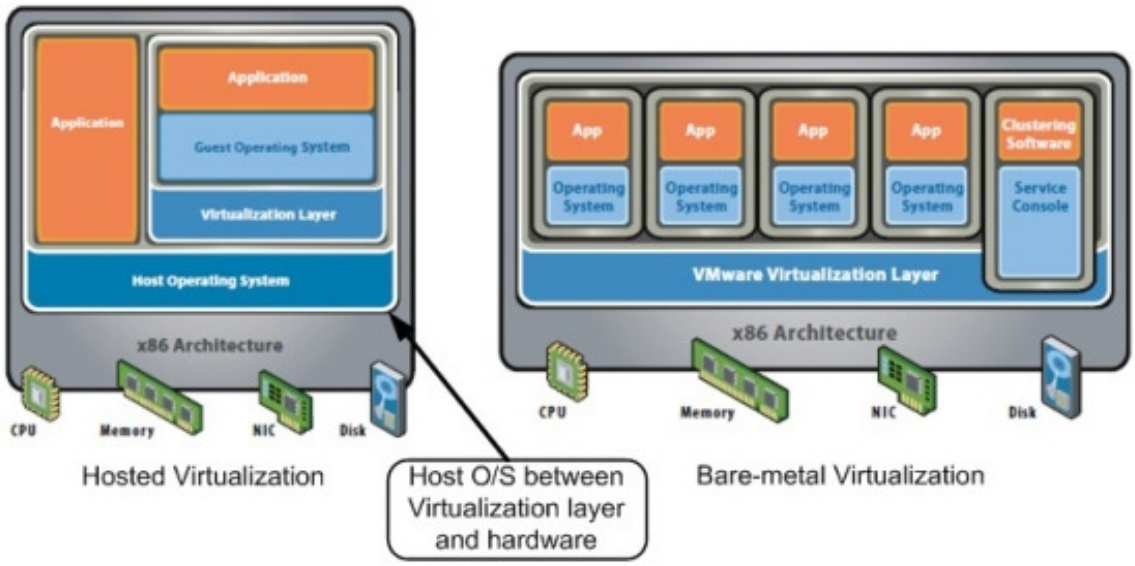
> *"1. Equivalence: a program running under the VMM should exhibit a behavior*
>
> *essentially identical to that demonstrated when running on the original machine directly.*
>
> *2. Resource control: the VMM must be in complete control of the virtualized resources.*
>
> *3. Efficiency: use the native hardware of the physical machine to as great a degree as possible."*

Man antar at dagens VMM`er fullstendig stiller kravet til punkt 1 og 2 i Popek og Goldbergs artikkel og delvis punkt 3, men dette avhenger av hvilken virtualiserings teknologi som benyttes.

2.2.1 FORSKJELLIGE TYPER VIRTUALISERING:

For å kunne virtualisere må vi som nevnt ha virtualisering software, også kjent som VMM eller en hypervisor[2]. Det en hypervisor gjør er å emulere datamaskin hardware, slik at flere operativsystemer kan kjøre på en og samme fysiske maskinen samtidig. Disse operativsystemene tror selv at de har ekte hardware og kan kontrollere dette selv, men faktisk er det hypervisoren som kontrollerer hva og hvor mye ressurser hver virtuelle maskin får.

Det finnes to forskjellige typer hypervisorer. Hypervisor type 1 og type 2 eller bare-metall og hosted. Vi skal ta for oss og forklare forskjellen på disse to.



Figur 2 – Viser forskjell på hosted virtualisering og Bare-metal virtualisering.

Hypervisor 1

Hypervisor 1 eller bare-metall virtualisering legger ett virtualiserings lag direkte på hardware som vist i figur 2. Dette er noe som gir hypervisoren direkte tilgang til hardware. Eksempler på denne typen programvare kan være Vmware EsXi og Citrix

XenServer. Det finnes ingen standard definisjon på hypervisor type 1, men i følge IBM[24]:

> *" a Type 1 hypervisor as running directly on the hardware with VM resources provided by the hypervisor".*

Hypervisor 2

Hypervisor 2 eller hosted virtualisering har ett vanlig operativsystem direkte på hardware og ett virtualiseringslag oppe på operativsystemet igjen. Eksempler på denne typen hypervisor er Sun VirtualBox, VMware Server og Microsoft Virtual PC.

Det finnes heller ingen standard definisjon til denne typen hypervisor men i følge IBM[25]definerer de:

> *" a Type 2 hypervisor runs on a host operating system to provide virtualization services".*

2.2.2 VMM TEKNIKKER

Forskjellen på hypervisor 1 og 2 har kort sagt å gjøre med hvor virtualiseringslaget (VMM) ligger på maskinen.

Men det er mange forskjellige måter for en VMM å presentere virtuelt hardware på. Hvilken måte VMM velger å presentere virtuelt hardware på kommer an på hva slags underliggende fysisk hardware maskinen har.

En måte å presentere logisk hardware på er ved å bruke en teknikk som heter full virtualisering[26]. For at man skal kunne ta i bruk denne teknikken må man ha hardware som støtter dette. Støtte for full virtualisering har aldring vært en del av den populære x86 arkitekturen. Dette har ført til at man har brukt andre metoder for og virtualisere kjøring av spesifikke instruksjoner.

X86 arkitekturen møter ikke Popek og Goldberg`s krav om virtualisering. Dette er fordi at VMM mister kontrollen over ressursene dersom operativsystemet som hoster VMM forsøker å utføre privilegerte operasjoner, som for eksempel forsøke å deaktivere interupts.

Selv om dette er tilfellet for x86 arkitekturen er det fortsatt mulig å utføre effektiv virtualisering på den. Dette er mulig fordi x86 arkitekturen støtter 4 forskjellige privilegerte nivåer[27]. Disse nivåene blir kaldt "ring", og de går fra ring 0 til ring 3. Ring 0 har høyst prioritet og ring 3 har lavest.

Til vanlig er det operativsystemet som kjører i ring 0, utfører og kjører all sin kode der, mens applikasjoner som kjører inne i operativsystemet kjører sin kode på ring 3. Dersom operativsystemet modifiseres slik at det kan kjøre på ring 1 og VMM kan kjøre på ring 0 vil man få ett system som ikke vil ta fra VMM kontrollen, fordi kode som kjøres i ring 1-3 ikke har rettigheter til dette, og all kode må da kjøres gjennom VMM.

Det er denne typen modifisering av operativsystemer som benyttes ved paravirtualisering[28].

Man kan si at en CPU arkitektur er ”virtualizable” dersom den støtter en grunnlegende VMM teknikk med direkte eksekvering. Dette betyr at man lar virtuelle maskiner eksekvere kode direkte på den ekte maskinen samtidig som det er VMM som har den faktiske kontrollen over CPU en.

For CPU arkitektur der som ikke er ”virtualizable” blir teknikkene paravirtualisering og binær oversetting[29] benyttet.

Full virtualisering:

Denne typen for virtualisering kan oppnås når en VMM simulerer hele den underliggende hardwaren til den ekte maskinen slik at umodifiserte operativsystemer og dets applikasjoner tror de kjører på det som ser ut som den ekte hardwaren. Operativsystemet som virtualiseres trenger ikke modifisering eller være designet for den underliggende hardwaren, fordi den ikke vil være klar over at den eksisterer.

Dersom man ønsker å oppnå full virtualisering med en arkitektur som ikke støtter ”native” virtualisering som for eksempel x86 arkitekturen, må man oversette deler av de originale instruksjonssettene for at VMM skal kunne ha kontroller over CPU.

Paravirtualisering:

Paravirtualisering er en virtualiserings teknikk der VMM presenterer logisk hardware for virtuelle maskiner som ligner den underliggende fysiske hardwaren men ikke er identisk.

Meningen med den paravirtualisering er å redusere eksekveringstiden til gjeste operativsystemene på oppgaver som er vanskeligere å kjøre i ett virtuelt miljø enn i ekte miljø.

Ved paravirtualisering har man definerte ”kroker” som tillater gjest og horst til å spørre og svare om hvor denne typen oppgaver skal eksekveres. Hadde det ikke vært for disse ”krokene” vill alle oppgaver blitt eksekvert i det virtuelle miljøet selv om det går seinere å utføre de der.

Kort fortalt det som skjer er at VMM byggeren definerer utseende til den virtuelle maskinen ved å erstatte instruksjoner som ikke lar seg virtualisere i den orginale koden med en mer effektiv tilsvarende kode som er lettere og virtualisere.

Paravirtualisering krever at gjeste operativsystemet er ”paravirtualization-aware”[30], operativsystemer som ikke er ”paravirtualization-aware” kan ikke kjøres på en paravirtuell VMM. Det er tilfeller der operativsystemer ikke kan modifiseres, i disse tilfellene er det tilgjengelige komponenter som skal gjøre det mulig å utnytte de fordeler som kommer med paravirtualisering. Ett eksempel på dette er XenWindowsGp1Pv[31] prosjektet som tilbyr en pakke ”paravirtualization-aware” drivere. Ideen her er at disse driverne skal installeres på en virtuell Windows maskin som kjører på en Xen hypervisor for gi den mulighet til å kjøre paravirtualisering.

2.3 PRODUSENTER AV VIRTUALISERINGS TEKNOLOGI

2.3.1 VMWARE

Kanskje den viktigste bidragesyteren til utvikling av virtualiserings teknologi. Selskapet Vmware ble grunnlagt i 1998 og har i dag sine hovedkvarter i Palo Alto, California.

Vmware leverte sitt første produkt Vmware Workstation i 1999, som var rettet mot enkle maskiner. De gikk ikke inn i server marked før i 2001 med VMware GSX Server[32] og VMware ESX Server.

VMware tilbys i dag en rekk produkter, VMware Workstation, WMware ESX server og WMware Server som er basert på Vmware GSX server som nå fungerer som ett gratis VMware alternativ. Denne gratis versjonen ble lansert i 2006 og er tilgjenglig på Windows og Linux plattformen.

<table>
<tr><td rowspan="2">vCenter Server ($)
(license manager)</td><td rowspan="2">Server Hardware</td><td colspan="3">ESX ($)
(vMotion, DRS, HA, Storage vMotion)</td><td>Guest OS
Guest OS
Guest OS...</td></tr>
<tr><td colspan="3">ESXi (freeware)
(ESXi freeware is managed by the Virtual Infrastructure (or vSphere) Client)
ESXi ($)
(vMotion, DRS, HA, Storage vMotion)</td><td>Guest OS
Guest OS
Guest OS...</td></tr>
<tr><td colspan="2" rowspan="3">Workstation Hardware</td><td rowspan="3">Windows or Linux OS</td><td colspan="2">VMware Server (freeware)</td><td rowspan="3">Guest OS
Guest OS
Guest OS...</td></tr>
<tr><td rowspan="2">User Session</td><td>VMware Workstation ($)
VMware Player (freeware)</td></tr>
<tr><td>vSphere Client for managing ESX(i) hosts (freeware)</td></tr>
</table>

Figur 3 – Viser forskjellige VMware produkter.

Som vi kan se ut fra figur 3 har VMware en rekke forskjellige produkter, med tanke på hvordan de er implementert, egenskaper og kostnad. Men de deler alle ett felles prinsipp om hvordan kode skal kjøres. Dette fungerer ved at segmenter med kode blir først skannet, ut fra denne skannen kan instruksjoner som ikke er mulige og virtualisere identifiseres. Disse instruksjonene blir enten oversatt direkte slik at det hopper til VMM eller at hele instruksjonen blir byttet ut som tilsvarer den originale instruksjonen slik at denne trygt kan bli utført.

Dette fører til at man har ett fult virtuelt system som tillater umodifiserte operativsystemer til å virke som alle instruksjonene faktisk blir utført på hardware.

Denne metoden for oversetting av kode heter binær oversetting. Felles for alle VMware produkter er at de støtter binær oversetting.

2.3.2 CITRIX/XEN

XenSoruce startet opprinelig som ett forsknings prosjekt ved universitetet i Cambridge. Selskapet jobber med utvikling av en rekke open source[33] prosjektet men også med større enterprice prosjekter. Det første XenSoruce produktet ble lansert i 2003.

I de tidligere versionen av Xen hypervisor ble det brukt paravirtualisering. Her var det ikke vært mulig og virtualisere linux maskiner uten å modifisere de, også kalt Xen-enabled[34]. De har ikke hatt dette "problemet" med windows maskiner grunnet avtaler de har hatt med Microsoft.

I oktober 2007 ble XenCource og Citrix Systems slått sammen. Citrix er også en aktør innen virtualisering. Gjennom deres samarbeid har de dratt nytte av hverandres teknologi og lanserte i 2009 Citrx XenServer.

2.4 HARDWARE

I del 2.2 nevnte vi forskjellige typer virtualisering gjennom bruk av forskjellige hypervisoerer og forskjellige virtualiserings teknikker. For at full virtualisering skal kunne foregå må man ha hardware med støtte for denne typen virtualisering.

En produsent av slik hardware er IBM. De har siden lansering av sin 370 hovedrammer[35] på 60 tallet hatt støtte for dette. Dette er noe som gjør IBM til en pioner med tanke på produsering av hardware med virtualiserings støtte.

Den populære x86 arkitekturen fikk ikke støtte for hardware virtualisering før i 2006 med lansering av intel og admes, intel VT[36] og AMD-V[37].

Det er på grunn av at x86 arkitekturen ikke fikk støtte for virtualisering før i 2006 at teknikkene som paravirtualisering og binær oversetting er utviklet og tatt i bruk.

2.5 FORDELER

Som tidligere nevnt er virtualisering noe som til stadig blir mer og mer popplært. Dette har mye med at dagens maskiner blir kraftigere og kraftigere. Noe som fører til at mye av ressursene til disse maskinene står urørte. Ved å ta i bruk virtualisering reduserer man behovet for mengden hardware og man får bedre utnytte av den hardware man allerede har.

Mindre nedetid:

Pålitelighet og redundans er kanskje to av de viktigste egenskapene i moderne virtuell teknologi. Bedrifts kritiske applikasjoner og tjenester kan ikke akseptere nede tid. Dersom applikasjoner eller tjenester går ned på grunn av vedlikehold, hardware/software feil eller ondsinnede angrep kan dette føre til store økonomiske tap for bedriften. Med en god virtuell infrastruktur er det mulig å migrere produksjons servere direkte mellom forskjellige maskiner i nettverket, dette kan

man gjøre for å utføre vedlikehold på maskinene eller på grunn av hardware feil. Dette er noe som vil kunne føre til langt mindre nede tid fordi det virtuelle systemet er bedre rustet til å håndtere feil som ellers ville vært katastrofale for ett vanlig system. For å benytte seg av denne egenskapen må man ha en virtualisering server med støtte for dette. VMware og Citrix xen tilbyr funkjsonene V-motion[38] og live motion[39] som gjør akkurat dette.

Redusert kostnad:

Bedrifter forsøker hele tiden å redusere kostnader og arbeide så effektivt som mulig for å nå høyst mulig overskudd. Dette gjelder også for bedrifters IT avdelinger. Det mange har gjort tidligere er å dedikere en server til hver tjeneste eller applikasjon de kjører, dette er en dyr og inneffektiv måte å drive IT på samt at det krever en god del arbeid å holde orden på alle serverne.

En metode for å gjøre dette enklere og billigere er å ta i bruk en virtuell infrastruktur.

Dette gir bedriften muligheten til å separere tjeneste og applikasjoner fra hardware og benytte all serverhardware ressurser på en ny måte. Dette vil tillate bruk av gammelt og nytt utstyr til dets fulle potensiell slik at ingen hardware ressurser i bedriften ikke blir utnyttet.

Vedlikehold:

Vedlikehold av servere er den største faktoren til nedetid til steder som stiller høye krav til tilgjenglighet. Studier viser at vedlikehold som er planlagt står for 75-90 % av all server nedetid[40]. Selv om dette er et kjent faktum finnes det en praktisk tilnærming for å løse dette problemet. En måte å redusere nedetiden på ett system er å benytte redundant hardware for å beskytte seg mot hardware feil. Eksempelvis to identiske disker, går en i stykker kan man fortsatt bruke den andre til man får byttet ut den som er ødelagt. Men dette hjelper ikke om det er komponenter som, hovedkort, disk kontroller eller en annen del i hardware som ikke lar seg kopiere i et redundant serveroppsett.

Ved bruk av virtuelle maskiner løser man dette problemet enkelt ved å flytte den virtuelle maskinen fra den fysiske maskinen som er skadet og/eller trenger vedlikehold til annen hardware server samtidig som den virtuelle maskinen kjører og leverer sin tjeneste. Når det er så enkelt å flytte virtuelle maskiner mellom forskjellig hardware gjør dette det også veldig enkelt å redusere nede tiden ved planlagt vedlikehold. Ved at man flytter over den virtuelle maskinen midlertidig til en annen server hardware, mens man utfører vedlikehold, vil den virtuelle maskinen kunne levere sin tjeneste. Denne måten flytting av virtuelle maskiner mellom forskjellig hardware kan også benyttes om skal bytte ut hardware komponenter eller lignende.

2.6 TAP AV YTELSE:

Det er ingen hemmelighet at virtualisering fører til ytelses tap av en viss grad. Tapet i ytelse kan variere fra noen få prosenter og oppover[41]. Men de fleste av dagens servere er så kraftige at man ikke alltid får utnyttet all ressursene som finnes i maskinen. Om dette er tilfellet er virtualisering et godt alternativ.

Når man er klar over at virtualisering er aktuelt i sin bedrift er det viktig at man utnytter virtualisering teknologien så effektivt som mulig. Det er da viktig å velge riktig VMM.

Tidligere tester av hypervisor 2, spesielt VMware workstation og Xen Hypervizor viser at deres ytelse ligger svært nær native ytelse. Dette kommer klar frem i arikkelen ”Xen and the art og virtulazation”[54].

Det finnes ikke tilsvarene arbeid gjort for testing av Hypervisor 1. Det finnes heller ingen standardisert metode for å teste denne typen hypervisor.

3. TESTING

3.1.1 INTRO TIL TESTING

Utvikling har gjort dagens datamaskin arkitektur veldig avansert. Dette har ført til at det er blitt vanskeligere å sammenligne ytelsen til forskjellige maskiner kun ved å se på maskinenes spesifikasjoner. Ett eksempel på dette er en Pentium 4[42] prosessor arbeider vanligvis på en høyere klokke frekvens enn en Athlon XP prosessor[43], selv om klokkefrekvensen er høyere på en Pentium 4 betyr ikke dette nødvendigvis at den er kraftigere enn en Athlon XP.

Målet med oppgaven er å utvikle en test metode for testing av virtualiseringsservere. En test metode kan defineres som en bestemt prosedyre som produserer et testresultat.

Det kan videre sies at et testresultat kan inneholde enkle ja/nei, kategorier eller målte verdier.

Ved testing er det viktig at alle involverte parter er enige om hvilken metode som skal benyttes og hvilke data testen skal generere.

I artikkelen ”Real-world virtualization benchmarking”[44] av AnAndTECH nevnes det fire viktige punkter ved utvikling av en benchmarking metode. Disse punktene gjelder også for utvikling av test metoder generelt.

- Repeterbar
- Relevant
- Sammenlignbar
- Tyngde

Repeterbar:

Testene som utføres må være repeterbare. Det som menes med dette er at de samme testene kan bli utført flere ganger. Og at testene som utføres gir samme resultat flere ganger dersom forutsetningen er de samme. Ved testing av virtuelle maskiner kan by på flere problemer enn native maskiner. Dette er spesielt med tanke på CPU last. Virtuelle maskiner rapporterer sjeldent 100 % CPU last[45]. Lasten som rapporters fra den virtuelle maskinen kan variere ved forskjellige typer cpu` er selv om den virtuelle maskinen er den samme. Dette kan føre til at det er vanskelig å regulere CPU lasten på virtuelle maskiner når de skal testes, siden lasten som generer er høy på noen maskiner men relativt lav på andre.

Relevant:

At testene som utføres, og de testresultater som genrers er relevante, er muligens det viktigste punktet. Dersom man utfører en test for å måle ytelse av ett produkt er det viktig å være klar over hvilken type testresultater man er ute etter. For eksempel om man utfører en test av CPU vil ytelse data om CPU være de relevante data man er ute etter og ikke disklesing.

For at dataen skal kunne kalles relevant er det viktig at man er klar over mulige flaskehalser.

En flaskehals er ett punkt eller en del i testen som kan redusere den gjennomsnittlige ytelsen til ett så lavt nivå at testresultatene blir påvirket i negativ retning. Dette kan føre til at man får tilnærmet de samme resultatene for to forskjellige produkter fordi de deler samme flaskehals. Det vil igjen føre til at resultatene blir irrelevante.

Derfor er det viktig at man er bevisst på mulige flaskehalser og kartlegger disse slik at man produserer og oppnår relevante testresultater.

Sammenlignbar:

Dette punktet er tett tilknyttet punktet om relevans. For at dette punktet skal oppfylles må det tilstrebes at den samme testen kan utføres på forskjellige produkter med samme forutsetning og oppsett. Dette er ikke noe problem om man utvikler en testmetode for å måle ytelsen til ett produkt, men på den andre siden bør man ha noe å sette resultatene opp imot. Derfor er det viktig at verdier som produseres er av samme type og menge slik at de kan sammenlignes med resultater fra andre produkter.

Verdien bør presenters på en slik måte at forskjeller mellom produkter kommer klart frem. Vi tenker da spesielt på tall verdier. Feil presentasjon av tall, da spesielt avrunding kan gjøre sammenligning vanskelig fordi de små forskjellene ikke kommer frem. Signifikansnivå eller feilmargin må også medregnes under sammenligning av test resultater.

Tyngde:

Testens tyngde eller belastning må være av en slik størrelse at forskjeller mellom produkter kommer klart frem. Dersom testens tyngde er for lav kan dette føre til at test resultatene ligger på et maksimalt nivå av det testen har satt til, som kan medføre at forskjeller mellom produkter ikke kommer frem.

Tyngde og relevans er to punkter som er avhengige av hverandre. Det er ikke nok med en test, med en tung belastning dersom belastning ikke er relevant for den typen test.

Regulering av tyngde kan også være en viktig del av en test for å finne eventuelle brekkpunkter eller flaskehalser.

3.1.2 TEST METODER:

I datamaskinens verden finnes det to tilnærminger for utvikling og gjennomføring av tester, Blackbox[46] og Whitebox[47] testing.

Blackbox:

I denne tilnærmingen er man klar over hva som sendes inn av input til testen og hva slags test resultat som blir produsert. Hva man ikke er klar over er hva som skjer inne i systemet og hvordan testresultat er blitt produsert. Ved denne typen tilnærming er det svært vanskelig å oppfylle alle de 4 punktene om relevans, repeterbarhet, sammenlignbarhet og tyngde. Selv om dette er tilfellet betyr ikke

dette at denne metoden ikke kan produsere valide test resultater. Men det vanskelig å avgjøre om de er valide på grunn av uvissheten som medfølger bruken av blackbox testing.

Whitebox:

Ved utvikling av denne typen test har man ett klart innsyn i hva som skjer inne i systemet og man bruker dette innsynet i den interne strukturen til å utvikle test metoden. I motsetning til blackbox testing er det enklere å oppfylle de 4 punktene om relevans, repeterbarhet, sammenlignbarhet og tyngde fordi man har innsikt i systemet og man vet hva som foregår.

3.1.3 BECNHMARKING:

Benchmarking er en metode der man kjører ett program, ett sett programmer eller en annen operasjon for å vurdere den relative ytelsen til produktet eller objektet. Benchmarking er ofte assosiert med å måle og vurdere ytelsen til datamaskin hardware som floting point[48] operasjon ytelsen til en CPU, men det er også mulig å måle ytelsen til software.

Benchmarking av hardware kan være svært viktig for utvikling av prosessorer[10], siden dette kan gi en pekepinne på hva som må gjøres for å bedre ytelsen til prosessoren.

Det finnes mange forskjellige former for benchmarking, men det velges å fokuseres på metodene som er mest relevante for oppgaven. Som er syntetisk og real life benchmarking.

3.2.1 SYNTETISK BENCHMARKING

Dette er den mest tradisjonelle formen for benchmarking, 90 % [49] av alle testrapporter som involverer benchmarking er syntetisk til en viss grad.

Det som kjennetegner syntetisk benchmarking er at det ikke blir utført noen funksjoner utenom å produsere tall og data som kan bli brukt til sammenligning. Syntetisk benchmarking sier veldig lite alene, det kan være ett sett tester som kjøres og produserer en poengsum. Denne summen alene sier deg ikke mye med mindre man har noe å sammenligne den med.

Syntetisk benchmarking blir veldig ofte benyttet til måling av hardware ytelse. Og da spesielt benyttet av hardware produsenter for å avdekke områder produktet har eventuelle ytelses problemer.

```
”for (int i = 0; i < 10000000;  i++) {
Getppid();
}”
```

Figur 4 – Kode eksempel på syntestisk benchmark

Figur 4 er et veldig enkelt eksempel på programkode som kan brukes til syntetisk benchmarking. Dette forgår ved at man tar tiden det tar å utføre systemkallet

getppid x antall ganger. Denne testen er lett å replisere på andre maskiner som kan brukes til sammenligning.

Whetstone[50] og Dhrystone[51] var de første standardiserte syntetiske benchmarkene. Whetstone ble utviklet som en generell benchmark for datamaskiner, Dhrystone var en spesifikk benchmark rettet mot integer programmering i CPU. Dhrystone vokste til å bli en representativ benchmark for generell cpu ytelse frem til den ble erstattet av CPU89 testen som i dag heter SPECint til Standrad Performance Evaluation Corporation[52].

3.2.2 TIDLIGERE ARBEID

Gruppen føler det er viktig å så se på hva som er gjort av tidligere arbeid når det gjelder syntetisk benchamrking av virtuelle maskiner. Rapporten ”A comparison of software and Hardware Techniques for x86 virtualization” [53] er ett godt eksempel på dette. Rapporten er relevant i den forstand at de har testet virtuelle maskiner. Men de har ikke satt forskjellige virtualiseringsprodukter opp mot hverandre. Det er en vitenskaplig rapport som peker på viktige forskjeller i ytelse mellom native, hardware virtualisering og software virtualisering. Testene som utføres varierer i tyngde og omfang men felles for de alle er de er syntetiske i den forstand at de kun er produsert for å måle ytelse på spesifikke områder. Ting blir ikke belastet med belasting som relevant for vanlig bruk av maskiner.

Resultatet av denne rapporten viser at det er en del forskjeller i ytelsen på hardware og software virtualisering, men at de begge ligger nær native ytelse.

Rapporten “Xen and the art of virtualization[54]” er en annen god rapport. Testene som blir utført her er også syntetisk i form av at det benyttes syntetiske spec tester og andre målrette benchmarks for å peke på forskjeller mellom Xen, VMware og native ytelse. De har her satt Xen og VMware opp mot hverandre men testene som utføres er kunstige og ikke relevante som ”normal” belastning.

3.3.1 REAL LIFE BENCHMARKING

Det som kjennetegner denne typen benchmarking er at det benyttes ”vanlige” programmer til å utføre testing. Med vanlige programmer menes programmer man bruker til daglig[49]. Det er viktig å huske her at disse programmene ikke produserer en poengsum eller annen tallverdi ut fra testen, som man lett kan sammenligne med andre resultater.

Forskjellen mellom syntetisk og real life benchmarking ligger i programmet som brukes for utføring av testing. Syntetisk benchmarking er spesielt designet til å utføre testen å produsere et testresultat. Men i real life brukes det programmer som opprinnelig har andre formål enn benchmarking. Real life benchmarking er en mer ”naturlig” måte å utføre en test på siden belastingen er som regel mer relevant enn den ville vært i en syntetisk benchmark. Problemet med real life benchmarking er at denne formen for tester er svært følsom for små endringer i variabler sammenlignet med en syntetisk benchmark. Dette er noe som kan gi store forskjeller i testresultater. Grunnen til dette er at en real life benchmark ofte berører og benytter seg av flere eller alle komponenter i maskinen. Noe som fører til at man har flere

baller i luften når man utfører en test som kan gjøre testen mer følsom for variable endringer.

Måten en real life benchmark virker på er at man tar ett program eller en funksjon som brukes mye til vanlig og finner en måte å måle ytelsen. Dette kan for eksempel gjøres ved måling av tiden det tar å utføre en gitt jobb, eksempelvis enckoding av en film. Men real life benchmarking kan også være ett program som er lagt spesielt til testing mot noe vanlig, som for eksempel benchmarking av en webserver.

3.3.2 SPEC

Ett eksempel på syntetisk benchmarking og real life er Standard Performance Evaluation Corporation (SPEC) tester. SPEC er en organisasjon som forsøker å standardisere performance benchmarking for datamaskiner. SPEC produserer tester for syntetisk og real life benchmarking. Eksempler på en syntetisk test kan være målrettede CPU tester, og på real life benchamrking finnes det tester rettet mot web servere.

Per dags dato tilbyr ikke Spec noen spesifikk testmetode for testing av virtuelle maskiner. Men de har nedsatt en komité som har startet arbeidet med utvikling av første generasjons spec tester for sammenligning av virtuelle maskiner[55].

Selv om dette er tilfellet, betyr ikke dette at man ikke kan bruke spec tester til testing av virtuelle maskiner. I rapporten ”Xen and the art of virtualization[54]” som en forsknings basert artikkel lansert av Xen, benyttes det syntetiske spec tester for testing av virtuelle maskiner. De samme testene benyttes også i VMwares atrikell “A comparison of software and Hardware Techniques for x86 virtualization”[56] for testing og benchmarking av forskjeller mellom hardware og software virtualisering.

Det benyttes også spectester i VMmark til testing av webserver og java.

Gruppen har vurdert å benytte spec tester til deler av oppgaven siden dette er kjent metode som produsere relevante og gode test data.

Men gruppen valgte å falle bort fra dette grunnet lisens kostnader for å få spec tester utført.

3.3.3 VMMARK

VMmark[57] er en benchmark pakke utviklet av VMware. Pakken måler ytelsen til flere virtuelle maskiner som kjører på samme fysiske hardware. Dette kreves av VMmark for at testen skal produsere relevante test data. Hver av de virtuelle maskinene som kjører på hypervisoren må konfigureres etter maler som blir gitt av VMmark programvaren. Disse malene inneholder informasjon om software som må installeres og settes opp riktig på de virtuelle maskinene. Denne softwaren er av den typen man ofte kan finne i et datasenter i en bedrift, slik som database servere, web servere og e-post servere.

Det er kun data som er relevante for den aktuelle applikasjonen som blir hentet ut under testene. Som hvor mange requests en webserver kan takle, eller transaksjoner i sekundet mot en database.

Hver virtuelle maskin får poeng basert på ytelsen. Til slutt vil hele settet av virtuelle maskiner som kjører på hardwaren få en total sum.

Spesielt for VMmark er det utføres mange små lette tester på mange virtuelle maskiner.

VMmark er en pakke bestående av mange forskjellige real life benchmarks. Det benyttes alt fra sysBench[58] for database testing til SPEC tester for webserver testing. VMmark er en god og omfattende benchmark, men det kreves at man går til innkjøp av spec, windowsserver og exchange lisenser.

3.3.4 VAPUS MARK

vApus Mark[44] er en real life benchmark utviklet av AnandTech. vApus har mange likhetstrekk med VMmark med tanke på typen tester som utføres. Til forskjell fra VMmark forsøker vApus å utføre tester mer rettet mot "real life" belastning. VMmark forsøker å fordele belastning ut over mange virtuelle maskiner. Dette fører til at belastning blir relativt lav per maskin noe som kan gjøre at testene ikke oppfyller kravet om tyngde. På den andre siden har vi vApus som satser på færre virtuelle maskiner men med en tyngre belastning.

vApus programmet er gratis. Men på samme måte som VMmark benytter vApus en rekke forskjellige applikasjoner til testing som krever at man må kjøpe lisenser.

4 UTSTYR

4.1 HARDWARE

Utstyret som benyttes i denne oppgaven er lånt av høgskolen i Gjøvik, med unntak av egne datamaskiner som er benyttet til å utvikle skript og som kontroll maskin for testing og logging. Følgende utstyr er lånt av skolen:

HP ProLiant 320 G5[59] server med støtte for hardware virtualisering.

Dell GX260[60] PC.

HP ProLiant 320 G5:

Intel Dual-Core Xeon 3060 2.4Gzh L2 4mb Cache FSB 1066Mzh

2 GB SDRAM

2x250 GB Harddisk

CD/DVD rom

Virtualiserings støtte

For virtualiserings støtte har prosessoren kun VT-x[61] som må aktiveres i bios. VT-x avbelaster VMM for lytting, trap og kjøring av enkelte intruksjoner for gjesteopprativsystemet slik at de kan gå utenom VMM på en sikker måte.

Dell GX260:

Dell Optiplex GX260

Intel Pentium 4

512MG RAM

40GB Harddisk

CDROM/DVDROM stasjon

4.2 SOFTWARE

Målet med oppgaven er å teste hypervisorer mot hverandre. Belastningen på testene som kjøres mot disse skal være relevante for it-tjenesten ved høgskolen i Gjøvik. Med dette menes det belastning som hadde vært reel på hypervisoren om it-tjenesten hadde brukt den til daglig drift altså server/klient applikasjoner. For å gjøre dette måtte gruppen benytte følgende operativsystemer og applikasjoner i oppgaven.

Operativsystemer/hypervisorer:

- Windows server 2008[62]
- Cent OS
- VMware esxi 4.0
- Citrix xen server 5.5

Programvare:

- Apache bench[63]
- Epost på linux
- Active directory[64]
- Open LDAP[65]
- FTP [66]
- Mysql database[67]
- Terminal

4.2.1 VMWARE ESXI 4.0

Vmware Esxi er server virtualiserings programvare fra VMware. VMware esxi og esx er begge bare-metall hypervisorer som kan installeres direkte på fysisk hardware. Forskjellen på esx og esxi ligger i at esx også installeres med en linux basert kontroll konsoll i stede for med en fjern kontroll tjeneste slik som esxi gjør. Esxi er gratis og koster ingen ting å laste ned og bruke. Med denne gratis versjonen har man tilgang til kjerne hypervisor funksjonalitet, virtuell SMP[68] og VFMS[69]. Men ønsker man å dra full nytte av dette produktet og funksjonalitet som VMotion og DRS[70] kreves tilleggsprogramvare som Virtual klient og vSphere 4.

Med VMwaare esxi har man muligheten til å installere en rekke virtuelle maskiner på samme hypervisor. Esxi tilbyr en rekke verktøy og egenskaper[71] for kontroll, vedlikehold, fjernstyring, konfigurering, brukertilgang og skalering av disse maskinene.

E. OPPSETT AV TEST MASKINER OG TEST MILJØ

Installasjon av esxi er enkelt og rett frem[Se vedlegg E, Oppsett av test maskiner og test miljæ, VMware Esxi 4.0 s.88], etter installasjon av esxi har man få opsjoner. Det finnes ingen måte for å sette opp eller kontrollere virtuelle maskiner direkte på serveren. Man har kun få enkle opsjoner for å sette brukernavn, passord og lignende på serven.

For installasjon og kontroll av virtuelle maskiner må ha bruke vSphere klienten som må lastes ned og installeres på en annen fjern maskin.

Vsphere klienten gir deg ett enkelt innloggings vindu der det bes om brukernavn, passord og ip til esxi serveren du ønsker å koble til.

Etter at innloggingen er vellykket blir man presentert med GUI. Man har muligheten til å sette admin rettigheter på virtuelle maskiner, se logg eller åpne innholdet på serveren. Det er fra serverinnholdet man oppretter, konfigurerer og håndterer virtuelle maskiner.

Oppretting av virtuelle maskiner i vSpher klient opp mot esxi er enkelt. For de som er kjent med VMware worstation bygger det på det samme prinsippet. For installasjon av helt nye maskin kan man enten benytte seg av iso bilder av

operativsystemer og importere disse fra fjerne maskiner inn til serveren eller benytte seg av cd/dvd rommen på serveren og installere på den ”tradisjonelle” måten.

For hver virtuelle maskin som opprettes på serveren blir den lagt i innholdslisten. Denne listen gir god oversikt over hvilke virtuelle maskiner som er befinner seg på serveren og hvilke av de virtuelle maskinene som kjører. Samt kan man velge enkelte maskiner og åpne en konsoll mot den virtuelle maskinen.

Konsoll er gjeste os’et sitt skjermbilde som kan brukes til videre konfigurering og installasjon av maskinen.

Man har også mulighet til å tildele virtuelle maskiner mer logiske ressurser, dette gjøres ved at man pauser en virtuell maskin og endrer på det logiske hardware oppsettet til maskinen. Det er også mulighet for å overvåke ressursene til den virtuelle maskinen.

Dersom man ønsker å migrere andre virtuelle maskiner som er opprettet av andre hypervisorer kan man gjøre dette ved bruk av VMware vCenter Converter. Den kan importere andre virtuelle maskiner som ble opprettet på VMware Server, Microsoft Virtual Server eller Microsoft Virtual PC version 7 og høyere.

4.2.2 CIRTIX XEN SERVER 5.5

I likhet med VMware esxi er Citric Xen Server 5.5 en ”bare-metall” hypervisor som kan installeres direkte på hardware. Xen server 5.5 ble lansert i juni 2009 og var men som en erstatter for xen server 5.0 og 4.1

I likhet med esxi installeres ikke xen server 5.5 med noen konsoll for kontroll, men avhenger på samme måte av en fjernkontroll tjeneste. Xen server 5.5 er gratis å laste ned å bruke. Funksjonalitet som følger med er mer omfattende enn VMware esxi, med funksjoner som live motion. Live motion er en lett utgave av VMware`s VMotion[8], som stiller krav til hardware som blir brukt til flytting av virtuelle maskiner.

Dersom man ønsker tilgang til alle egenskaper i Xenserver 5.5 må man kjøpe lisens til Citrix essentials på samme måte som man må med vmware esxi

Installasjon[Se vedlegg E. Oppsett av test maskiner og test miljø, Xen Server 5.5 s.90] av Xen server 5.5 er rett frem. Etter installasjonen er ferdig har man ingen direkte kontroll eller mulighet til å sette opp eller kontrollere virtuelle maskiner.

For at det skal være mulig må det installeres Xen Center 5.5 på en PC. Xen Center 5.5 ligner og fungerer på samme måte som VMware sin vSphere klient.

Det skal sies at det er en del forskjeller på begge produktene men man finner mange likhetstrekk mellom dem. I form av: oppretting av nye virtuelle maskiner, snapshot managment[72], oversikt over de virtuelle maskinene og overvåking av virtuelle maskinressurser.

Dersom man ser bort i fra ”utvidelsene” som Citrix essentials og vsphere 4 og bare ser på gratis utgavene, er det Xenserver 5.5 som tilbyr flest ekstra egenskaper. Disse

egenskapene er ikke av betydning for arbeidet videre i oppgaven siden det er den generelle ytelsen til hypervisoren vi ønsker å teste.

I versjon 4 av ESXi prøver vmware å skjule linux terminalen[73] som ellers kan være nyttig ved automatisering med script og lignende. På Xen server derimot er linux terminalen lagt tilgjenglig.

5 UTFØRELSE

Gruppen ønsker å utvikle en metode for stresstesting av virtualiseringsservere. Det finnes mange metoder for å utføre en stresstest eller benchmark på. Gruppen ønsker å utvikle en reallife benchmark av virtualiseringsserverne. Rettet mot oppdragsgivers ønsker.

I samtaler med oppdragsgiver er gruppen blitt informert om hva it-tjenesten bruker virtualisering til i deres serverløsning. Det ønskes å ta utgangspunkt i denne informasjonen for hvilke type tester som skal utføres på virtualiseringsserveren. Og på denne måten sikre at tester og belastning er relevante for oppdragsgiver ved å utvikle tester som belaster serverapplikasjoner ut fra hvilke roller deres virtuelle servere har i dag, og andre servertjenester som eventuelt kan overtas i framtiden.

Det er kommet frem til at IT tjenesten ved høgskolen i Gjøvik bruker eller kunne tenkte seg å bruke en virtualiseringsserver til følgende serverløsninger:

- Epost
- AD/LDAP kontroller
- Filserver (FTP)
- Web server (Apache)
- Terminal server
- MySQL Database server

Disse overstående punktene er utgangspunktet for hva slags tester som utvikles og utføres. For at testene skal være så reelle og relevante som mulig er det viktig at de blir utført på rett operativsystem i henhold til det oppdragsgiver bruker på sine virtuelle maskiner. Som er Windows Server 2008 og Cent OS.

Det neste gruppen måtte ta stilling til var hva som skal måles. Det var allerede klart hvor belastingen kom fra, men det er viktig å vite hva vi skal se på med tanke hvilke typer testresultater vi ønsker å sammenligne med. Slik gruppen så det stod det mellom to valg. Den ene muligheten var å teste virtuelle maskiner med gitte roller og se på ytelsen i den logiske hardwaren som maskinen tror den har. Ved å se på hvor mye en gitt maskin bruker ved en bestemt mengde belastning. Den andre muligheten var å se på resultater generert fra spesifikke tester. Eksempelvis å se på transaksjoner i sekundet ved en database test. Siden det skulle utvikles en metode for real life testing av serverapplikasjoner var det den andre muligheten som falt mest naturlig.

Ett krav vi selv stilte var at testene som utføres var gratis og ikke kostet noe å bruke. Det ble derfor valgt å generere belastning gjennom selvproduserte skript. Her ble det valgt å bruke perl som scriptspråk siden det finnes flere moduler i perl som kan kommunisere med flere av serverapplikasjonene. Dette gjorde det mulig å lage små klienter i form av script kun for å belaste serverne.

Ved å vektlegge resultatsverdier som er spesifikt for en enkel test av serverapplikasjon, er den beste verdien å se på med tanke på relevans. Dette vil gi oppdragsgiver en pekepinne på hvilke av hypervisorene som fungerer best til hver enkel serverapplikasjon.

5.1 TEST SENARIO

Når det er snakk om en virtualiseringsserver er det flere ting enn ytelsen til en enkelt maskin vi ønsket å se. Ved bruk av en virtuell server har man muligheten til å kjøre flere virtuelle maskiner samtidig. Dette er noe det må tas hensyn til når det skal testes. I ett forsøk på å avdekke så flest mulig forskjeller på eventuelle virtualiseringsservere som testes, har vi utvikle 4 forskjellige typer testsenarioer. Alle senarioer tar utgangspunkt i at det er VMware Esxi og Citrix XenServer som settes opp mot hverandre. Etter eget ønske det mulig å utføre disse på andre typer virtualiseringsservere.

Senario: Bare-Metall v.s Virtualisering

Bare metall

Installasjon av Cent OS og Windows 2008 Server rett på hardware uten noen virtualiseringsplattform i bunn. Operativsystemene blir satt opp med alle serverapplikasjonene som også skal testes under virtualisering. Ved å kjøre testene rett på metallet med helt likt arbeid får vi et resultat hvor serverapplikasjonene kjører under forhold hvor virtualisering ikke brukes. Ved å se på dette resultatet kan vi sette dette opp mot et enderesultat utført under virtualisering for å se eventuelle ytelses tap eller forbedring.

Virtualisering alene

Dette utføres på Vmware Esxi og XenServer. Disse installeres hver for seg direkte på hardware. Videre blir det opprettet en virtuell maskin som skal testes på samme måte som ved bare metall testen. Ideen bak dette er å se tapet eller økt ytelsen ved virtualisering. Eksempler på hvordan Vare metall og Virtualisering alene utføres Eksempler på dette senarioet finnes i vedlegg F. Test Senario.

Senario 1 - Støy og test intervaller

Det som ønskes å se her er hvordan hypervisorene oppfører seg dersom det er flere gjesteoperativsystem som trenger hardware ressurser. I denne testen fokuseres det på hvor god ytelse et gjesteoperativsystem levere rett etter det har vært uten noen applikasjonsbelastning i en angitt tid. Samtidig som gjesteoprativsystemet er stille vil et annet gjesteoprativsystem belaste hypervisoren. Denne testen utføres på Esxi og XenServer. Her ønsker vi å se om det er noen forskjell på hvordan hypervisorene yter i denne settingen. Eksempler på dette senarioet finnes i vedlegg F. Test Senario.

Senario 2 Støy og test samtidig kjøring

Samme oppsett som overstående eksempel. Forskjellen her ligger i at støymaskinen kjører samtidig som målmaskinen utfører tester. Det vil bli utført tester og støy i

flere sykluser med en nedkjølingsperiode imellom hver av dem. Mengden støy vil øke ved hver syklus. Eksempler på dette senarioet finnes i vedlegg F. Test Senario

Senario 3: Blanding av intervall og samtidig kjøring

Denne metoden er en hybrid av de to overstående punktene. Her startes det med å belaste hypervisoren med en fast belastning i loop. Etter at belastningen har fått kjørt alene i en angitt tidslengde vil selve applikasjonstesten starte og kjøre samtidig som belastningen. Eksempler på dette senarioet finnes i vedlegg F. Test Senario.

5.2 PROGRAMMER OG SCRIPT

For å utføre de forskjellige testene på målmaskinene skal gruppen benytte skript, her vil hver test ha ett eget skript. Skriptene vil utføre en bestemt jobb på maskinen som skal måle og lagre unna resultat for testen. Dette skal gjøres ved at skriptene enten kjører ett sett med kommandoer og spørring rettet mot den tjenesten som skal testes, eller at skriptet setter i gang en ferdig laget benchmark rettet mot applikasjonen. Ett eksempel på dette er Apache benchmark, dette er en benchmark for webservere. I tilegg til scriptene for belasting av server applikasjonene er det blitt utviklet et skript for generering av cpulast kalt støyscriptet.

5.2.1 STØY

Er et perl script ” cpuLastKontroll.pl”[s.190] som er utviklet for å kunne ha en repeterbar og statisk belastning på et gitt operativsystem av typen Linux eller Windows med perl kompilator. Scriptet er helt nødvendig for å kunne kjøre testsenarioer hvor hypervisoren skal belastes av to gjesteoprativsystemer. I et slikt testsenario vil et gjesteoprativsystem kun brukes til støyscriptet, mens det andre gjesteoprativsystemet blir brukt til en serverapplikasjon som skal testes.

Som nevnt er sckriptet statisk, fordi det kjører med en fast belastning i loop. Hensikten med dette er å gjøre støyscriptet repeterbar. Ved å sette belastningsnivåer kan den etterligne en serverapplikasjon som er lett eller kraftig, belastet etter hvor mange prosent av prosessorens ytelse man ønsker å belaste. Man ønsker her å se hvor mye de ulike belastningsnivåene av prosessoren påvirker en serverapplikasjon kjørt i et annet gjesteoperativsystem.

Når støyscriptet skal brukes i et testsenario[s.104] må det først kalkuleres konfigurasjons verdier for de ulike cpu belastningsprosentene man ønsker å utsette hypervisoren for. Kalkuleringen kjøres på et gjesteoperativsystem alene i hypervisoren før man starter kjøring av noen testsenarioer. Hensikten med kalkulering er å finne belastningsverdier unikt for en ny maskin som skal testes. Grunnen til dette er at bestemte belastingstyngder i støyscriptet vil variere ved kjøring på ulike hardware med forskjellig ytelse.

Når støyscriptet kjører under et testsenario vil den bli styrt igjennom rene ascii kommandoer over en forhånds bestemt nettverksport. Her kan scriptet styres ved å

angi hvor mye scriptet skal belaste serveren med forhånds bestemte kalkuleringer. Samme port brukes til å finne konfigurasjons verdier for ulike tyngder man ønsker å belaste prosessoren med. Får å få sendt kommandoer til støyscriptet manuelt kan dette gjøres ved bruk av telenet eller nc. Ved kjøring av testsenarioer sendes nettverkskommandoer automatisk til kjørende støyscript med modulen stoyKlient.pm

Modulen "stoyKlient.pm"[s.195] er inkludert og brukt i alle testsenarioene. Her sørges det for at konfigurasjoner leses inn fra filen stoyKlient.conf. I denne filen står konfigurasjoner som addresse til støymaskinen og de ulike støymengdene som skal kjøres igjennom et testsenario. Hensikten er å kunne gjøre endringer på støytyngdene til en test uten å måtte skrive belastningene direkte i perl koden til et testsenario.

Styrings Script Senario1 og Senario2

Felles for alle belastningsscriptene er at de har to styringsscript med navninnhold senario 1 og senario 2. Disse er styringsscript som sørger for at testgjennomføring skjer i en bestemt rekkefølge og variasjon. Variasjonene er mengden last støyscriptet kjører med for hver test og kjøre lengde. Avhengig av hvilke senario som kjøres, varierer også når støy kjørers i forhold til applikasjonstesten.

Senario 1 er "Virtualisering med støy i intervaller" som nevnt i testmetoder. Det brukes her to gjesteoperativsystemer på hypervisoren. Det første opprativsystemet brukes til serverapplikasjonen som skal testes mens det andre oprativsystemet blir brukt til støyscriptet. Støyscriptet har ingen direkte tilknyttet det som skal testes på den andre virtuelle maskinen. Ved kjøring av senario 1 vil belastningstesten først kjøre mot serverapplikasjonen. Når denne testen er ferdig vil operativsystemet stå idle og den virtuelle maskinen med støyscriptet vil starte og generere last i et bestemt tidslengde. Når støyscriptet er ferdig vil den virtuelle maskinen med serverapplikasjonen igjen bli belastet for en i gjentatt test.

Belastningen på serverapplikasjonen og fjernstyringen av den virtuelle maskinen med støyscriptet kjøres fra en ekstern maskin. Her lagres også alle testresultatene seg. Det er på denne maskinen testsenarioene startes og blir styrt i riktig rekkefølge.

Senario 2 er "Applikasjonstest kjørt samtidig med støy" testsenario. Det er også her brukt to gjesteoprativsystemer med samme oppsett som i senario 1. En for serverapplikasjonen og et for støyscript. Hele testen styres og logges fra en ekstern maskin. I denne testen vil serverapplikasjonen bli testet to ganger før noen støybelastning blir kjørt. Ved neste kjøring vil begge de virtuelle maskinene belaste hypervisoren. Dette skjer ved at belastingsscripet for serverapplikasjonen igjen tas og støyscriptet vil belaste etter en forhåndsbestemt belastningsmengde for testen, eksempel 50 % cpu belastning. Etter som testen for applikasjonen blir ferdig vil testen gjentas dersom flere belastningsverdier for støyscriptet er skrevet inn i "stoyKlient.conf". Etter at hele testsenarioet er kjørt vil man kunne se logresultater med støymengde merket for vært resultat gitt av en testkjøring mot serverapplikasjonen.

5.2.2 WEB TEST:

I testen brukes apache benchmark for belastning av en webside. Men for å kunne automatisere prosessen er det laget script ”abTester.pl”[s.149] for å styre når apache benchmark skal kjøre. Slik kan testen settes inn i testsenarioer og kombineres med støyscriptet.

Et testoppsett består av en ekstern klient med apache benchmark og belastningsscript med testsenarioer for automatisering av testen. På hypervisoren har man to gjesteoperativsystemer. En som fungerer som en webserver og den andre som støymaskin. Ingen kontrollscript er nødvendig å ha på webserveren for å gjennomføre testen. Kun støyscriptet kontrolleres av den eksterne maskinen.

Ved kjøring av en test settes adressen til serverer, antall simulerte klienter og antall samtidige forespørsler per klient. Resultatene blir produsert av apache benchmark og kontrollscripet sørger for å logge hver testsenario til fil.

5.2.3 OPENLDAP/AD TEST:

Et script ” ldapTest.pl”[s.174]som er laget for å belaste serverapplikasjoner som kan håndtere ldap, for eksempel OpenLDAP og Microsoft Active diractory. Scriptet er skrevet i perl og bruker perl modul Net::LDAP for å kunne kommunisere med ldapsevere for autentisering, lesing og skriving av data til databasen under kjøring[76].

Et testoppsett [s.94] ldapscriptet kjøres det eksternt fra hypervisoren på en egen klientmaskin. I tilegg til testscriptet for ldap serveren er det testscript for å kjøre ldap tester kombinert med støyscriptet i to ulike testsenarioer som beskrevet tidligere i ”Styrings Script Senario1 og Senario2”.

Selve scriptet for ldap testen er en maks ytelsestest. Her er hvert datahåndteringsantall bestemt på forhånd. Målingen i resultatet er hvor lag tid hver datahåndtering tar.

Følgende oprasjoner blir utført mot ldap serveren:

- Opplasting av gitt antall orginasjons klasser (OU)
- Opplasting av gitt antall bruker klasser (CN)
- Søk etter gitt antall nøkkel attributter
- Søk etter gitt antall ikke nøkkel attributter i brukere (eksempel attributt beskrivelse)
- 3 endringer av attributter i alle brukere (endre, legg til og slett attributt)
- Sletting av alle bruker klasser
- Sletting av alle orginasjons klasser

Ved førstegangs kjøring av testen må det genereres en ldif[77] fil for organisasjonsklassene og brukerklassene som blir gjenbrukt når testen skal kjøres under ulike forhold. Dette kan være kjøring under nativ eller ulike visualiseringsplattformer. På den måten blir testen repiterbar og resultatene kan settes opp imot hverandre. Grunnen til at det skal lages en ldapfil er at man skal ha muligheten til å velge hvor mange brukere testen skal ha, og hvor lavt en organisasjonsklasse kan være under en annen organisasjonsklasse. Under genereringen sørger scriptet for å spre brukerne utover alle ordinasjonsklassene for å få et ujevnt tre struktur i ldap-databasen. Dette gjøres for å få en mer realistisk spredning av antall brukerklasser under hver organisasjonsklasse.

5.2.4 TERMINAL TEST:

Test script [s.183] for å belaste Windows Server operativsystemer satt opp for håndtering av tynne klienter[78] over RDP protokollen[79] til Microsoft. Scriptet er laget for å kjøre under linux platformen med perl kompilator. For å kunne kommunisere med RDP protokollen til Microsoft bruker scriptet applikasjonen rdesktop[80]. For å kunne håndtere oppretting av brukere i Active diractory automatisk via script, må gjesteoprativsystemet med Windows Server 2008 ha en perl kompilator.

Et oppsett for terminaltesten [s.102] vil bestå av en ekstern klientmaskin med rdesktop og testscriptet. På hypervisoren er det to kjørende gjesteoprativsystemer hvor det ene er Windows Server med terminaltjenesten og det andre et dedikert oprativsystem for støyscriptet. Terminalserveren har et nettverks script lyttende etter kommando for oppretting av brukere. Etter en test er ferdigkjørt vil scriptet sørge for å fjerne brukerene den opprettet for testen og profildata som er laget ved innlogging. Hensikten med å fjerne profildataen og brukeren etter en test er å få hele prosessen til å generere profildata til å skje på nytt når testen kjøres flere ganger. Dermed får man ikke raskere pålogging på grunn av eksisterende brukerprofil. Oppretting av nye brukerprofiler er også ment for å belaste hypervisoren i testen.

Selve testen er innlogging av ny bruker via terminalserveren og oppretting av profildata for brukerne. I det brukeren er logget inn, blir brukeren raskt logget ut igjen via et påloggings skript. De målbare verdiene er tiden det tar å logge et gitt antall brukere inn og ut på terminalserveren. I tilegg til den totale tiden innlogging tar, blir tiden for hver bruker innlogging logget. Man kan dermed se alle innloggingstidene for hver bruker i en testkjøring. Dersom en terminalserver blir overbelastet vil man få terminaloppkoblinger som feiler, og en ny oppkoblig må skje. Antall feil en test eventuelt får, blir logget i antall og hvor mange sekunder fra start til vellykket oppkobling tar.

I tilegg til scriptet som belaster terminaltjenesten har scriptet også som de andre scriptene test Senario1 og Senario2 for å kunne kombinere testen med støyscriptet.

5.2.5 EPOST TEST

Test script [s.152] for å teste e-post mottak og håndtering av mottatte e-post direkte på serveren via IMAP. Testen er laget i perl og trenger modul filene i Net::IMAP::Simple for å kunne kommunisere med IMAP protokollen. Scriptet sender e-post ved bruk av postfix.

Et testoppsett [s.95] består av en ekstern klientmaskin for sending og lesing av epost. På hypervisoren har man to kjørende gjesteoperativsystemer med e-post tjener og et for støyscriptet.

Scriptet kjører i to faser, sending og mottak. Før kjøring kan man angi hvor mange e-poster som skal sendes ut, størrelsen på hver e-post, mapper for sotering av e-post og hvor mange e-post kontoer som skal være med i testen. Antall kontoer som blir angitt gir antall samtidige IMAP oppkoblinger for simulering av lesing av e-post og samtidige e-post sendinger.

Når første fase av testen er ferdig, som er sending av e-poster vil resultatet for testen være tiden det tok å sende ut alle e-postene til angitt antall adresser. Først nå kan fase 2 starte som skal simulere angitt samtidige brukere som skal behandle sine mottatte epost. Scriptet prøver og etterligner et e-post program på følgende måte.

- Registerer antall epost i mappen
- Ser på funksjonalitet som er på mappen index
- Sjekker lese statusen, e-post størrelse, og laster ned emne til hver epost
- Avhengig av gitt antall oppretting av nye mapper blir det spredd ut kopi av eposter utover mappene. Dette skal simulere sortering av epost av brukeren.
- Hver enkelt melding blir så lastet ned.
- Etter at alle meldinger er lastet ned blir en og en melding slettet.
- Etter at alle meldinger i index er slettet blir så alle mappene med epost kopiene slettet.

Etter i gjennomført test har man resultater på tid for sending, mottak og den totale tiden på hele prosessen. E-post størrelsen for alle brukermeldingene blir også logget.

5.2.6 MYSQL

Her brukes ferdige benchmark script for mysql[81]. Men for å kunne automatisere kjøringen av scriptene slik at de kan kjøres i testsenarioer vil serveren ha et perl script [s.166] for starting og lagring av resultat til en ekstern maskin utenfor hypervisoren. På den måten kan de ferdige scriptene kombineres med støyscriptet.

Oppsettet [s.100] består av en ekstern maskin for kontroll og lagring av MySQL resultatene. På hypervisoren vil to gjesteoprativsystemer kjøre. Det første oprativsystemet kjører mysql server og dens benchmark script mens det andre operativsystemet kjører støyscriptet.

Ved kjøring av testscriptene startes og styres dette fra den eksterne maskinen. Etter at testen er gjennomført blir resultatet sendt tilbake til den eksterne maskinen for logging.

5.2.7 FTP TEST

Består en ftp klient [s.160] og fil oppretter [s.162] skrevet i perl. Klienten trenger modul filene i Net::FTP for at perl FTP klienten skal fungere. På ftp serveren må oprativsystemet være en linux variant hvor vsftpd er installert som serverapplikasjon.

Oppsettet [s. 98] består av en ekstern maskin som fungerer som en klient. På hypervisoren har man to gjesteoperativsystemer. Et for ftp server og et for støyscriptet.

Ved førstegangs kjøring brukes det et script for generering av filer for ftp serveren. Scriptet vil lage filer i en bestemt størrelse og antall. Hvilken størrelse og antall man vil ha på filene kan redigeres i scriptet. Med filer på serveren kan en test gjennomføres. Dette gjøres ved å kjøre testscriptet på den eksterne klientmaskinen.

Ved en gjennomført test vil følgene tall bli presentert:

- Overføringshastighet for hver av filstørrelsene, tid og antall filer
- Totaltid, total størrelse i mb og gjennomsnitts overføringshastighet for hele testen

5.3 FREMGANGSMÅTE

Fremgangsmåten ved utføring av testene representeres gjennom testsenarioer og skript, som gruppen har utviklet. Ett testsenario viser oppsett, operativsystemer, skript og verdier av betydning for testen. Et skript blir brukt i og av ett testsenario til utføring av selve testen og generering av støy.

5.4 GJENNOMFØRING

Alle tester skal gjennomføres i henhold til testsenarioer som finnes som vedlegg i rapporten. Dette krever riktig oppsett av hypervisor og virtuelle maskiner.

Alle tester som gjennomføres skal ha samme utgangspunkt. For å oppnå dette må hypervisoren gjennomgå en omstart mellom hver test for tømming av data som befinner seg i cach[82] ram og virtuelt minne.

For å kunne kjøre scriptene forutsetter det at serverapplikasjoner er installert og konfigurert på forhånd. Scriptene er ikke i stand til å oppdage manglende applikasjoner eller feil konfigurasjoner som er nødvendig for den gjeldende testen. Det har ikke blitt laget noe grafisk grensesnitt for testscriptene. Resultatfilene som blir presentert blir kun laget i tekstfiler. Alle scriptene som er laget for testingene sender meldinger over nettverk i ren tekst for synkronisering av oppgaveutføringene. Scriptene er laget kun for lab testing og har derfor ingen nettverks sikkerhet.

5.4.1 INSTALLASJON OG OPPSETT AV VIRTUELLE MASKINER

Installasjon av virtuelle maskiner kan enkeltgjøres ved bruk av snapshot funksjonen som er tilgjenglig i Vmware esxi og XenServer. Det utføres først en grunnleggende installasjon av Cent OS 5.4 og Windows Server 2008 der det tas snapshot av en ren installasjon som blir kopiert opp flere ganger. Dette gjør oppretting av virtuelle maskiner raskere siden man kun gjennomfører installasjonen en gang og jobber med kopier av denne.

Følgende maskiner må installeres og settes opp for gjennomføringen av alle tester.

- Cent OS 5.4 Web Server(Apache)
- Cent OS 5.4 Epost Server
- Cent OS 5.4 FTP server
- Cent OS 5.4 Database server(Mysql)
- Cent OS 5.4 LDAP kontroller
- Windows Server 2008 Active Directory kontroller
- Windows Server 2008 terminal server

Vedlegg s.87 for hvordan disse maskinene skal settes opp og installert.

6 RESULTATER

Det er valgt å dele resultatet for arbeidet i to deler. I den ene delen vil det pressenters resultat i form av test metoden som er utviklet for stress testingen.

I den andre delen presenteres testresultater gruppen selv har produsert ved bruk av testmetodene som er utviklet.

6.1 METODEN

Metoden gruppen utviklet for stresstesting av virtualiseringsservere består hovedsakelig av flere forskjellige skript. Der skriptene er rettet mot real life testing av forskjellige serverapplikasjoner.

I tillegg til skript er det utarbeidet en rekke forskjellige testsenarioer som forsøker å teste forskjellige egenskaper til hypervisoren. Det finnes senarioer fra oppretting av baseline til mer komplekse tester der det forsøkes å se hvordan hypervisoren oppfører seg når det er forskjellige virtuelle maskiner som konkurrerer om ressursene.

Resultat av metoden består altså av skript og testsenarioer, disse er beskrevet og forklart i del 5 av rapporten og som vedlegg i denne rapporten.

6.2 TEST RESULTATER

Gruppen har selv valgt å utføre en rekke tester på hypervisoren. Det er valgt å gjennomføre alle testsenarioer av type 1 og 2. Grunnet tid er det valgt og ikke gjennomføre testing av testsenario 3 på noen av serverapplikasjonene og alle senarioer av webserver testing. Av samme grunn er alle tall og diagrammer som fremvises i denne delen av rapporten et resultat av en gjennomføring av testscenarioet. Selv om et testsenario er kjørt opp til 4 ganger etter hverandre tilhører dette en og samme test. Dette begrunnes med at forutsetningen er forskjellig for hver test på grunn av arbeidet som er gjort på hypervisoren i forkant av hver kjøring er forskjellige for alle tester.

Som nevnt er det usikkerhet rundt tallene som presenteres i denne rapporten, men gruppen har gjort enkle tester under utvikling av testmetoden som viser at den samme testen faktisk produserer tilnærmet det samme resultatet ved flere separate kjøringer. Men tallene er ikke uendelig sikre og det kan derfor ikke trekkes noen endelig konklusjon på dette grunnlaget. Resultatet kan benyttes som en pekepinne på hvilken hypervisor som yter best på gjeldende test som kjører. På grunn av at servere er sammensatt av forskjellige maskinvarekomponenter av ulike teknologier, vil våres testresultater være unik for vårt oppsett. Også versjon av opprativsystem og serverapplikasjoner kan påvirke resultatene ytterligere.

Om presentasjon av resultater

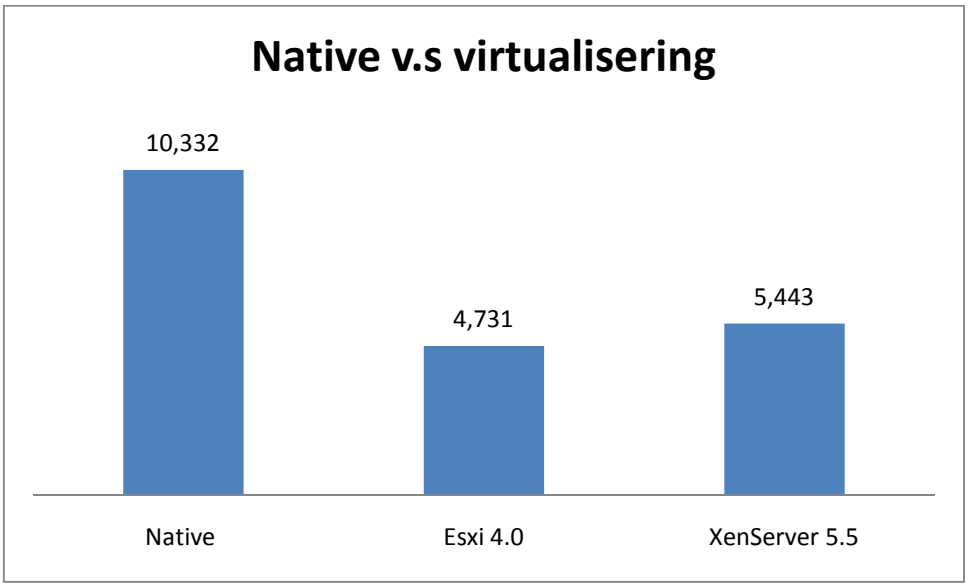
Alle verdier som representeres i denne delene er hentet fra log filer generert av testene som er kjørt. Log filene inneholder langt mer data enn det vises i denne

delen av rapporten, men det er valgt å fremvise den mest betydningsfulle verdien for hver test. Dette gjelder ikke for testing av AD og LDAP kontroller, siden at verdier generert i denne testen er av stor betydning i forhold til ytelsen i hypervisoren.

6.2.1 FTP SERVER

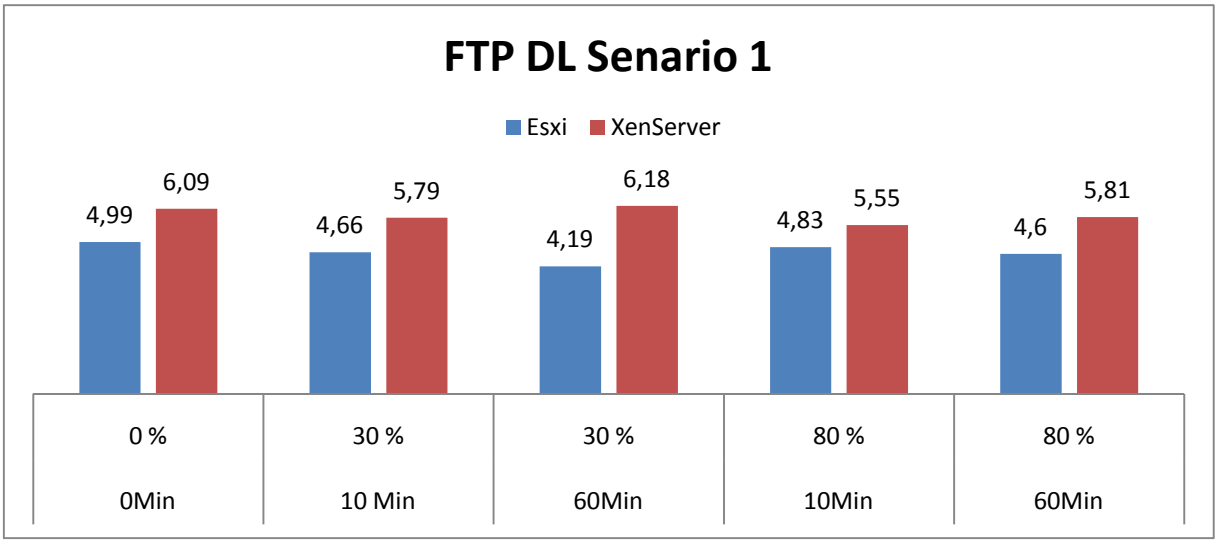
FTP testene er gjennomførte på en virtuell maskin av typen CentOS 5.4 på virtualiserings serveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til test senarioene [FTP Senario, Senario 1, Senario 2]. FTP serveren er satt opp i henhold til vedlegg "Konfigurasjons Oppsett for FTP test, s 99". Følgende resultater er hentet fra testene.

FTP NEDLASTING



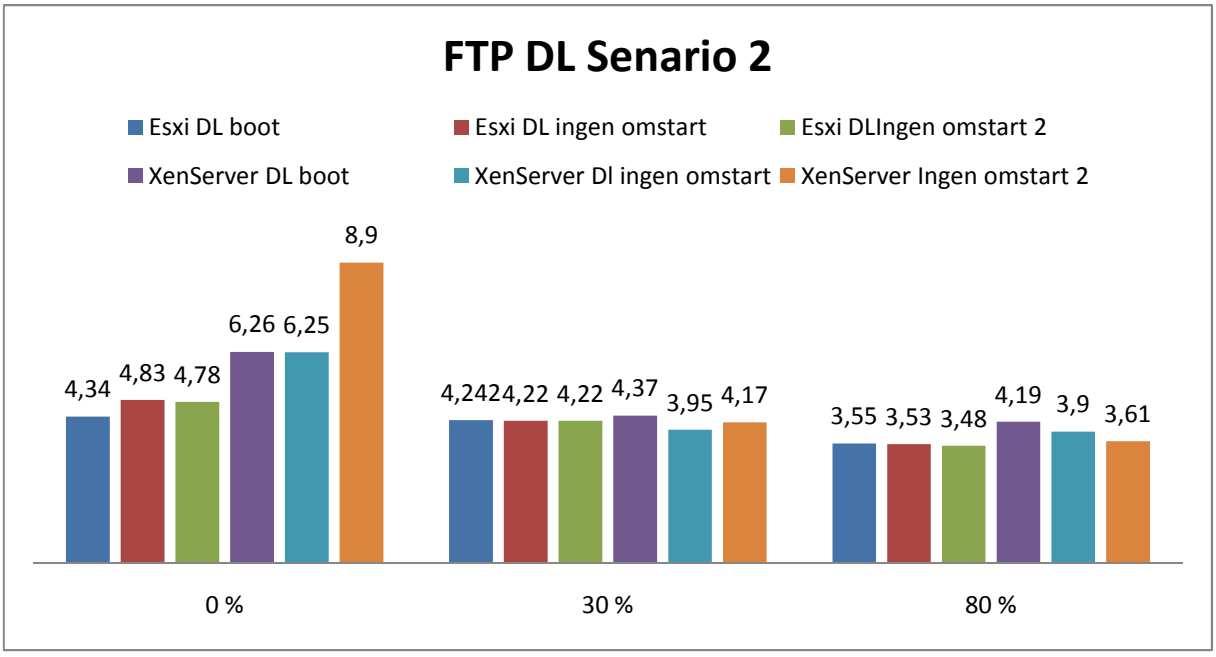
Figur 7 – Native satt opp mot virtulaiserings plattformer for FTP nedlastings test.

Figur 7 viser resultatet av en FTP nedlastnings test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene som vises er i MB/s for nedlastingen av data fra ftp serveren.



Figur 8 – FTP Nedlastings senario 1.

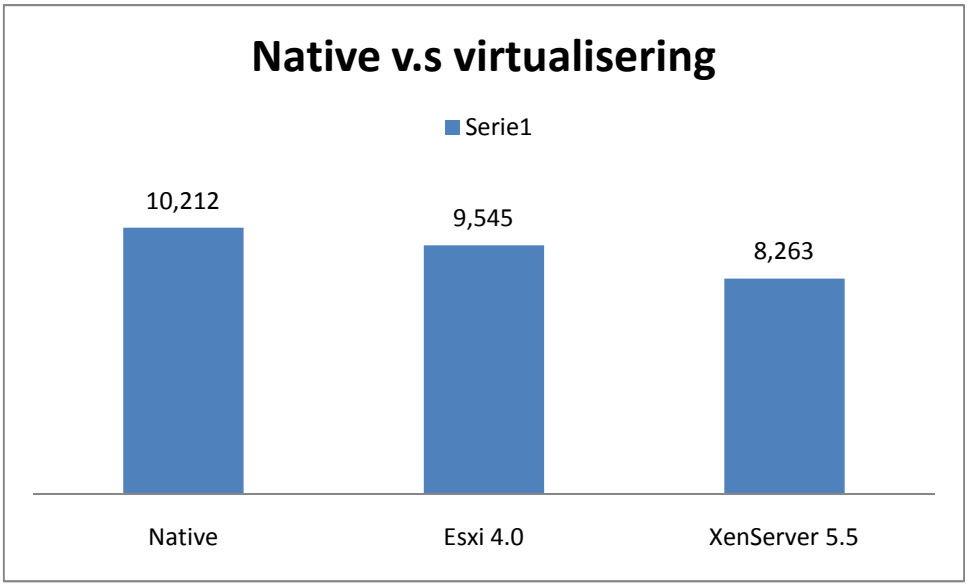
Figur 8 viser resultat av en FTP senario 1nedlastnings test. Verdiene i søylene representer MB/s for nedlasting. Denne testen er kjørt en gang for hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkelt ftp test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt i angitt minutter før hver test.



Figur 9 – FTP Nedlastings senario 2.

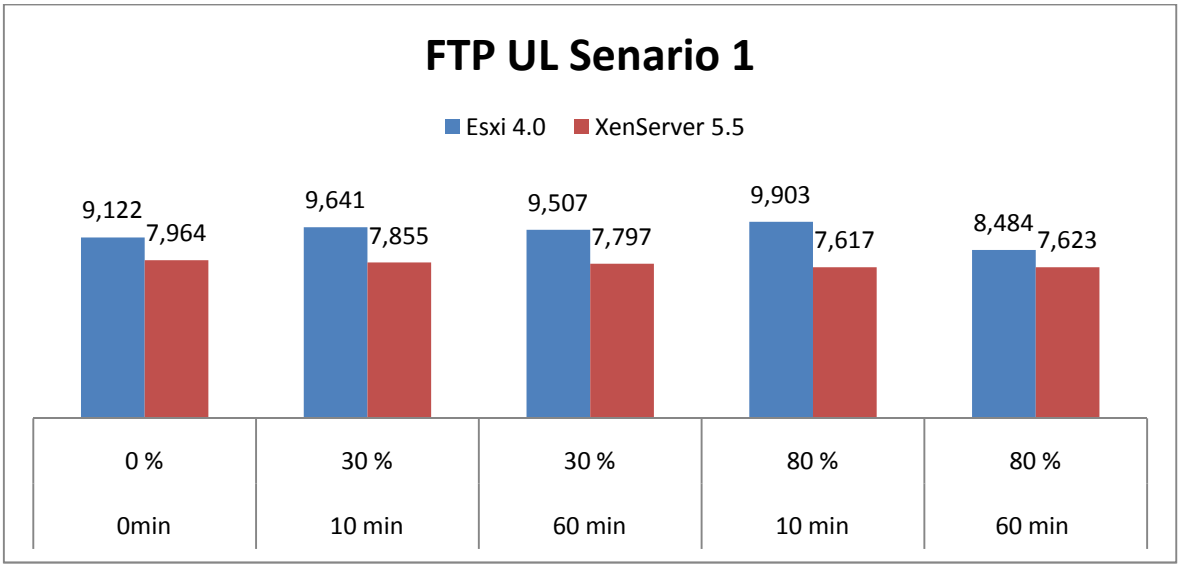
Figur 9 viser resultat av en FTP senario 2 nedlastnings test. Verdiene i søylene representer MB/s for nedlasting. Denne testen er kjørt sammenhengende tre ganger for hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkelt ftp test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt samtidig med testen.

FTP UPPLOAD



Figur 10 – Native satt opp mot virtulaiserings plattformer for FTP opplastings test.

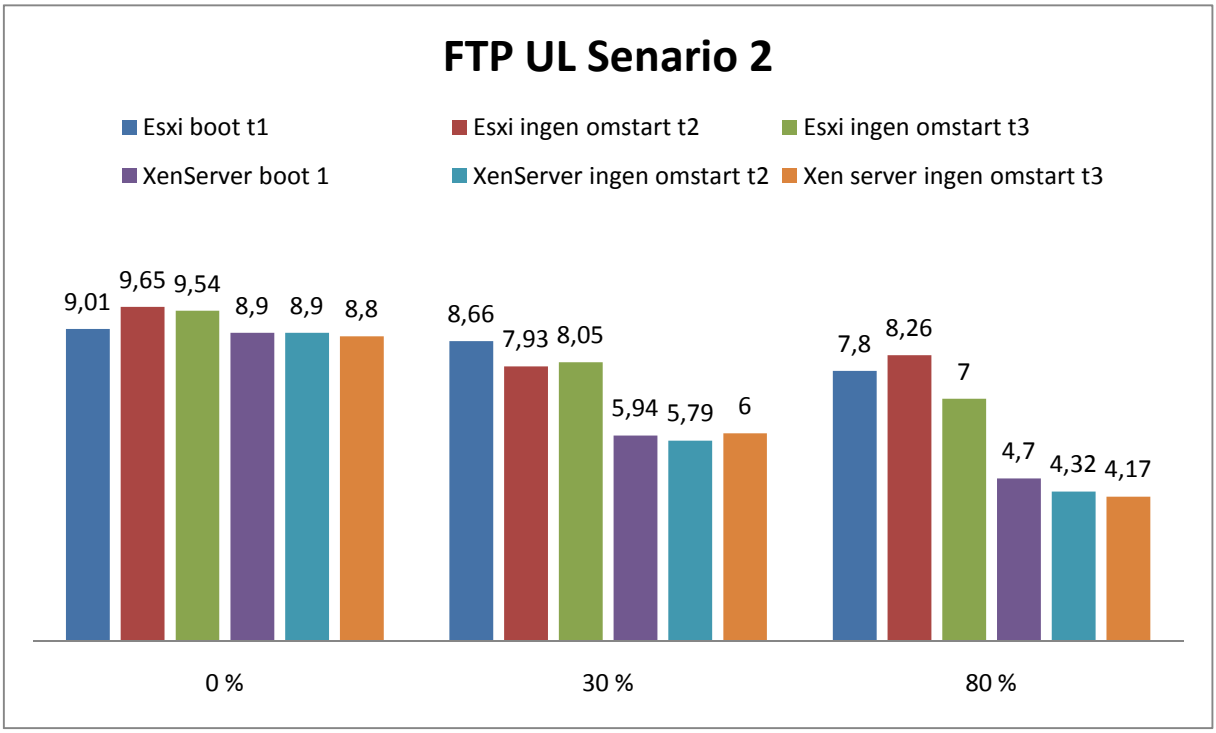
Figur 10 viser resultatet av en FTP opplastings test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene som vises er i MB/s for opplastning av data fra ftp serveren.



Figur 11 – FTP opplastings senario 1.

Figur 11 viser resultat av en FTP senario 1 opplastings test. Verdiene i søylene representer MB/s for opplasting. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkelt ftp test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt i angitt minutter før hver test.

Senario 2



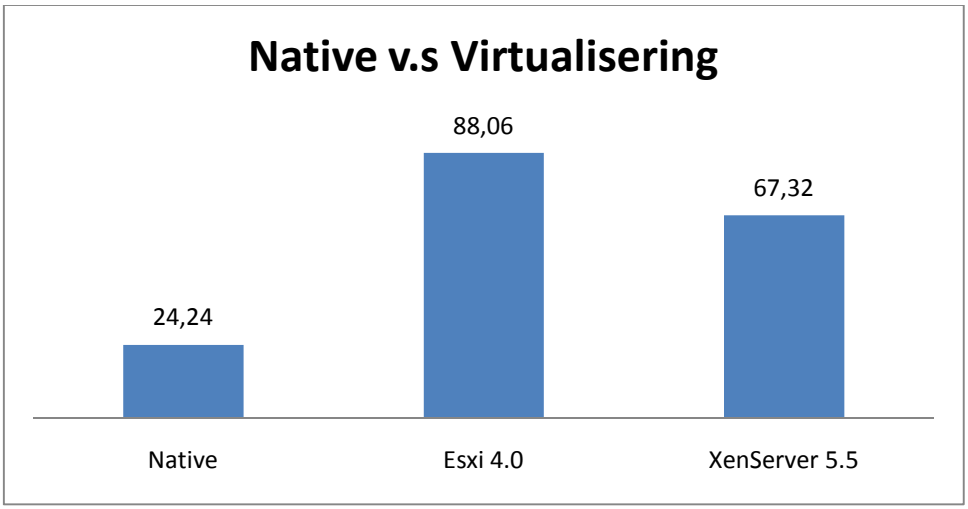
Figur 12 – FTP opplastings senario 2.

Figur 12 viser resultat av en FTP senario 2 opplastings test. Verdiene i søylene representer MB/s for nedlasting. Denne testen er kjørt sammenhengende tre ganger på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkelt ftp test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt samtidig med testen.

6.2.2 TERMINAL SERVER

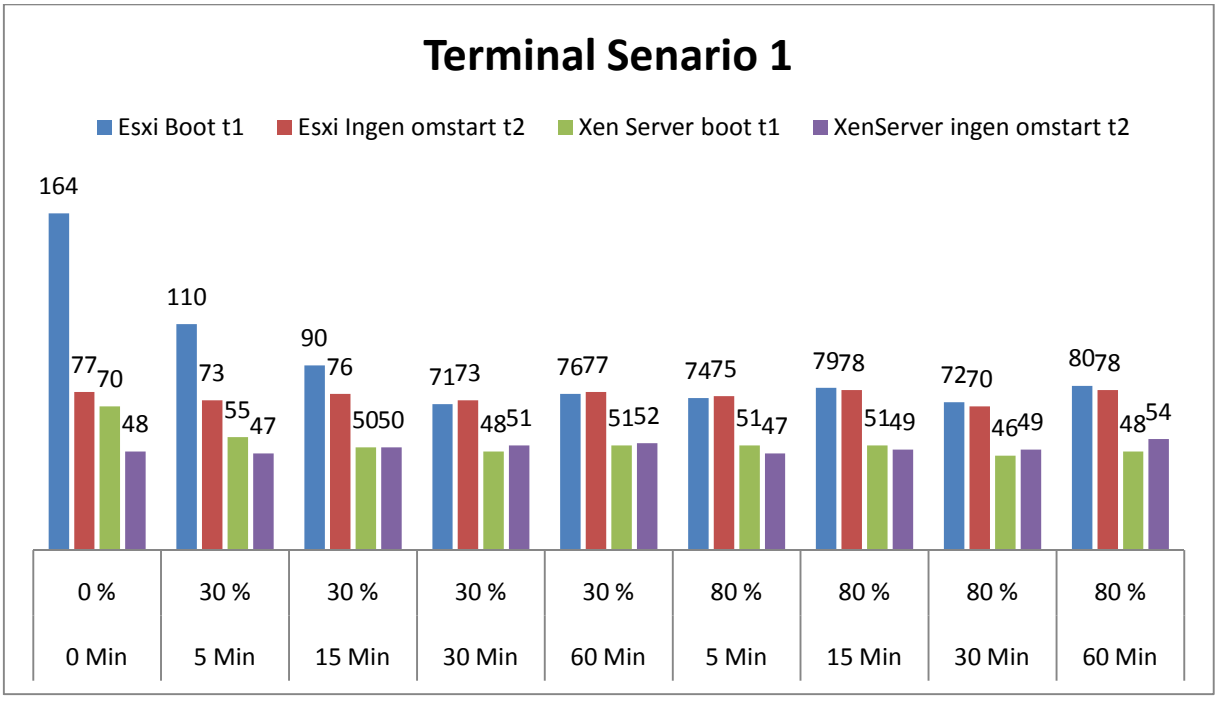
Terminalservertestene er gjennomførte på en virtuell maskin av typen Windows Server 2008 på virtualiserings serveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til test senarioene [Terminal Senario, Senario 1, Senario 2] Terminal serveren er satt opp i henhold til vedlegg "Konfigurasjons oppsett for Microsoft Terminal Server Test, s 102".

Virtualisering:



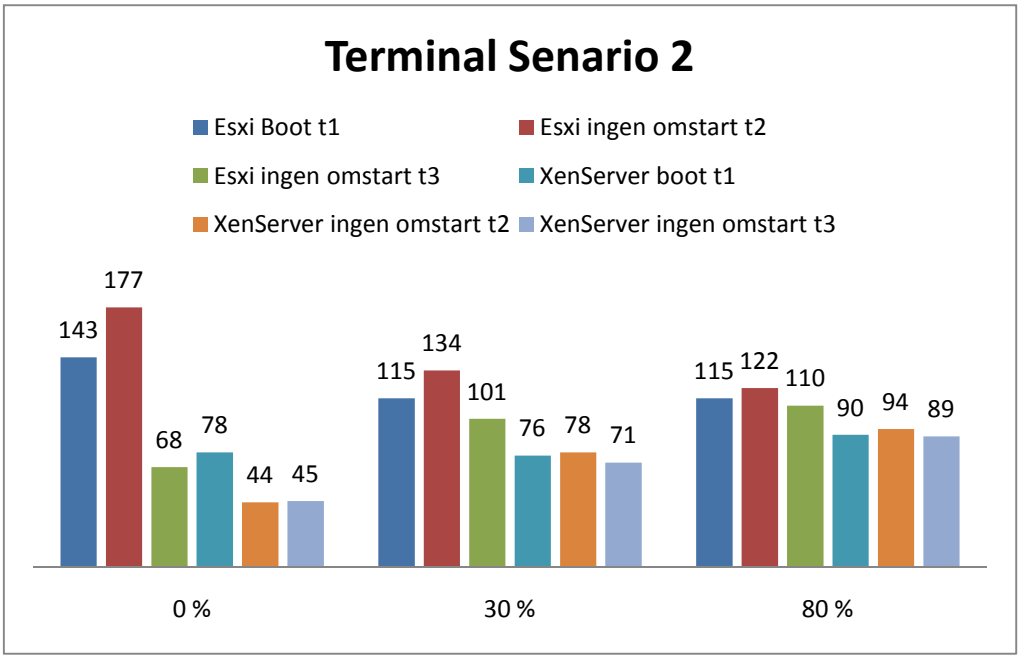
Figur 13 – Native satt opp mot virtulaiserings plattformer for Terminal server test.

Figur 13 viser resultatet av en terminalservertest. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene som vises er totaltiden i sekunder for gjennomføring av testen.



Figur 14 – Terminal server senario 1.

Figur 14 viser resultat av en terminal senario 1 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel terminaltest uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt i angitt minutter før hver test.

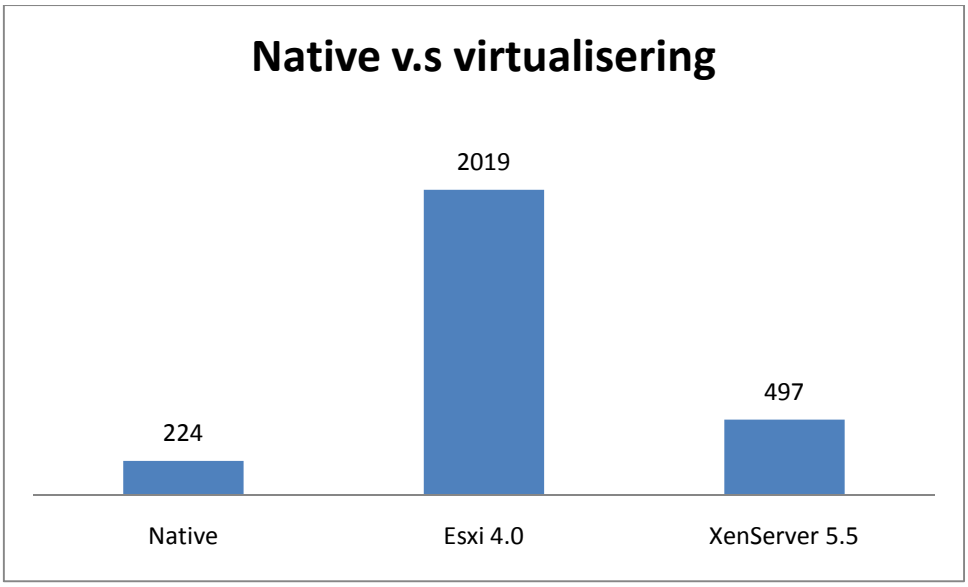


Figur 15 – Terminal server senario 2.

Figur 15 viser resultat av en terminal senario 2 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt sammenhengende tre ganger på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisoerne som består av en enkel terminaltest uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt samtidig med testen.

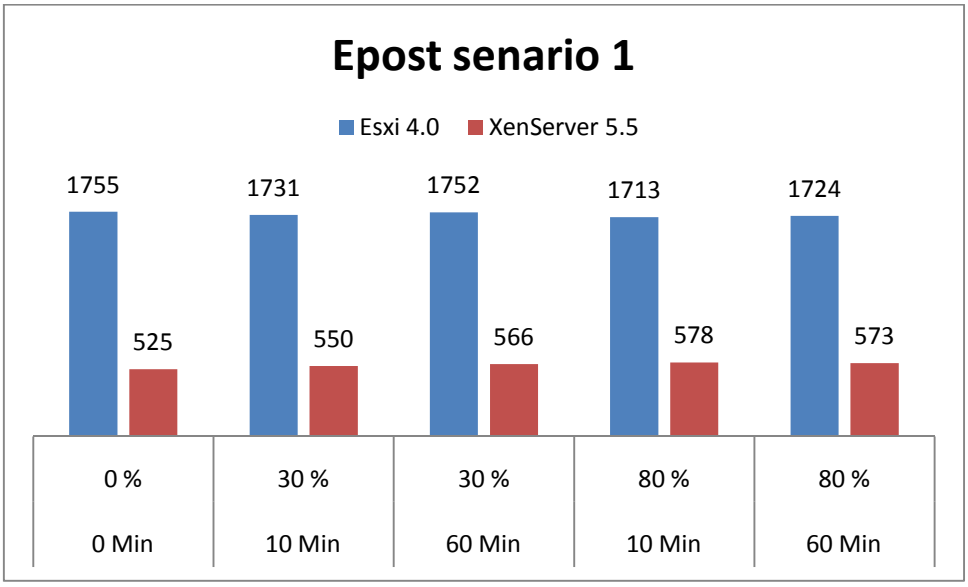
6.2.3 EPOST SERVER

E-post servertestene er gjennomførte på en virtuell maskin av typen Cent OS 5.4 på virtualiseringsserveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til testsenarioene [E-post Senario, Senario 1, Senario 2]. Epost serveren er satt opp i henhold til vedlegg "Konfigurasjons oppsett for e-post test, s96".



Figur 16 – Native satt opp mot virtulaiserings plattformer for epost server test.

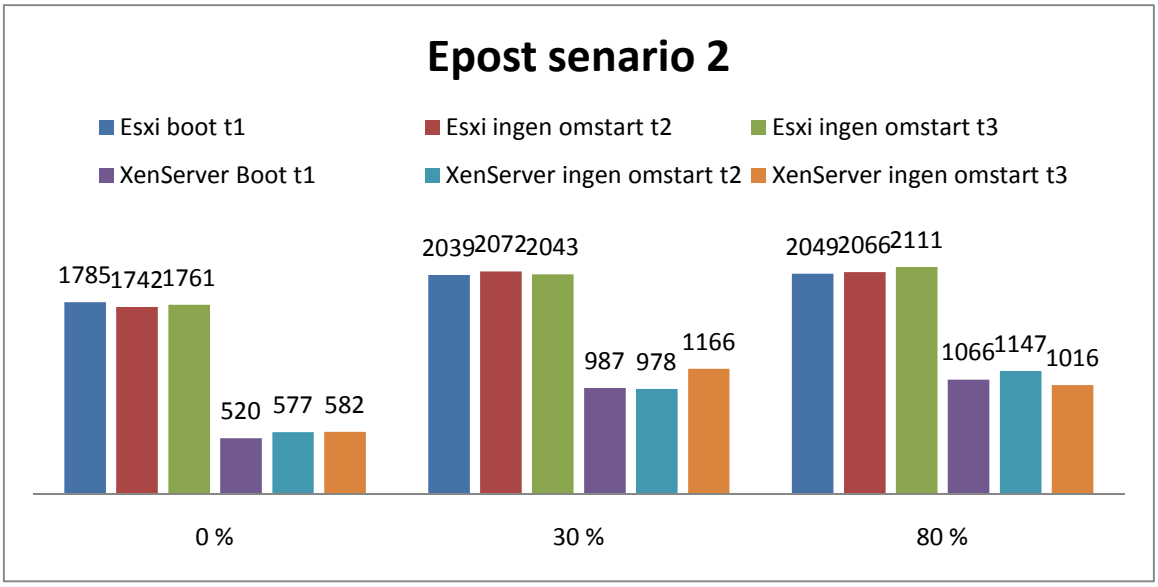
Figur 16 viser resultatet av e-post test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene viser totaltiden i sekunder for gjennomføring av testen.



Figur 17 – Epost server senario 1.

Figur 17 viser resultat av en e-post senario 1 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt en gang på

hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisoerne som består av en enkel e-post test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt i angitt minutter før hver test.

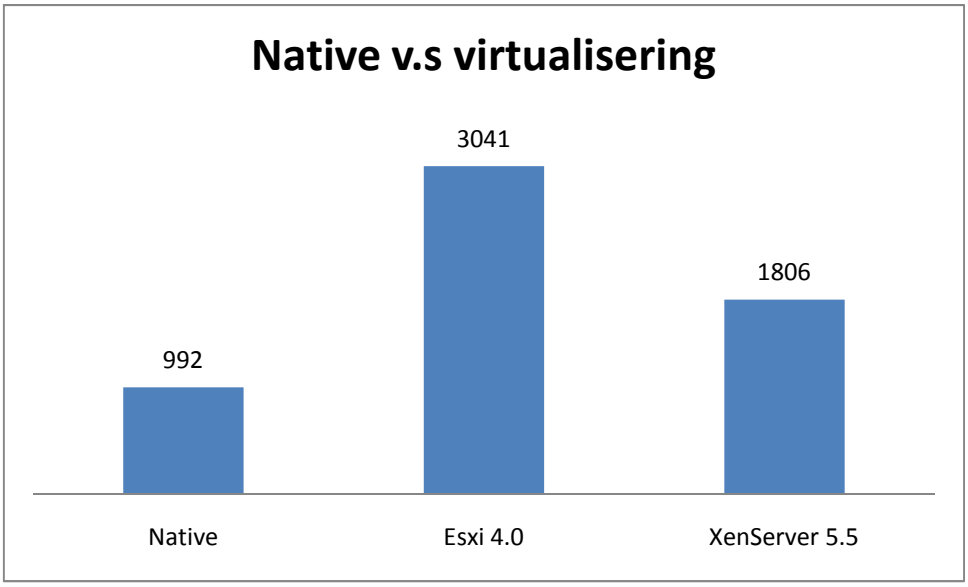


Figur 18 – E-post server senario 2.

Figur 18 viser resultat av en e-post senario 2 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt sammenhengende tre ganger på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisoerne som består av en enkel e-posttest uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt samtidig med testen.

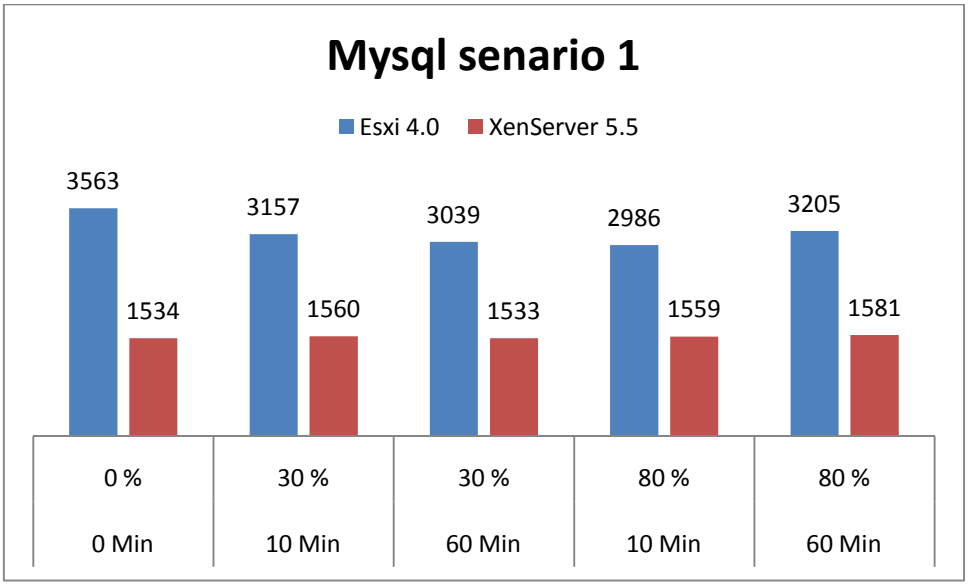
6.2.4 DATABASE SERVER

Database servertestene er gjennomførte på en virtuell maskin av typen Cent OS 5.4 på virtualiseringsserveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til testsenarioene [Database Senario, Senario 1, Senario 2]. Database serveren er satt opp i henhold til vedlegg "Konfigurasjons oppsett for Mysql test, s101".



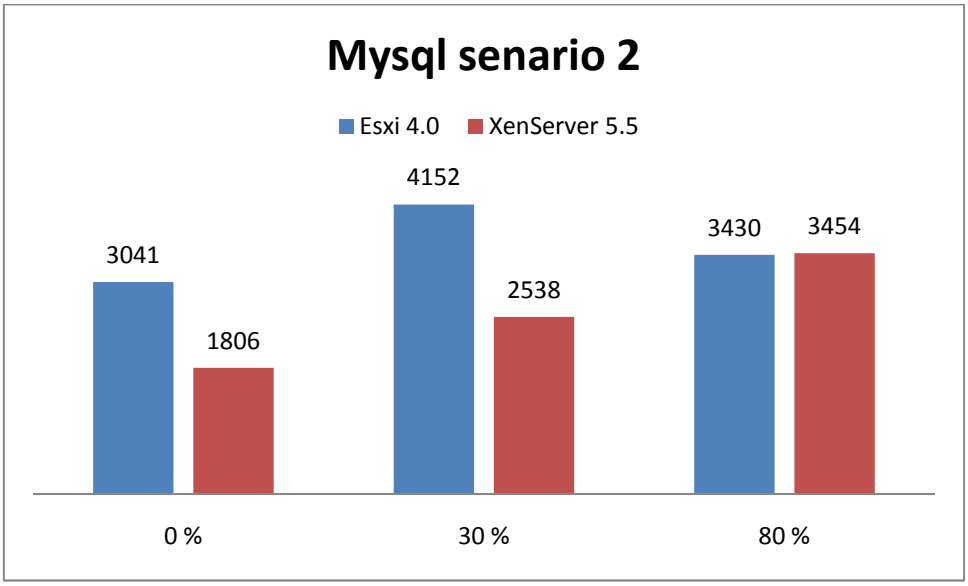
Figur 19 – Native satt opp mot virtulaiserings plattformer for database server test.

Figur 19 viser resultatet av en mysql datbase test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene viser totaltiden i sekunder for gjennomføring av testen.



Figur 20 – Database server senario 1.

Figur 20 viser resultat av en database senario 1 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel database test uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt i angitt minutter før hver test.



Figur 21 – Database server senario 2.

Figur 21 viser resultat av en database senario 2 test. Verdiene i søylene representer totaltiden i sekunder for gjennomføring av testen. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel databasetest uten noen form for støy. Prosent mengden er kalkulert støy mengde kjørt samtidig med testen.

6.2.5 LDAP KONTROLLER

Ldap kontroller testene er gjennomførte på en virtuell maskin av typen Cent OS 5.4 på virtualiserings serveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til testsenarioene [LDAP Senario, Senario1 , Senario 2]. LDAP kontrolleren er satt opp i henhold til vedlegg "Konfigurasjons oppsett for LDAP test på OPENLDAP, s94".

Det er i dette tilfellet valgt og ikke fremstille resultatet i ett diagram. Dette har med at denne typen tester tester ikke har noen totaltid eller totalfart som kan representere det endelige resultatet. I denne typen test er det mange forskjellige operasjoner som utføres, og som benytter seg av forskjellige ressurser i maskinen. Det er derfor valgt å fremvise alle de målte verdiene i tabeller.

Native	VMware Esxi 4.0	Citrix XenServer 5.5
Last opp Org (76): 0.332367 - 228.663 i sek	Last opp Org (76): 2.720453 - 27.937 i sek	Last opp Org (76): 0.544649 - 139.539 i sek
Last opp Brukere (3000): 132.858343 - 22.580 i sek	Last opp Brukere (3000): 1772.744689 - 1.692 i sek	Last opp Brukere (3000): 220.065924 - 13.632 i sek
Sok etter CN: 3.665962 - 272.780 i sek	Sok etter CN: 4.578617 - 218.407 i sek	Sok etter CN: 5.714907 - 174.981 i sek
Sok etter Atributter: 94.878489 - 10.540 i sek	Sok etter Atributter: 109.066488 - 9.169 i sek	Sok etter Atributter: 115.233129 - 8.678 i sek
Enring av brukere: 79.131534 - 37.912 i sek	Enring av brukere: 766.502404 - 3.914 i sek	Enring av brukere: 130.847982 - 22.927 i sek
Slette alle brukere: 190.854515 - 15.719 i sek	Slette alle brukere: 2568.548856 - 1.168 i sek	Slette alle brukere: 329.972024 - 9.092 i sek
Slette alle Org: 0.616846 - 123.207 i sek	Slette alle Org: 3.166368 - 24.002 i sek	Slette alle Org: 1.004321 - 75.673 i sek

Tabell 1 – Native satt opp mot virtualiserings plattformer for LDAP test.

Tabell 1 viser resultatet av en LDAP test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene i tabellen viser mengden av en gitt operasjon og hvor mange operasjoner i sekundet for den gitte operasjon som ble utført.

Vmware Esxi 4.0	Citrix XenServer 5.5
StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 4.535981 - 16.755 i sek Last opp Brukere (3000): 1825.708589 - 1.643 i sek Sok etter CN: 4.684371 - 213.476 i sek Sok etter Atributter: 122.015234 - 8.196 i sek Enring av brukere: 844.023975 - 3.554 i sek Slette alle brukere: 2677.949031 - 1.120 i sek Slette alle Org: 4.972829 - 15.283 i sek	StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 0.480945 - 158.022 i sek Last opp Brukere (3000): 235.318756 - 12.749 i sek Sok etter CN: 5.892253 - 169.714 i sek Sok etter Atributter: 115.14586 - 8.685 i sek Enring av brukere: 144.677311 - 20.736 i sek Slette alle brukere: 346.412443 - 8.660 i sek Slette alle Org: 0.804458 - 94.474 i sek
StoyProsent: 30 - StoyKjoreTid: 600 Last opp Org (76): 3.817151 - 19.910 i sek Last opp Brukere (3000): 1845.253236 - 1.626 i sek Sok etter CN: 4.489234 - 222.755 i sek Sok etter Atributter: 135.330606 - 7.389 i sek Enring av brukere: 844.012122 - 3.554 i sek Slette alle brukere: 2777.531265 - 1.080 i sek Slette alle Org: 4.968969 - 15.295 i sek	StoyProsent: 30 - StoyKjoreTid: 600 Last opp Org (76): 1.796207 - 42.311 i sek Last opp Brukere (3000): 235.582876 - 12.734 i sek Sok etter CN: 5.821883 - 171.766 i sek Sok etter Atributter: 124.427398 - 8.037 i sek Enring av brukere: 142.875121 - 20.997 i sek Slette alle brukere: 375.07181 - 7.998 i sek Slette alle Org: 0.919409 - 82.662 i sek
StoyProsent: 30 - StoyKjoreTid: 3600 Last opp Org (76): 4.311837 - 17.626 i sek Last opp Brukere (3000): 1881.565375 - 1.594 i sek Sok etter CN: 4.946834 - 202.149 i sek Sok etter Atributter: 138.257618 - 7.233 i sek Enring av brukere: 889.596525 - 3.372 i sek Slette alle brukere: 2756.274012 - 1.088 i sek Slette alle Org: 5.120821 - 14.841 i sek	StoyProsent: 30 - StoyKjoreTid: 3600 Last opp Org (76): 2.981098 - 25.494 i sek Last opp Brukere (3000): 242.768568 - 12.357 i sek Sok etter CN: 5.943303 - 168.257 i sek Sok etter Atributter: 139.677882 - 7.159 i sek Enring av brukere: 144.661718 - 20.738 i sek Slette alle brukere: 359.807315 - 8.338 i sek Slette alle Org: 0.815174 - 93.232 i sek
StoyProsent: 80 - StoyKjoreTid: 600 Last opp Org (76): 3.429567 - 22.160 i sek Last opp Brukere (3000): 1848.172287 - 1.623 i sek	StoyProsent: 80 - StoyKjoreTid: 600 Last opp Org (76): 0.958737 - 79.271 i sek Last opp Brukere (3000): 238.101859 - 12.600 i sek

Sok etter CN: 5.504782 - 181.660 i sek Sok etter Atributter: 156.512272 - 6.389 i sek Enring av brukere: 932.496849 - 3.217 i sek Slette alle brukere: 2748.328234 - 1.092 i sek Slette alle Org: 5.002189 - 15.193 i sek	Sok etter CN: 6.144677 - 162.742 i sek Sok etter Atributter: 151.581224 - 6.597 i sek Enring av brukere: 142.714202 - 21.021 i sek Slette alle brukere: 343.696744 - 8.729 i sek Slette alle Org: 1.378306 - 55.140 i sek
StoyProsent: 80 - StoyKjoreTid: 3600 Last opp Org (76): 3.966486 - 19.161 i sek Last opp Brukere (3000): 1929.212989 - 1.555 i sek Sok etter CN: 4.602388 - 217.279 i sek Sok etter Atributter: 183.351493 - 5.454 i sek Enring av brukere: 873.473706 - 3.435 i sek Slette alle brukere: 2794.472707 - 1.074 i sek Slette alle Org: 4.633469 - 16.402 isek	StoyProsent: 80 - StoyKjoreTid: 3600 Last opp Org (76): 2.531558 - 30.021 i sek Last opp Brukere (3000): 234.519375 - 12.792 i sek Sok etter CN: 5.805938 - 172.237 i sek Sok etter Atributter: 164.291043 - 6.087 i sek Enring av brukere: 143.107967 - 20.963 i sek Slette alle brukere: 368.140695 - 8.149 i sek Slette alle Org: 0.898179 - 84.616 i sek

Tabell 2 – LDAP Senario 1.

Tabell 2 viser resultat etter en LDAP senario 1 test. Verdiene i tabellen viser typen operasjon, mengden av den gitte operasjon og antall slike operasjoner i sekundet. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel LDAP test uten noen form for støy.

VMware esxi 4.0	Citrix Xenserver 5.5
StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 2.906719 - 26.146 i sek Last opp Brukere (3000): 1724.415822 - 1.740 i sek Sok etter CN: 4.966581 - 201.346 i sek Sok etter Atributter: 144.270836 - 6.931 i sek Enring av brukere: 779.262391 - 3.850 i sek Slette alle brukere: 2490.929878 - 1.204 i sek Slette alle Org: 3.590323 - 21.168 i sek	StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 1.023465 - 74.258 i sek Last opp Brukere (3000): 244.396988 - 12.275 i sek Sok etter CN: 5.99069 - 166.926 i sek Sok etter Atributter: 143.089457 - 6.989 i sek Enring av brukere: 147.620423 - 20.322 i sek Slette alle brukere: 361.406098 - 8.301 i sek Slette alle Org: 0.832883 - 91.249 i sek
StoyProsent: 30 - StoyKjoreTid:	StoyProsent: 30 - StoyKjoreTid:

SAMTIDIG Last opp Org (76): 4.22663 - 17.981 i sek Last opp Brukere (3000): 1992.992606 - 1.505 i sek Sok etter CN: 6.026755 - 165.927 i sek Sok etter Atributter: 223.192349 - 4.480 i sek Enring av brukere: 851.125138 - 3.525 i sek Slette alle brukere: 2864.588808 - 1.047 i sek Slette alle Org: 5.403437 - 14.065 i sek	SAMTIDIG Last opp Org (76): 4.698256 - 16.176 i sek Last opp Brukere (3000): 787.61393 - 3.809 i sek Sok etter CN: 6.120068 - 163.397 i sek Sok etter Atributter: 158.714223 - 6.301 i sek Enring av brukere: 279.67872 - 10.727 i sek Slette alle brukere: 1121.167142 - 2.676 i sek Slette alle Org: 0.902625 - 84.199 i sek
StoyProsent: 80 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 3.422875 - 22.204 i sek Last opp Brukere (3000): 2009.436712 - 1.493 i sek Sok etter CN: 5.809007 - 172.146 i sek Sok etter Atributter: 193.295971 - 5.173 i sek Enring av brukere: 855.031279 - 3.509 i sek Slette alle brukere: 3170.904808 - 0.946 i sek Slette alle Org: 4.654161 - 16.329 i sek	StoyProsent: 80 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 0.822692 - 92.380 i sek Last opp Brukere (3000): 780.312041 - 3.845 i sek Sok etter CN: 7.604482 - 131.501 i sek Sok etter Atributter: 233.477069 - 4.283 i sek Enring av brukere: 340.588944 - 8.808 i sek Slette alle brukere: 1589.663068 - 1.887 i sek Slette alle Org: 1.811468 - 41.955 i sek

Tabell 3 – LDAP Senario 2.

Tabell 3 viser resultat etter en LDAP senario 2 test. Verdiene i tabellen viser typen operasjon, mengden av den gitte operasjon og antall slike operasjoner i sekundet. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel LDAP test uten noen form for støy.

6.2.6 AD KONTROLLER

AD kontroller testene er gjennomførte på en virtuell maskin av typen Windows Server 2008 på virtualiserings serveren VMware Esxi 4.0 og Citrix XenServer 5.5 i henhold til testsenarioene [AD Senario, Senario 1, Senario "]. AD kontrolleren er satt opp i henhold til vedlegg "Konfigurasjons oppsett for LDAP test på Microsfot Active Directory,s 93".

I likhet med LDAP testen er det også valgt å ikke å fremstille resultatet i ett diagram. Dette har med at denne typen tester ikke har noen totaltid eller totalfart som kan representere det endelige resultatet. I denne typen test er det mange forskjellige operasjoner som utføres som benytter seg av forskjellige ressurser i maskinen. Det er derfor valgt å fremvise alle de målte verdiene i tabeller.

Native	VMware Esxi 4.0	Citrx XenServer 5.5
Last opp Org (76): 0.592197 - 128.336 i sek	Last opp Org (76): 3.797464 - 20.013 i sek	Last opp Org (76): 3.286511 - 23.125 i sek
Last opp Brukere (3000): 27.371238 - 109.604 i sek	Last opp Brukere (3000): 150.158081 - 19.979 i sek	Last opp Brukere (3000): 225.771043 - 13.288 i sek
Sok etter CN: 8.117729 - 123.187 i sek	Sok etter CN: 9.537298 - 104.851 i sek	Sok etter CN: 8.290595 - 120.619 i sek
Sok etter Atributter: 324.492525 - 3.082 i sek	Sok etter Atributter: 372.508601 - 2.685 i sek	Sok etter Atributter: 355.586083 - 2.812 i sek
Enring av brukere: 48.766495 - 61.518 i sek	Enring av brukere: 280.953524 - 10.678 i sek	Enring av brukere: 277.697634 - 10.803 i sek
Slette alle brukere: 38.4343 - 78.055 i sek	Slette alle brukere: 151.722279 - 19.773 i sek	Slette alle brukere: 148.937426 - 20.143 i sek
Slette alle Org: 0.644627 - 117.898 i sek	Slette alle Org: 3.146631 - 24.153 i sek	Slette alle Org: 3.359749 - 22.621 i sek

Tabell 4 – Native satt opp mot virtualiserings plattformer for Active Directory test.

Tabell 4 viser resultatet av en AD test. Vi ser her forskjeller i ytelse på Esxi og XenServer kontra native ytelse. Tallene i tabellen viser mengden av en gitt operasjon og hvor mange operasjoner i sekundet for den gitte operasjon som ble utført.

Vmware Esxi 4.0	Citrix XenServer 5.5
StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 2.650465 - 28.674 i sek Last opp Brukere (3000): 139.922925 - 21.440 i sek Sok etter CN: 9.643587 - 103.696 i sek Sok etter Atributter: 333.15225 - 3.002 i sek Enring av brukere: 283.018839 - 10.600 i sek Slette alle brukere: 144.841271 - 20.712 i sek Slette alle Org: 3.176899 - 23.923 i sek	StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 3.056924 - 24.862 i sek Last opp Brukere (3000): 119.814792 - 25.039 i sek Sok etter CN: 8.837124 - 113.159 i sek Sok etter Atributter: 351.944717 - 2.841 i sek Enring av brukere: 277.747362 - 10.801 i sek Slette alle brukere: 144.285536 - 20.792 i sek Slette alle Org: 3.06374 - 24.806 i sek
StoyProsent: 30 - StoyKjoreTid: 600 Last opp Org (76): 4.731213 - 16.064 i sek Last opp Brukere (3000): 145.001837 - 20.689 i sek Sok etter CN: 9.558208 - 104.622 i sek Sok etter Atributter: 355.264389 - 2.815 i sek Enring av brukere: 282.965133 - 10.602 i sek Slette alle brukere: 150.689237 - 19.909 i sek Slette alle Org: 3.133928 - 24.251 i sek	StoyProsent: 30 - StoyKjoreTid: 600 Last opp Org (76): 4.144759 - 18.336 i sek Last opp Brukere (3000): 121.203911 - 24.752 i sek Sok etter CN: 8.391854 - 119.163 i sek Sok etter Atributter: 347.162118 - 2.880 i sek Enring av brukere: 306.716686 - 9.781 i sek Slette alle brukere: 147.747142 - 20.305 i sek Slette alle Org: 3.877968 - 19.598 i sek
StoyProsent: 30 - StoyKjoreTid: 3600 Last opp Org (76): 4.155888 - 18.287 i sek Last opp Brukere (3000): 143.974364 - 20.837 i sek Sok etter CN: 9.577413 - 104.412 i sek Sok etter Atributter: 377.031383 - 2.652 i sek Enring av brukere: 290.299364 - 10.334 i sek Slette alle brukere: 147.161186 - 20.386 i sek Slette alle Org: 3.17158 - 23.963 i sek	StoyProsent: 30 - StoyKjoreTid: 3600 Last opp Org (76): 3.580283 - 21.227 i sek Last opp Brukere (3000): 123.330656 - 24.325 i sek Sok etter CN: 8.820346 - 113.374 i sek Sok etter Atributter: 350.875024 - 2.850 i sek Enring av brukere: 279.974666 - 10.715 i sek Slette alle brukere: 141.473413 - 21.205 i sek Slette alle Org: 2.999487 - 25.338 i sek
StoyProsent: 80 - StoyKjoreTid: 600 Last opp Org (76): 2.830517 - 26.850 i sek Last opp Brukere (3000): 143.030478 - 20.975 i sek Sok etter CN: 10.260593 - 97.460 i	StoyProsent: 80 - StoyKjoreTid: 600 Last opp Org (76): 3.024863 - 25.125 i sek Last opp Brukere (3000): 119.775849 - 25.047 i sek Sok etter CN: 9.049574 - 110.502 i

sek Sok etter Atributter: 340.802208 - 2.934 i sek Enring av brukere: 287.20404 - 10.446 i sek Slette alle brukere: 149.0913 - 20.122 i sek Slette alle Org: 3.269984 - 23.242 i sek	sek Sok etter Atributter: 367.502989 - 2.721 i sek Enring av brukere: 279.965639 - 10.716 i sek Slette alle brukere: 141.980431 - 21.130 i sek Slette alle Org: 3.03955 - 25.004 i sek
StoyProsent: 80 - StoyKjoreTid: 3600 Last opp Org (76): 3.571569 - 21.279 i sek Last opp Brukere (3000): 141.903782 - 21.141 i sek Sok etter CN: 9.701123 - 103.081 i sek Sok etter Atributter: 357.485593 - 2.797 i sek Enring av brukere: 284.317639 - 10.552 i sek Slette alle brukere: 146.410812 - 20.490 i sek Slette alle Org: 3.410091 - 22.287 i sek	StoyProsent: 80 - StoyKjoreTid: 3600 Last opp Org (76): 4.71682 - 16.113 i sek Last opp Brukere (3000): 121.055251 - 24.782 i sek Sok etter CN: 8.887446 - 112.518 i sek Sok etter Atributter: 380.991447 - 2.625 i sek Enring av brukere: 285.787485 - 10.497 i sek Slette alle brukere: 142.370693 - 21.072 i sek Slette alle Org: 3.106237 - 24.467 i sek

Tabell 5 – Active Directory Senario 1.

Tabell 5 viser resultat etter en AD senario 1 test. Verdiene i tabellen viser typen operasjon, mengden av den gitte operasjon og antall slike operasjoner i sekundet. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervioserne som består av en enkel AD test uten noen form for støy.

VMware esxi 4.0	Citrix XenServer 5.5
StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 2.673518 - 28.427 i sek Last opp Brukere (3000): 134.623937 - 22.284 i sek Sok etter CN: 9.670709 - 103.405 i sek Sok etter Atributter: 331.240422 - 3.019 i sek Enring av brukere: 287.211947 - 10.445 i sek Slette alle brukere: 146.704771 - 20.449 i sek Slette alle Org: 3.186115 - 23.854 i sek	StoyProsent: 0 - StoyKjoreTid: 0 Last opp Org (76): 6.679544 - 11.378 i sek Last opp Brukere (3000): 130.772593 - 22.941 i sek Sok etter CN: 8.724189 - 114.624 i sek Sok etter Atributter: 328.291321 - 3.046 i sek Enring av brukere: 286.831175 - 10.459 i sek Slette alle brukere: 143.42571 - 20.917 i sek Slette alle Org: 3.181541 - 23.888 i sek
StoyProsent: 30 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 3.613609 - 21.032 i sek Last opp Brukere (3000): 161.037118 - 18.629 i sek Sok etter CN: 11.225461 - 89.083 i sek Sok etter Atributter: 121.557363 - 8.227 i sek Enring av brukere: 308.791213 - 9.715 i sek Slette alle brukere: 145.31358 - 20.645 i sek Slette alle Org: 2.401924 - 31.641 i sek	StoyProsent: 30 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 2.710681 - 28.037 i sek Last opp Brukere (3000): 143.782784 - 20.865 i sek Sok etter CN: 8.309862 - 120.339 i sek Sok etter Atributter: 98.045643 - 10.199 i sek Enring av brukere: 345.536518 - 8.682 i sek Slette alle brukere: 175.672767 - 17.077 i sek Slette alle Org: 3.348743 - 22.695 i sek
StoyProsent: 80 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 2.952447 - 25.741 i sek Last opp Brukere (3000): 134.703846 - 22.271 i sek Sok etter CN: 11.272228 - 88.714 i sek Sok etter Atributter: 82.875228 - 12.066 i sek Enring av brukere: 306.814918 - 9.778 i sek Slette alle brukere: 129.597864 - 23.149 i sek Slette alle Org: 4.423369 - 17.181 i sek	StoyProsent: 80 - StoyKjoreTid: SAMTIDIG Last opp Org (76): 2.703115 - 28.116 i sek Last opp Brukere (3000): 161.613586 - 18.563 i sek Sok etter CN: 11.489701 - 87.034 i sek Sok etter Atributter: 131.00256 - 7.633 i sek Enring av brukere: 391.055602 - 7.672 i sek Slette alle brukere: 160.75139 - 18.662 i sek Slette alle Org: 2.954222 - 25.726 i sek

Tabell 6 – Active Directory Senario 2.

Tabell 6 viser resultat etter en AD senario 2 test. Verdiene i tabellen viser typen operasjon, mengden av den gitte operasjon og antall slike operasjoner i sekundet. Denne testen er kjørt en gang på hver av de respektive hypervisorene. I forkant av denne testen er det kjørt en kort oppvarmingsfase på begge hypervisorene som består av en enkel AD test uten noen form for støy.

7 DISKUSJON AV RESULTATER

Grunnet oppgavens natur er resultatet todelt. Den ene delen av resultat er metoden for testing som utviklet. Den andre delen består av resultater gruppen selv produserte ved gjennomføring av test metoden som er utviklet.

7.1 METODEN

Det er fokusert på real life testing for å møte oppdragsgivers krav. Med dette menes testing av hvordan en maskin yter ved kjøring av ett vanlig program. Oppdragsgiver ønsket en metode for å teste forskjellige virtualiseringsservere med belastning som er relevant for deres daglige arbeid.

Det endelige resultatet av metoden er ett sett skript som forutsetter en rekke forskjellige serverapplikasjoner for gjennomføring, samt at de gjennomføres i henhold til testsenarioene.

I kapittel 3 ble det nevnt fire punkter som er kritiske når det skal utvikles en testmetode, repeterbarhet, relevans, tyngde og sammenlignbarhet.

Repeterbarheten av testmetoden er møtt ved at testene enkelt lar seg gjennomføre på nytt

> "*I den empiriske vitenskapen er ingen verdier kjente med uendelig presisjon eller absolutt sikkerhet*." [84]

Dette gjenspeiles også i denne testmetoden. Det er registrert avvik i resultater på to identiske tester, selv om avviket er lite er det fortsatt ett avvik. Det er derfor valgt at alle verdier i testresultatene skal ses på med en feilmargin på +/- 5 %. Eksempelvis en test som har produsert ett tall for MB/s ved opplasting av data på 10. Vil den faktiske verdien strekke seg fra 9,5 – 10,5 MB/s.

Med denne forutsetning på +/- 5 % feilmargin møter metoden kravet om repeterbarhet, ved at testen lar seg gjennomføre på nytt og produserer tilnærmet den samme verdien om forutsetningene er de samme.

Kravet om relevans er møtt gjennom å utføre real life testing av serverapplikasjoner som oppdragsgiver bruker til daglig. Det er forsøkt å replisere de faktiske situasjonene en eventuell virtualiseringsserver hadde blitt brukt til av oppdragsgiver. Dette er gjort gjennom kjøring av testsenarioene som benytter seg av støymaskiner. Ideen her er at støymaskinen skal representere en annen type serverapplikasjon som kjører på hypervisoren samtidig.

Kravet om tyngde er møtt med de forutsetninger gruppen hadde. Det siktes da her til serveren lånt av oppdragsgiver som testene er utført og tilpasset til. Dersom testene skulle utføres på en annen server hardware som er kraftigere eller svakere, kan dette føre til at resultatene ikke viser kritiske forskjeller i ytelsen, og resultat ikke er gyldig. For å jobbe seg rundt dette problemet må variabler endres i skriptene for og eventuelt øke eller redusere belastningen i testene. Men det antas at testmetoden vil møte kravet om tyngde på de fleste typer server hardware.

Sammenlignbarhet er møtt gjennom at alle verdier produsert av testene er tallverdier med stor nøyaktighet. Dette gjør det enkelt å sammenligne verdiene på to forskjellige tester. Det skal også nevnes at feilmarginen på +/- 5 % må medregnes ved sammenligning av verdier dersom det skal trekkes en konklusjon av resultatene. Dette vil si at to forskjellige verdier på henholdsvis 9,7 og 10,1 har overlappende verdier i sine intervaller, og at de da teoretisk sett er like.

Etter egen vurdering møter resultatet av metoden kravene om repeterbarhet, sammenlignbarhet, relevans og tyngde beskrevet i kapittel 3.

Resultatet som helhet er en metode for og real life teste virtuelle maskiner med serverapplikasjoner som web server, ad/ldap kontroller, e-post server, database server og terminalserver. Brukervennlighet av metoden er ikke særlig god med tanke på alt arbeid som kreves med oppsettet av testsenarioer og virtuelle maskiner. Gruppen har hele tiden vært klar over den dårlige brukervennligheten, men har fortsatt valgt en løsning der alle serverapplikasjoner er installert på forskjellige virtuelle maskiner. Alternativet til dette hadde vært og installere alle serverapplikasjoner på en virtuell maskin. Dette ville gjort det mulig å utføre alle tester i en eneste omgang uten noe administrering. Likevel er det valgt ikke å gjøre dette fordi det var ønskelig å utføre testene på isolerte virtuelle maskiner som kun hadde en enkelt serverapplikasjon installert. Dette for å sikre at den virtuelle maskinen var mest mulig lik en maskin som eksister i ett produksjonsmiljø, altså minimalt med installerte applikasjoner utover det absolutt nødvendige for å utføre ønsket jobb. Denne løsning ville også gjort det vanskeligere og spesifikt teste en enkel serverapplikasjon dersom det skulle vært ønskelig.

7.1.2 ALTERNATIVET

Dersom metoden tas i bruk av andre kan forsøksbetingelser i testsenarioene endres etter eget ønske. Variabler som støy prosent og varighet på støy velges selv ved oppsettet av testsenarioet. Videre er det også mulig å installere alle serverapplikasjoner på en og samme virtuelle maskin, og på denne måten redusere mengden administrering som kreves for gjennomføring av alle testtyper.

Det vil også være mulig å skripte kjøringen av alle skript fra metoden og på denne måten teste alle serverapplikasjoner og alle testsenarioer i en og samme kjøring uten noen form for administrering.

Denne fremgangsmåten vil tilføre metoden flere usikkerhets momenter rundt verdiene som produseres. Det kan også føre til at kravet om relevans ikke lenger oppfylles med tanke på oppdraget som er gitt av arbeidsgiver.

7.2 TEST RESULTATER

Resultatene fremvist i kapittel 6.2 er generert ved kjøring av tester på server applikasjonene FTP, e-post, Mysql database, terminal, Active Directorty og open LDAP. Server applikasjonen er testet ved kjøring av testskript kombinert med

testsenario 1 og 2 beskrevet i kapittel 5. Testene er utført, og server hardware lånt av oppdragsgiver.

Målet med denne testen er hovedsakelig å se hvilken av hypervisorene VMware esxi 4.0 og Citrix XenServer 5.5. som yter best, dette ved bruk av testmetoden utarbeidet av gruppen.

Den største overraskelsen i testene er resultater produsert ved bruk av testsenario 1. Gruppen hadde på forhånd trodd at denne testen ville gi testene et svært dårlig resultat ved økning av støyprosent og varigheten ved kjøring av støy. Men testene viser som regel kun en minimal ingen endring i resultatene, foruten om terminal testen. Her er testen treg ved de 3 første kjøringene. Når testen i gjentas uten omstart av hypervisoren holder testen seg på rundt +/- 70 sekunder.

Videre er det observert at testsenario 2 har fører til en klar reduksjon av ytelsen ettersom belastningen på støymaskinen øker. Dette er tilfellet for begge hypervisorene.

Dette kan indikere at å ha flere virtuelle maskiner kjørende samtidig kan påvirke ytelsen til hverandre ved at det er en kamp om ressurssene tilgjenglig på den virtuelle maskinen.

Forskjeller mellom VMware Esxi og Citric XenServer kommer klart frem i alle tester og testsenarioer. Og det er XenServer som kommer best ut av dette sammenlignet med VMware. XenServer yter best på samtlige tester med unntak av FTP opplasningenstest og open LDAP CN søketest. Dette er allikevel resultater generert med vår testingsserver og sin sammensatte hardware komponenter. Resultatene kan derfor på ingen måte konkludere at noen av hypervisorene er bedre enn den andre i alle forhold. VMware ESXi 4 er blant annet ikke listet som offentlig kompatibel vår testingserver [83].

I artikkelen ”Xen and the art of virtulazation” kommer det frem av deres tester at ytelsen til vmware og xen ligger tett oppunder native ytelse. Dette er riktignok tester utført på VMware workstation og XeonLinux .

Resultater fremvist i denne rapporten viser at både VMware og Xen ligger langt bak native ytelse. Denne forskjellen kan komme av at de fleste tester utført i denne rapporten er tester som krever mye I/O men i xen artikkelen er det mer fokus på prosessor rettede tester.

8 KONKLUSJON

Oppgavebeskrivelsen gitt av it tjeneste ved hig var å utvikle en metode for og systematisk stressteste og benchmarke forskjellige serverapplikasjoner på gjesteoperativsystemer kjørende på en virtualiserings server. Dette var spesielt med tank på virtualiseringsservere av typen VMware Esxi 4.0 og Citrix XenServer 5.5.

Oppgaven gitt av oppdragsgiver begrenser seg til uvikling av en metode for testing av serverapplikasjoner på en virtualiseringsserver, vi tolker dette som oppdragsgiver kun ønsker seg verktøyet for selv å kunne gjennomføre testen.

Vi ønsket også å benytte denne metoden til å gjennomføre en test av Esxi 4.0 og XenServer 5.5 for å trekke en slutning ut fra egenproduserte resultater.

Vi har gjennom oppgaven utviklet en metode for testing av serverapplikasjoner på virtuelle servere. Dette gjøres ved bruk av perl skrip. Skriptene benytter seg enten av ferdig produserte benchmark verktøy som Apache Bench og Mysql Bench eller selv produserte benchmarker der en gitt jobb skriptes og måles. Etter samtaler med oppdragsgiver er det utviklet en rekke testsenarioer. Disse testsenarioene har som mål å fremtvinge situasjoner der oppdragsgiver tidligere har hatt problemer med ytelsen til virtuelle maskiner.

I kapittel 3 av rapporten er det beskrevet fire kriterier for utvikling av testmetoder. Vi føler vi oppfyller disse kriteriene.

Tester gjennomført viser at den samme testen ikke leverer det eksakt samme resultatet men det er svært nært. For sikkerhets skyld har gruppen valgt å sette en feilmargin på 5 % ved sammenligning av resultater. Dette vil si at verdien produsert av en test befinner seg i ett intervall som strekker seg fra +/- 5 % av målt verdi. Dersom to målte verdier har overlappende intervaller kan det konkluderes med at de er like.

Ved gruppens egen gjennomføring av testmetoden for testing av VMware Esxi 4.0 v.s XenServer 5.5 kom det fram at XenServer 5.5 yter bedre enn VMware Esxi 4.0 ved alle tester med unntak av FTP opplasting og Cn søk i open LDAP. Det kom også frem at ytelsen til begge hypervisorene lå langt under native ytelse. VMware Esxi 4 hadde svært dårlig skrivehastighet i forhold til Xen Server. Den store forsjellen i resultatene tror vi skyldes test serverens støtte for VMware Esxi. Det kan også argumenters med at testene ikke forteller hele sannheten fordi de ble utført på en standard installasjon av hypervisorene uten noen form for tilpassingen og optimalisering. Dersom dette hadde vært gjort kunne kanskje resultatet blitt noe annet.

For å oppsummere har gruppen utviklet en metode for og teste serverapplikasjoner på virtualiseringsservere. Gruppen har også gjennomført en test ved bruk av metoden for å avgjøre hvilke av hypervisorene VMware Esxi 4.0 og Citrix XenServer 5.5 som yter best.

Dette tatt i betraktning kan gruppen konkludere med at metoden som helhet kan benyttes til å avgjøre hvilken hypervisor som yter best med typen belastning det er

test med. Det kan også konkluderes med at XenServer 5.5 yter bedre en Esxi 4.0 med gruppens testmetode på maskinvaren testene ble utført på.

8.1 HVA HAR VI LÆRT I LØPET AV PROSJEKTET

Det vi sitter igjen med etter prosjektet er en bedre forståelse av virtualiseringsteknologien generelt. I løp av prosjektet har vi også fått bedre kunnskap om skripting og da spesielt perl skript. Vi har også tilegnet oss en del kunnskaper som ikke kommer frem i denne rapporten. Herunder faller emner som å utføre mer daglig arbeid i linux systemer og kunnskap om en rekke andre applikasjoner for overvåking av system ressurser og administrering av XenServer og Esxi.

Vi har også tilegnet oss kunnskaper og erfaring om hvordan det er å jobbe med større prosjekter og at planlegging i forkant av dette er spesielt viktig for utnytting av tiden man har til rådighet.

8.2 PROBLEMER I PROSJEKTET

Et av problemene vi har hatt under dett prosjektet er at oppgaven var veldig åpen. Det var i start uklart for oss hva slags metode oppdragsgiver ønsket. Dette førte til at vi brukte mye tid og energi på å sette seg inn i teknologier som ikke er benyttet i oppgaven. Men mye av tiden brukt på dette har senere dannet grunnlaget for flere valg gruppen har tatt med tanke på løsningen.

Et annet problem gruppen hadde var å bestemme seg for om hvorvidt vi skulle gjennomføre testene ved bruk av testmetoden som var utviklet, og tilføye dette som en del av oppgaven.

Dette førte også til at det lenge var usikkerheter rundt hvorvidt dette var en utviklingsoppgave eller en analyseoppgave, med tanke på rapportskrivingen og bruk av utviklingsmodell.

8.3 HVA KUNNE VÆRT GJORT ANNERLEDES

Etter å ha gjennomført dette prosjektet er det en del ting vi ønsker vi hadde gjort annerledes. Dette er knyttet til punktene over. Fra starten av prosjektet burde vi ha kartlagt oppgaven bedre. Dette hadde ført til at vi hadde brukt langt mindre tid i starten av prosjektet til å sette oss inn i feil teknologier. Dette ville igjen gitt oss bedre tid til å utføre testene selv slik at vi kunne hatt flere gjennomføringer og konkludert med større sikkerhet.

Vi hadde også vært tjent med kun å fokusere på utvikling av testmetoden. Dette vil gitt bedre tid til testing av metoden og skriving av rapport. Vi kunne da forsøkt å forenkle metoden til noe som krever mindre administrasjon.

8.4 FORSLAG TIL VIDERE ARBEID

Vi føler dette er en oppgave som har ett stort potensial når det gjelder videre arbeid. Metoden kan videreutvikles med å tilføre nye testsenarioer eller tilføre metoden andre måter å teste serverapplikasjoner på. Det er også mulig å tilføre testing av andre typer serverapplikasjoner.

Det kunne også arbeides videre med å gjøre metoden brukervennlig å redusere mengden administrering som kreves for gjennomføring av testene.

Det kunne vært intressangt å byttet ut støyscriptet med et belastet serverapplikasjon og hatt en lignende regulerbar belastnings mulighet.

Arbeid med å teste XenServer mot Vmware esxi kan også fortsette. Vi føler vi ikke fikk gjennomført på langt nær så mange tester som vi opprinnelig ønsker. Med flere tester har man mer data å basere en eventuell konklusjon på. Testene bør desutten testes på en maskinvare med dokumentert støtte for begge hyptervisorplattformene.

8.5 VURDERING AV GRUPPEARBEID

Begge gruppens medlemmer kjente hverandre godt fra før, og har tidligere arbeidet sammen på mindre prosjekter. Det tidligere arbeidet har ført til ett godt samarbeid som har blitt utnyttet under dette prosjektet. Gruppemedlemmene har hatt daglig dialog igjennom avsatt tid for prosjektet igjennom hele perioden. Gruppen har for hver uke planlagt videre arbeids oppgaver og fordeling av oppgavene. Uenigheter for avgjørelser har blitt videre diskutert med veileder for endelig avgjørelse.

Siden det bare er to personer i gruppen har informasjonsflyt ikke har vært noe problem under prosjektet, grunnet god bruk av SVN, grupperom og skype som kommunikasjons kanaler.

9 LITTERATURLISTE

[1]: It avdeling ved høgskolen i Gjøvik. 17.5.2010. Tilgjenglig fra:
http://hig.no/it_tjenesten

[2]: Wikipedia. Virtulaserings plattform. 17.5.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Hardware_virtualization

[3]: Wikipedia. Mac OS . 10.5.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Mac_OS

[4]: Microsoft windows. Generell informasjon om windows operativsystemet. 5.5.2010. Tilgjenglig fra:http://www.microsoft.com/windows/

[5]: Linux. Generell informasjons om linux operativsystemet. 5.5.2010. Tilgjenglig fra:
http://www.linux.org/info/

[6]:VMware. Hjemmesiden for Vmware. 5.5.2010. Tilgjenglig fra:
http://www.vmware.com/

[7]:VMware. Esx og Esxi. 5.5.2010. Tilgjenglig fra:
http://www.vmware.com/products/esx/

[8]: VMware communities. WMware esxi 4.0 overview. 17.5.2010. Tilgjenglig fra:
http://communities.vmware.com/community/vmtn/vsphere/esxi

[9]: Citrix systems. Citrix XenServer. 17.5.2010. Tilgjenglig fra:
http://citrix.com/English/ps2/products/feature.asp?contentID=1686939

[10]: Wikipedia. Benchmarking(computing). 17.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Benchmark_%28computing%29

[11]:VMware inc. Vmware server overview. 17.05.2010. Tilgjenglig fra:
http://www.vmware.com/products/server/

[12]: Xen. Hjemmeside for Xen. 17.05.2010. Tilgjenglig fra:
http://www.xen.org/

[13]: CentOS. CentOS information. 17.05.2010. Tilgjenglig fra:
http://www.centos.org/modules/tinycontent/index.php?id=2

[14]:VMware inc. WMware Esxi 3.5. 15.05.2010. Tilgjenglig fra:
http://www.vmware.com/files/pdf/vmware_esxi_datasheet.pdf

[15]:Debian. Hjemmeside for Debian. 15.05.2010. Tilgjenglig fra:
http://www.debian.org/intro/about

[16]: Hans Nordhaug. Meget kort om skripting. 02.04.2009, 15.05.2010. Tilgjenglig fra;
http://kursinfo.himolde.no/in-kurs/IBE115/2009/forelesninger/13-skripting.pdf

[17]: Wikipedia. Scrum. 17.05.2010. Tilgjenglig fra:
http://no.wikipedia.org/wiki/Scrum

[18]: VMware inc. History of virtualization. 20.04.2010. Tilgjenglig fra:
http://www.vmware.com/virtualization/history.html

[19]:Wikipedia. Stormaskin. 10.05.2010. Tilgjenglig fra:
http://no.wikipedia.org/wiki/Stormaskin

[20]:Wikipedia. Hypervisor. 17.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Hypervisor

[21]:Wikipedia. X86-arkitektur. 17.05.2010. Tilgjenglig fra:
http://no.wikipedia.org/wiki/X86-arkitektur

[22]: WindowsNetworking. The Pros and Cons of running Virtual Server. 17.05.2010. Tilgjenglig fra:
http://www.windowsnetworking.com/articles_tutorials/Pros-Cons-Virtual-Server.html

[23]: Gerald J.Popek, Robert P.Goldberg.Formal Requirements for Virtualizable Third Generation Architectures. 10.03.2010. Tilgjenglig fra;
http://www-users.itlabs.umn.edu/classes/Fall-2009/csci8980-virtual/papers/popek-virt-reqmts.pdf

[24, 25]: Virtualizationreview. Type 1 and Type 2 Hypervisors Explained. 14.05.2010. Tilgjenglig fra;
http://virtualizationreview.com/blogs/everyday-virtualization/2009/06/type-1-and-type-2-hypervisors-explained.aspx

[26, 27, 28, 29]: VMware inc. Understanding Full Virtualization, Paravirtualization, and HardwareAssist. 20.04.2010. Tilgjenglig fra:
http://www.vmware.com/files/pdf/VMware_paravirtualization.pdf

[30,31]: Wikipedia.Paravirtualisation (2 avsnitt). 10.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Paravirtualization

[32]:Wikipedia. Wmware Server(VMware Gsx Server). 15.04.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/VMware_Server

[33]:Open soruce initiative. Hjemmeside for open source initiative. 17.05.2010. Tilgjenglig fra:
http://www.opensource.org/

[34]: Xdnet. Red Hat releases Xen-enabled Linux beta. 11.09.2006, 13.05.2010. Tilgjenglig fra:
http://www.zdnet.com/news/red-hat-releases-xen-enabled-linux-beta/149493

[35]:Wikipedia. IBM systems/370. 10.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/IBM_System/370

[36, 61]:Intel. Virtulization. 15.05.2010. Tilgjenglig fra:
http://www.intel.com/technology/virtualization/

[37]:AMD. AMD virtulazation. 15.05.2010. Tilgjenglig fra:
http://sites.amd.com/us/business/it-solutions/virtualization/Pages/virtualization.aspx

[38]: Searchvmware. VMotion. 04.05.2010. Tilgjenglig fra:

http://searchvmware.techtarget.com/tip/0,289483,sid179_gci1279864,00.html

[39]: Citrix. New Citrix XenServer Release Makes Enterprise-class, Cloud-Proven Virtulization Free for Evryone. 12.05.2010. Tilgjenglig fra:
http://www.citrix.com/English/ne/news/news.asp?newsID=1687130

[40]: Creative Data Concepts.The Hidden Cost of Computer downtime. 12.05.2010.
Tilgjenglig fra:
http://www.creativedata.net/index.cfm?webid=207

[41]: Mark F.Mergen, Volkmar Uhlig, Orran Krieger, Jimi Xenidis. Virtualization for High-Performance Computing. 10.05.2010. Tilgjenglig fra:
http://www.research.ibm.com/K42/papers/osr06-virt.pdf

[42]:Wikipedia. Pentium 4. 17.05.2010. Tilgjenglig fra
http://en.wikipedia.org/wiki/Pentium_4

[43]: CPU world. AMD Athlon XP processor. 17.05.2010. Tilgjenglig fra:
http://www.cpu-world.com/CPUs/K7/TYPE-Athlon%20XP.html

[44]: Anandtech. Real world VirtulalizationBenchmarking. 14.04.2010. Tilgjenglig fra:
http://it.anandtech.com/show/2770

[45]: Computerzen. Virtual machine cpu performance. 14.04.2010. Tilgjenglig fra:
http://www.hanselman.com/blog/VirtualMachineCPUPerformance.aspx

[46]:Wikipedia. Black box testing. 15.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Black_box_testing

[47]: Wikipedia. White box testing. 15.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/White_box_testing

[48]: Wikipedia. Floating point. 15.05.2010 Tilgjenglig fra:
http://en.wikipedia.org/wiki/Floating_point

[49]: Techwarelabs. Synthetic VS Resl World benchmarking. 17.05.2010. Tilgjenglig fra:
http://www.techwarelabs.com/articles/editorials/real-vs-synthetic

[50]: Wikipedia. Whetstone. 16.05.2010 Tilgjenglig fra:
http://en.wikipedia.org/wiki/Whetstone_%28benchmark%29

[51]: Wikipedia. Dhrystone. 16.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/Dhrystone

[52]: SPEC. Hjemmesider til SPEC. 17.05.2010. Tilgjenglig fra:
http://www.spec.org/spec/

[53, 56]: Keith Adams, Ole Agesen. A comparison of software and Hardware Techniques for x86 virtualization. 13.04.2010. Tilgjenglig fra:
http://www.vmware.com/pdf/asplos235_adams.pdf

[54]: Paul Barham_, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris,
Alex Ho, Rolf Neugebauer, Ian Pratt, Andrew Warfield. Xen and the Art of Virtualization.
17.05.2010. Tilgjenglig fra:

http://www.cl.cam.ac.uk/research/srg/netos/papers/2003-xensosp.pdf

[55]: SPEC. Benchmarkings. 17.05.2010. Tilgjenglig fra:
http://www.spec.org/benchmarks.html#virtual

[57]: Wikipedia. WMmark. 15.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/VMmark

[58]: Sysbench. SysBench manual. 15.05.2010. Tilgjenglig fra:
http://sysbench.sourceforge.net/docs/

[59]: Hewlett-packard. HP ProLiant DL320 G5 Server series - specifications and warranty. 17.05.2010. Tilgjenglig fra:
http://h10010.www1.hp.com/wwpc/ca/en/sm/WF06a/15351-15351-3328412-241475-241475-3201178.html

[60]:Dell. Dell™ OptiPlex™ GX260 Systems User's Guide. 17.05.2010. Tilgjenglig fra:
http://support.dell.com/support/edocs/systems/opgx260/en/ug/specs.htm#1110653

[62]:Microsoft corporation. Windows server 2008 R2. 17.05.2010. Tilgjenglig fra:
http://www.microsoft.com/windowsserver2008/en/us/default.aspx

[63]:Apache. Apache http server benchmarking tool. 15.05.2010. Tilgjenglig fra:
http://httpd.apache.org/docs/2.0/programs/ab.html

[64]:Microsoft corporation. Active directory. 17.05.2010. Tilgjenglig fra:
http://www.microsoft.com/windowsserver2008/en/us/ad-main.aspx

[65]:Wikipedia.Open ldap. 17.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/OpenLDAP

[66]:Wikipedia. FTP. 17.05.2010. Tilgjenglig fra:
http://no.wikipedia.org/wiki/FTP

[67]:Mysql. About MySql. 15.05.2010. Tilgjenglig fra:
http://www.mysql.com/about/

[68]: SearchVMware. VMware Virtual SMP. 15.05.2010. Tilgjenglig fra:
http://searchvmware.techtarget.com/sDefinition/0,,sid179_gci1307757,00.html
[69]: Wikipedia. VMware_VMFS. 15.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/VMware_VMFS

[70]:Vmware inc. Features. 12.05.2010. Tilgjenglig fra:
http://www.vmware.com/products/drs/features.html

[71]: VMware inc. VMware Esxi Features. 12.05.2010. Tilgjenglig fra:
http://www.vmware.com/products/esxi/features.html

[72]:Enterprice desktop. What is a virtul machine snapshot?. 17.05.2010. Tilgjenglig fra:
http://searchenterprisedesktop.techtarget.com/sDefinition/0,,sid192_gci1323483,00.html

[73]: Wikipedia. System console. 16.05.2010. Tilgjenglig fra:
http://en.wikipedia.org/wiki/System_console

[74]: Linux bt examples. When netcat act as telnet client, it becomes better. 15.05.2010. Tilgjenglig fra: http://linux.byexamples.com/archives/258/when-netcat-act-as-telnet-client-it-becomes-better/

[76]: Cpan. Net::LDAP - Lightweight Directory Access Protocol. 17.05.2010. Tilgjenglig fra: http://search.cpan.org/~gbarr/perl-ldap-0.4001/lib/Net/LDAP.pod

[77]: Wikipedia. LDAP Data Interchange Format.10.05.2010. Tilgjenglig fra: http://en.wikipedia.org/wiki/LDAP_Data_Interchange_Format

[78]:Wikipedia. Tynnklient.06.05.2010. Tilgjenglig fra: http://no.wikipedia.org/wiki/Tynnklient

[79]:Microsoft developer Network. Remote desktop protocol. 12.04.2010. Tilgjenglig fra: http://msdn.microsoft.com/en-us/library/aa383015.aspx

[80]:Linuc die.net. Rdesktop. 15.05.2010. Tilgjanglig fra: http://www.rdesktop.org/

[81]: benchmark scriptene til mysql http://dev.mysql.com/doc/refman/5.0/en/mysql-benchmarks.html

[82]:Wikipedia. Cache. 14.04.2010. Tilgjenglig fra: http://en.wikipedia.org/wiki/Cache

[83]: VMware ESXi 4 er blant annet ikke listet som offentlig kompatibel med vår testing server HP ProLiant 320 G5 http://h71028.www7.hp.com/enterprise/us/en/servers/4x-servers.html

[84]: Wikipedia. Konfidensintervall, sistat andre avsnitt. 19.05.2010. Tilgjenglig fra: http://no.wikipedia.org/wiki/Konfidensintervall

10 VEDLEGG

A.ORDLISTE

"real word"	Type benchmarking der det benyttes ”vanlige” programmer for testing
"Rett på metalet"	Hypervisor kjører med direkte kontroll over hardware.
Active Directory	Katalogtjeneste for Windows servere. Administrering av brukere, datamakiner og Windows policy.
Apache	Webtjener i åpen kildekode
Bash	Script språk for shell i Linux
Benchmark	En eller et sett med tester for ytelses teste hardware
CentOS	Gratis Opprativsystem basert på Red Hat Enterprise Linux
CSS	Cascading Style Sheets. Bestemmer utsender til elementer i en webside
Dovecot	E-post server for Linux med blant annet Pop3 og IMAP støtte
FTP	File Transfer Protocol, uavhengig filoverførings protokoll for kopiering av filer over nettverk
Hypervisor	Enkelt datamaskin fremstår som flere separate maskiner virtuelt
IMAP	Internet Message Access Protocol. Behandlig av e-post meldinger, kataloger og melding status forblir på server.
ISO Fil	Arkivfil med fil innhold til CD/DVD plate
LDAP	Lightweight Directory Access Protocoll. Protokoll til oppslag i en katalogtjeneste på en server
Microsoft Exchange	E-post server fra Microsoft
moduler / modul (Perl)	Separat kildekode, designet for å utføre en spesefik jobb med deler av hoved programmets elementer.
MySQL	SQL basert Database server lisensiert under GPL
Perl	Script språk som kjører over flere oprativsystem plattformer
Postfix	Epost overførings agent
RDP	Remote Desktop Protocol. En protokol laget av Microsoft for fjernstyring
Script	Et sett med kommandoer som tolkes og utføres av et separat program. Kildekoden kompileres aldri helt til prosessorinstrukser.
Støyscript	Prosjektes regulerbare cpu belastnings script
Terminal Klient marskin	Maskin som viser tekst eller grafisk grensesnitt som eksikveres på en ekstern
Virtualisering ressursene	Kjøring av programvare i et miljø adskilt fra den underliggende maskinvaren
VM /Gjest Oprativsystem	Et oprativsystem som kjører med et underliggende hypervisor som behandler hardware instruksjonene til oprativsystemet.
VMM	"Virtual Machine Monitor”. I virtualisering også kalt hypervisor
Vmware ESX	Virtuel server familie av typen hypervisor 1
Windows 2008 Server	Server basert Oprativsystem fra Microsoft
x86	Instruksjonssett brukt i en menge prosessorer
Xen Server 5.5	Virtuel server av typen hypervisor 1

B.MØTEREFERAT

MØTEREFERAT 1

Tidspunkt: 22.01.2010

Sted: It-tjenesten

Møte Tid: 3 timer

Neste møte: (Fast møte tidspunkt settes i Uke 4)

Tilstede:
Kristoffer Elde
Øyvind Haugedal
Jon Langseth

Ikke tilstede:
Erik Hjelmås

Agenda:
Spørsmål og fordypelse til oppgaven
Tips om gjennomføring av Bachlor oppgaven
Fast møtetidspunkt
Underskriving av prosjekt avtale

Spørsmål og fordypelse til oppgaven:
Fikk informasjon om hvordan it-tjenesten bruker virtualisering i dag og hvilken mål de har for å forbedre løsningen med vår oppgave. Vi fikk også kartlagt hvilken server os og applikasjoner som ble kjørt undervirtualisering og antall servere de hadde i maskinparken samt klientmaskiner.

Tips om gjennomføring av Bachlor oppgaven:
Vi framla våre ider og vår forståelse av oppgaven og hvordan tanker vi hadde gjort for å løse den. Vi ble oppfordret til flere programvarer som vi burde gjøre og kjente med å bruke i ferdig løsningen. Arbeidsgiver fortalte at de var ute etter en ferdig test løsning for man på forhånd skulle gjøre så lite klar gjørig som mulig. Det var ønskelig at løsningen skulle opprettes og gjentas på andre virtualiserings plattformer om ønkselig. Man vi helst sett at alt av testing kunne

skje ved et taste trykk men dersom dette skulle være for vanskelig kunne løsningen også bestå av flere test applikasjoner som gjorde de forskjellige testene.

Fast møtetidspunkt:
Vi fikk kartlagt tid hvor Jon Langset var tilgjenglig før å sette et fast møte. Det er på forhånd bestemt at møtene med Arbeidsgiver og veileder skal skje samtidig. Vi ble derfor enige om at gruppen bestemmer et møte tidspunkt så fort de får tilbake melding om ledige tidspunkter fra Erik Hjelmås. Endelig fast møte bestemmes i Uke 4. Arbeidsgiver så helst for seg møte annenhver uke i første omgang siden han ellers er meget tilgjenglig for gruppen ved skolens IT-tjeneste. Arbeidsgiver la ønske om å være tilstede og vitne større lab tester som gruppen utfører senere i prosjektet.

Underskriving av prosjekt avtale:
Arbeidsgiver ga sin underskrift og samtykke til prosjektavtale.

MØTEREFERAT 2

Tidspunkt: 16.02.2010

Sted: a018c(gruppe rom)

Møte Tid: 1 timer

Neste møte: 3 uker

Tilstede:
Kristoffer Elde
Øyvind Haugedal
Jon Langseth

Erik Hjelmås

Ikke tilstede:
Ingen

Agenda:
Legge frem det gruppen har arbeidet med frem til nå.

Statusrapporter

Hva skal gjøres fremover.

Hva forventes fremover.

Spec.org tester.

Annet.

Legge frem det gruppen har arbeidet med frem til nå.
Informerte Jon og Erik om hva vi har jobbet med så langt. Problemet vi hadde med å identifisere prosjektet om hva slags type det var. Hvilken av de rettingingslinjene/modellene vi skal følge når jobber med oppgaven. Om vi skal ha med kravspesifisering eller ikke.

Kom frem til at vi skal følge modellen uten kravspesifisering.

Statusrapporter:

Statusrapporter skal levers hver 4 uke. Skal kort fortelle om hva vi har jobbet med og hvordan vi ligger ann. Trenger ikke følge noen mal(blir ikke noe trekk i karakter om vi ikke følger mal). Størrelsen og innholdet på dette bør ligne den første rapporten vi leverte.

Hva skal gjøres fremover:

Forsette arbeidet med å finne gode verktøy til overvåking. Finne programmer og skript til å generere belastning(skripta til kyrre for sql belastning).

Skrive teste senarioer, her er det viktig å få fram hva vi vil med testen(hva slags belastning, varighet, hva skal det testes på, etc)

Spec.org tester:

Oppdragsgiver føler ikke disse testene er relevante for denne settingen. Sepc.org tester er veldig bechmark orientert og tester bare maks ytelse. Dette er noe han mener ikke er veldig relevant for vår testing.

Annet:

Bruke latex/lyx i stede for word. Word filer fyller opp repositorien veldig fort.

Perfmon kan brukes til å hente ut en del data fra windows maskiner.

SCP er en god løsning for å hente data til en sentralisert maskin.

Loggen kan undras fra det offentlige.

MØTEREFERAT 3

Tidspunkt: 16.03.2010

Sted: a018c(gruppe rom)

Møte Tid: 1 timer

Neste møte: 3 uker

Tilstede:
Kristoffer Elde
Øyvind Haugedal
Jon Langseth

Erik Hjelmås

Ikke tilstede:
Ingen

Agenda:
Gjort til dags dato

Utvalgte test verktøy

Test oppsett og senarioer

Hva skal gjøres fremover

Legge frem det gruppen har arbeidet med frem til nå.
Gruppen føler seg ferdig med arbeid med konfigurering og utforskingen av testverktøyene som skal brukes i den endelige løsningen har begynt scripting av støymaskiner og filserver testen (FTP).

Utvalgte test verktøy

Oppsummert og drøftet hvilken testverktøy som skal brukes til den endelige løsningen. Det mangler fortsatt en belasnings verktøy for OpenLDAP. Gruppen har forsøkt å lage et script i bash med på grunn av at scriptet behandler dataendinger via OpenLDAP sine shell kommandoer går endringene fortregt slik at det ikke kan gi resultater godt nok til den endelige løsningen. Arbeidsgiver foreslo bruk av perl script med OpenLDAP api'er som var lagt raskere og kan hjelpe gruppen med å lage et fungerende script.

Gruppen av lurt på om Postfix og dovecot var ytelsesmessig bra nok til å brukes som epost server testing på linux platformen noe som ble bekreftet av arbeidsgiver at kunne godt brukes i test løsningen.

Test oppsett og senarioer

Gruppen presenterte flere diagram av testoppsett som skal scriptes de kommende 3 ukene. Her kom arbeidsgiver med forslag om senarioer han ønsket skulle være med i den endelige løsningen. Dette har blitt notert ned og blir tatt med i utviklingen av scriptinget.

I de fleste testoppsettene er det viktig med gjeste os som lager støy. Arbeidsgiver og veileder ga et veldig godt tips om å bruke programvare som pakker filer siden de belaster både cpu og io. I tilegg hadde veileder et C program med ”dining filosofer” som vil være svært CPU belastende.

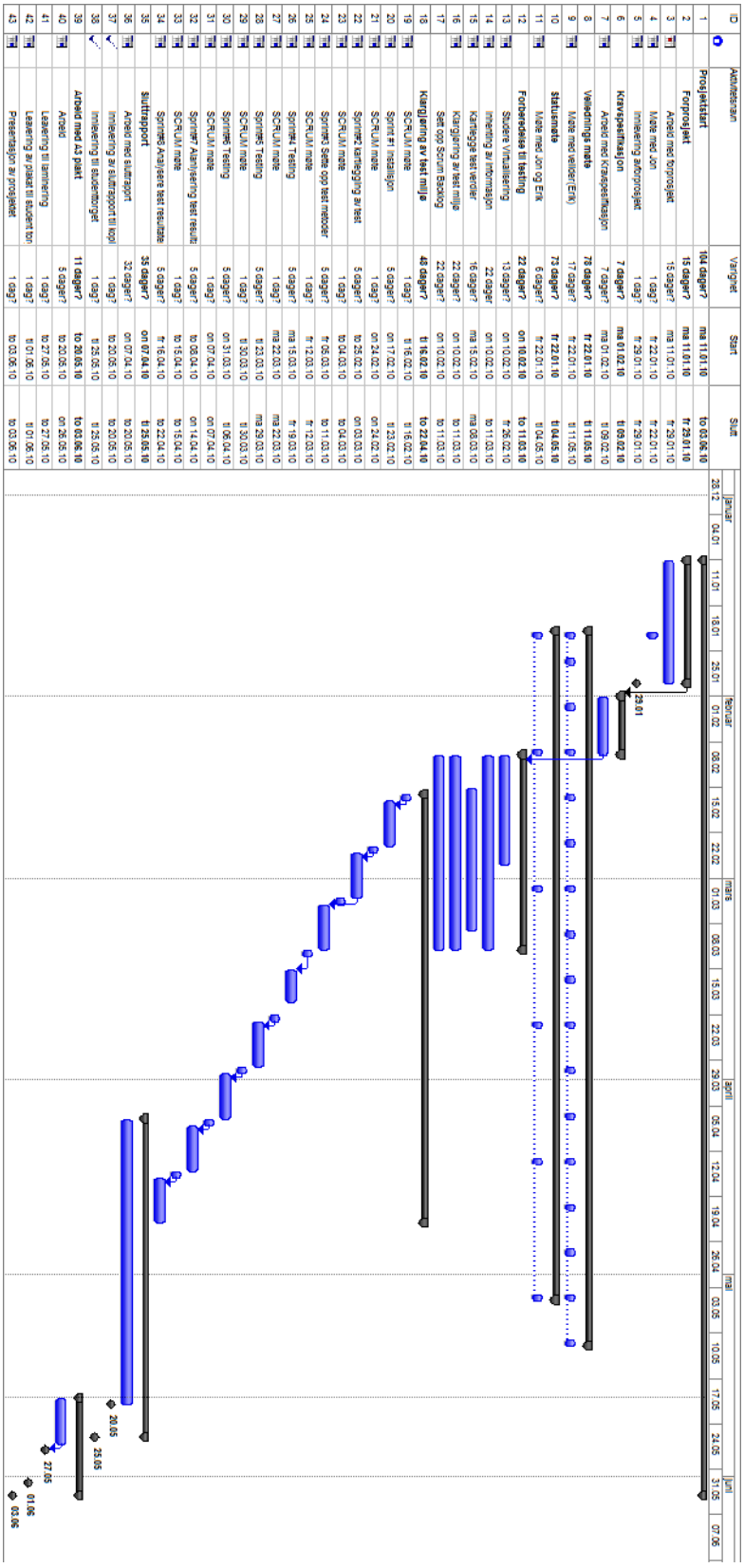
Hva skal gjøres fremover

Etter 17. mars med ”lynnkurs” forelesing angående rapport skriving starter gruppen med minst en dag i uken med rapport skriving. Har fikk vi noen tips av veileder om vi ikke bør bruke fotnoter på mer en hver 3 side og samling av begreps forklaringer bak i rapporten.

Ellers så satser gruppen på å ha alle script laget for å kjøre de forskjellige senarioene klare til 10. april slik at testene kan kjøre etter dette.

C. GANT SKJEMA

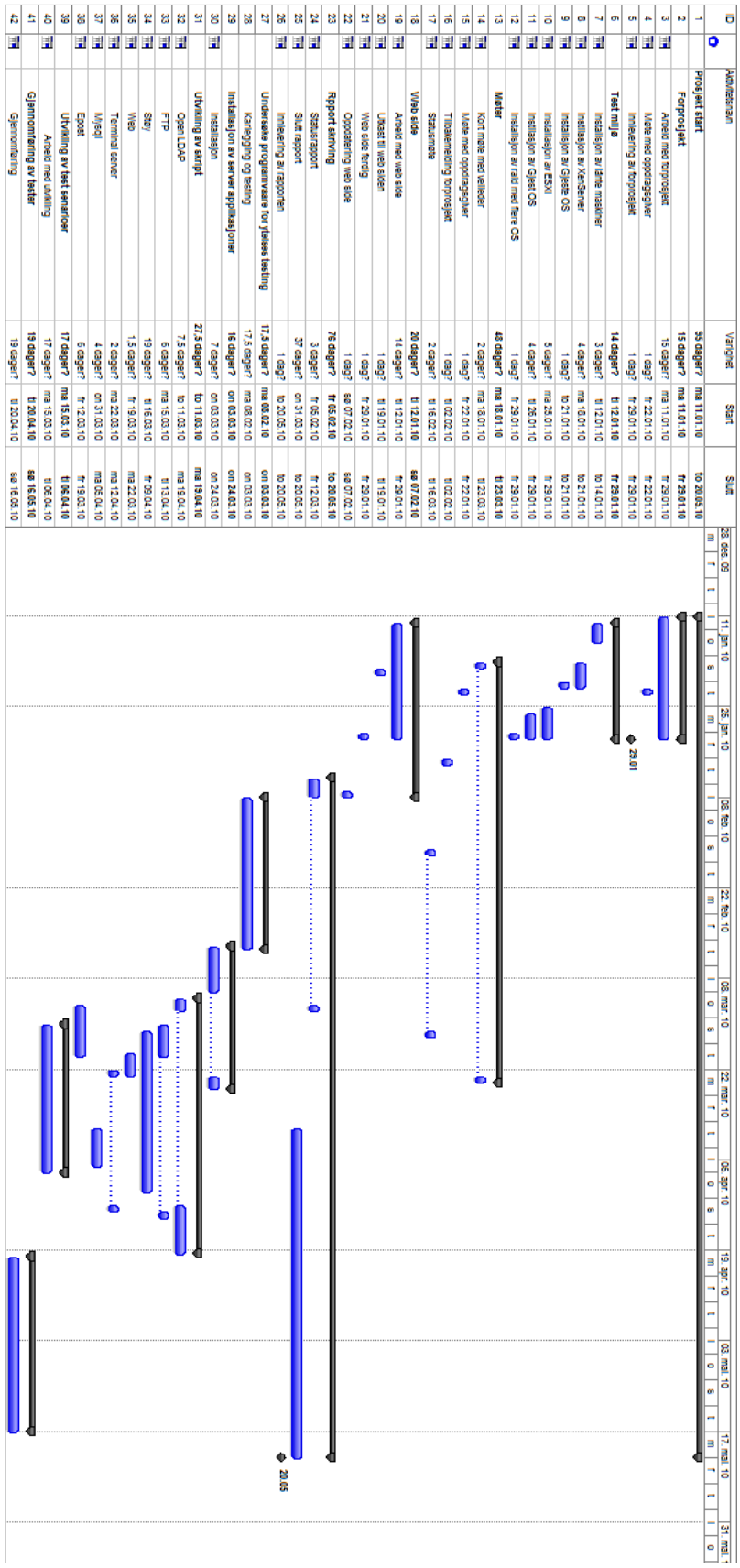
Opprinnelig gant skjema



Figur 22 – Opprinnelig gant skjema

10 Vedlegg

Revidert Gant skjema



Figur 23 – Revidert Gant skjema

D. LOGGBOK

Dato	Hva	Timer
13.01.2010	Har sett litt på tidiligere oppgaver for å se hvordan vi skal utforme forprosjektet. Og funnet en del spørsmål vi skal stille oppdrags giver. Vi har også brukt litt tid for å få opprettet og få SVN til å fungere.	4 Timer - Øyvind 4 Timer - Kristoffer
14.01.2010	Sett på hjemmeside for prosjektet. Diskutert test oppsettet av de Virtualle Produktene. Skrevet forprosjekt rapport. Lest om oppsett og konfigurasjon av VMware EDXi og Xen.	7 Timer - Øyvind 7 Timer - Kristoffer
15.01.2010	Repitering av enkelte systemutviklings modeller. Diskusjon og valg av utvilkings modell. Videre jobbing med forprosjekts rapport. Installering av Windows og Linux os på utlånt grupperom arbeids pc. Lesing av Xen Server 5 installering og konfigurering.	7 Timer - Øyvind 7 Timer - Kristoffer
16.01.2010	Sett på og diskutert forskjellige SU modeller, Begynt å fordele arbeidsoppgaver. Installert OS på lånt PC.	5 Timer - Øyvind 5 Timer - Kristoffer
18.01.2010	Møte med Erik og fått besvart noen spørsmål angående gjennomføring av bachloroppgaven. Videre jobbing med forprosjekt rapport. Utsettelse av møte med Jon, neste møte fredag.	7 Timer - Øyvind 7 Timer - Kristoffer
19.01.2010	Forprosjekt utbedring, retting hverandres skrivedel. Utkast til hjemmeside. Undersøking av aktuelle benchmark løsninger.	7 Timer - Øyvind 7 Timer - Kristoffer
20.01.2010	Forprosjekt utbedring. Installering av Xen Server 5.5 og XenCenter og Gjeste os Cent 5.4, Ubuntu og Windows XP. Sett på Latex dokumentformatering.	7 Timer - Øyvind 7 Timer - Kristoffer
21.01.2010	Laget gant skjema. Gjort avtale med Jon om nytt møte fredag(Morgen). Rettskriving på Forprosjekt rapport.	4 Timer - Øyvind 4 Timer - Kristoffer
22.01.2010	VMware ESXi registering og klar gjøring. Møte med Arbeidsgive	6 Timer - Øyvind 6 Timer - Kristoffer
23.01.2010	Lesing av tidligere leverte Bachlor oppgaver fra Bibsys	4 Timer - Øyvind
24.01.2010	Skrevet møte referat fra Fredag (22.01.2010). Videre arbeid med Forprosjekt rapport.	4 Timer - Øyvind 3 Timer - Kristoffer
25.01.2010	Start med skriving av kravspek. Levering av arbeidsgiver avtale. Installering av VMware ESXi 4 og gjeste Opprativsystem Cent 5 og Windows 2008 Server men Installasjon Ustabil (Mulig raid feil).	7 Timer - Øyvind 7 Timer - Kristoffer
26.01.2010	Tilbake melding på forprosjekt av Erik. Kravspek skriving. Videre jobbing med VMware ESXi 4, noe galt med Raid oppsettet gjør serveren meget treg.	7 Timer - Øyvind 7 Timer - Kristoffer
27.01.2010	Lage prosjektes hjemmeside mer dynamisk med CSS. Rettskrining og setting av utsende mal for forprosjekt rapport.	5 Timer - Øyvind 5 Timer - Kristoffer
28.01.2010	Levering av forprosjekt og fast sette møte med veileder og arbeidsgiver. Utprøving av CPU og minne avlesing i Windows Powersell. hardware data med sysstat i Linux (Skrive til fil).	7 Timer - Øyvind 7 Timer - Kristoffer
29.01.2010	Hentet lisener til Windows 7. Feil søkt feil ved VMware Installering, feil funnet "Begrenset støtte	5 Timer - Øyvind 5 Timer - Kristoffer

	for 64-bit" på HP server. Installering av flere Harddisker på server. Opplasting av webside. Videre jobbing med kravspek.	
01.02.2010	Møte om oppbygning av løsningen. Forsøk på henting av lisenser til Windows Server 2008.	7 Timer - Øyvind 7 Timer - Kristoffer
02.02.2010	Skriving av utkast til kravspek. Forkastet. Endring av rapportmal, ser bort fra vanlig systemutviklings mal. Kort samtale med Erik (veileder) om spec.org.	7 Timer - Øyvind 7 Timer - Kristoffer
04.02.2010	Logging av CPU last (Windows/Linux). Sende log over nettverk. Test av MySQL Benchmark	7 Timer - Øyvind 7 Timer - Kristoffer
05.02.2010	Satt opp Xen Server med RAID oppsett etter 64-bits feilen på server. Satt opp Nagios og Munin. Skrevet status rapport.Sett på perfmon	4 Timer - Øyvind 7 Timer - Kristoffer
07.02.2010	Oppdatert Hjemmeside. Klargjøring for møte tirsdag 9. feb. Rettskriving av status rapport.	4 Timer - Øyvind 2,5 Timer - Kristoffer
08.02.2010	Utarbeidet plan for uken. Laget ett gant skjema for fortløpende arbeid(hvordan vi faktisk jobber) Sett videre på Munin og perfmon.	Øyvind 7 Timer Kristoffer 7,5 Timer
09.02.2010	Planlagt møte med oppdragsgiver og veileder ble avlyst, utarbeidet forslag til product backlog, Set på programvare for å hente data ab WMI.	Kristoffer 6 Timer Øyvind SYK
11.02.2010	Linux: Sett på forskjellige måter å hente ut data fra Sysstat i gitte tidspunkter. Graftegning fra sysstat med Kstat. sysbenc testings kommandoer med CPU og harddisk lesing. Windows: sett på¨å hente ut data gjennom WMI. Sett på verktøy for testing av exchange server.	Kristoffer 7 Timer Øyvind 7 Timer
16.02.2010	Statusmøte med veilder og oppdragsgiver. Skrivie møte referat, lete videre etter overvåkings løsninger til windows.	Øyvind 4 Timer Kristoffer 4 Timer
17.02.2010	Sett på hardware belastnings måling på Windows plattformen. Autogenering og utlasting via script. Sette opp remote perfmon, med logset som kan startes fra komando linjen.	Kristoffer 7 Timer Øyvind 7 Timer
18.02.2010	Uthenting av belastnings data i powershell via WMI objekter. IIS Windows Webserver med PHP Støtte.	Kristoffer 7 Timer Øyvind 7 Timer
22.02.2010	Satt opp httperf med Autobench og gjennomført tester for å finne maks belasting. Forsøkt oppsetting av epost postfix på linux miljø.	Kristoffer 7 Timer Øyvind 7 Timer
08.03.2010	Satt opp epost tjener i Linux med Postfix og dovecot for POP3 og IMAP mottak. Utført epost belastnings forsøk med smtp-source. Gjort forsøk mot active diractory med adtest.	Kristoffer 7 Timer Øyvind 7 Timer
09.03.2010	Arbeid med å sette opp LDAP i linux. Videre arbeid med å få adtest til å fungere. Kort møte med erik,	Kristoffer 7 timer Øyvind 7 timer
10.03.2010	Feilsøking for å få adtest til å fungere, feilen funnet og det skal nå virke.	Kristoffer 7 timer Øyvind 7 timer
11.03.2010	Laget script for OpenLDAP. Repitert bech og Powershell script. Fått adtest til å fungere tilfredsstillende.	Kristoffer 7 timer Øyvind 7 timer
12.03.2010	Startet skriving av statusrapport (hver mnd) til arbeidsgiver og veileder. Satt opp Agenda for neste fellesmøte. Videre scripting i bach. Prøvd å se på API for OpenLDAP.	Kristoffer 4 timer Øyvind 4 timer
15.03.2010	Startet scripting bash for filserver test (FTP) og utformet et dokument for testsenarioer	Kristoffer 7 timer Øyvind 7 timer
16.03.2010	Forsatt skriving av filserver test script og startet script for støymaskin. Har hatt møte med Veileder og oppdragsgiver.	Kristoffer 7 timer Øyvind 7 timer

17.03.2010	Videre jobbing med ftp og støymaskin script. Følgt Frode Haug sitt lynnkurs i rapport skriving for bachlor oppgaven.	Kristoffer 3 timer Øyvind 5 timer
18.03.2010	Videre jobbing med ftp og støymaskiner.	Kristoffer 7 Timer Øyvind 7 Timer
19.03.2010	Videre arbeid med ftp og støymaskiner. Samt startet arbeid med skripting av Apache bensh.	Kristoffer 5 Timer Øyvind 5 Timer
20.03.2010	Videre arbeid med skripting av Ab test og støy skript. Startet belastnings script for Terminal Server. Feilsøking for CentOS 5 på Xen Server 5, lar seg ikke installere.	Kristoffer 4 Timer Øyvind 4 Timer
21.03.2010	Belastnings script for Terminal Server. Feilsøking for CentOS 5 på Xen Server 5, lar seg ikke installere. Prøver med eldre versjoner.	Øyvind 7 Timer Kristoffer 7 Timer
22.03.2010	Scripting av terminalserver test.Scriptiong av Adtest. Skriving av test senarioer.	Kristoffer 7 Timer Øyvind 7 Timer
23.03.2010	Møte med Erik, snakket om rapport skrivingen og fått hjelp med powershell script. Jobbet Videre med Terminal server test og støy script.	Kristoffer 7 Timer Øyvind 7 Timer
24.03.2010	Startet med utvikling av script for E-post belastning. Møtt på feil ved installering av Cent 5.4 på Xen Server løst. Feilsøking ved installering av MS Exchange 2007. Forsetting med utvikling av støy script.	Kristoffer 6 Timer Øyvind 6 Timer
25.03.2010	Utvikling av script for E-post belastning. startet med Linux oppsett først mens installering av MS Echange 2007 gjøres samtidig. MS echange har avhengigheter som skal konfigureres som har gjort at vi enda ikke har en fungerende installasjon av echange.	Kristoffer 7 Timer Øyvind 7 Timer
25.03.2010	Utvikling av script for E-post belastning. Sett på nettverks protocolen til IMAP for å lage egent belastnings skrip. Har ikke fått dovecot og mstone som belastnings verktøy til å fungere på E-post serveren.	Kristoffer 7 Timer Øyvind 7 Timer
26.03.2010	Jobbet med selvlaget E-post belastnings script i perl. Rapport skriving	Kristoffer 4 Timer Øyvind 6 Timer
27.03.2010	Fått scriptet e-post script til å kjøres som flere tåder for simulering av flere pcer. Rapport skriving	Kristoffer 4 Timer Øyvind 6 Timer
28.03.2010	Jobbet med E-post scriptet. Sett på Master oppgaver fra drift ved HiO. Rapport skriving	Kristoffer 4 Timer Øyvind 6 Timer
29.03.2010	Jobbet med E-post scriptet.Rapport skriving	Kristoffer 2 Timer Øyvind 4 Timer
30.03.2010	Jobbet med E-post scriptet.Rapport skriving	Kristoffer 3 Timer Øyvind 6 Timer
31.03.2010	Testet og gjort ferdig E-post teste script. Skriver nå resulater til fil og åpen for Sentralkontroll via nettverk. Startet med MySQL styrings script Rapport skriving	Kristoffer 4 Timer Øyvind 6 Timer
04.04.2010	Scripting av MySQL script. Rapport skriving	Kristoffer 2 Timer Øyvind 6 Timer
05.04.2010	Gjort ferdig MySQL scriptet. Videre jobbing med støyscrip. Laget mulighet for støynivå justering og startet med implementering av harddisk belastning. Rapport skriving	Kristoffer 4 Timer Øyvind 6 Timer
06.04.2010	Jobbet med disk skriving i belastnings scriptet. Ved testing av scriptet på serveren viser det seg at vi har en DMA eller Raid driver feil som gir	Kristoffer 7 Timer Øyvind 7 Timer

	oss lav disk skrive hastighet (3mb/sek). Pratet med Erik om fremgang i prosjektet og problemmer (Exchange, dreping av tråder i perl, Vår IO problem på hovedserveren). Kristoffer har jobbet videre med kapittel 1 i rapporten.	
08.04.2010	Følgt en forelesing av Erik i faget OS om Virtualisering. Fått hjelp av Jon Langseth med treg harddisk på grunn av RAID på serveren. Har også fått hjelp til å lage et script til å beregne cpu last via /proc/status. Jobbet videre med støy scriptet og rapport skrivingen.	Kristoffer 6 Timer Øyvind 6 Timer
09.04.2010	Videre jobbing med støy script. Kan regulere cpu last dynamisk og fjernstyres. Harddisk skriving også lagt til. Rapport skriving.	Kristoffer 7 Timer Øyvind 7 Timer
11.04.2010	Laget Terminal Klient belastnings script. Klient på CentOS og Terminal Server på Windows 2008 Server.	Kristoffer 0 Timer Øyvind 7 Timer
12.04.2010	Gjort ferdig Terminal Klient belastnings script. Startet med apache script. Videre rapporten. Levert utkast til Erik for vurdering.	Kristoffer 7 Timer Øyvind 7 Timer
13.04.2010	Konvertert ftp scriptet til perl fra bash. Laget forbedringer med tids angivelser. Tilbakemeldinger og lite møte med Erik anngående kapitler som er skrevet til rapporten	Kristoffer 7 Timer Øyvind 7 Timer
14.04.2010	Startet med LDAP script og vidrere skriving av rapport.	Kristoffer 7 Timer Øyvind 7 Timer
15.04.2010	Jobbet med Script for ADtest og få en overskikt for belastningene som gjennomførest i programmet. Gjort tester med Net::Ldap biblioteket i Perl mot ADtest for å se om det er et bedre Alternativ. Jobbet videre med skriving av rapporten.	Kristoffer 7 Timer Øyvind 7 Timer
16.04.2010	Pratet med Erik og Jon om hvor blackbox test kan være i prosjektet for avgjøring av prioritering av hvor stor del enkelte tester skal dekkes i rapporten. Har også Pratet med arbeidsgiver Jon om OpenLdap scriptet og fått krav til hvordan han vil ha katalog strukturen generert. Diskutereing hva som er viktig å teste etter at data er lagt inn er også diskutert. Skriving av OpenLDAP skriptet har startet. Det skrives også videre på rapporten.	Kristoffer 7 Timer Øyvind 7 Timer
17.04.2010	Jobbet med OpenLDAP Script.	Kristoffer 0 Timer Øyvind 7 Timer
18.04.2010	Gjort ferdig OpenLDAP Script. Noe skriving på rapporten.	Kristoffer 3 Timer Øyvind 5 Timer
19.04.2010	Gjør klart for kjøring av script på Virtualiserings serveren. Noe små endringer og testing var nødvendig når støy scriptet kjørte native. Skriving på rapporten.	Kristoffer 7 Timer Øyvind 7 Timer
20.04.2010	Gjør klart for kjøring av script på Virtualiserings serveren. Fått hjelp av Jon til å stabilisere støy scriptet slik at den lettere kan finne belastings % av cpuen. Videre skriving av rapport	Kristoffer 7 Timer Øyvind 7 Timer
21.04.2010	Startet¨å lage en senario styrings script for test og støy script. Videre	Kristoffer 7 Timer Øyvind 7 Timer

	skriving av rapport. _____	
22.04.2010	Tilpasset OpenLDAP test scriptet styrings scriptet for OpenLDAP og støy kjøring. Videre skriving av rapport.	Kristoffer 7 Timer Øyvind 7 Timer
23.04.2010	Startet styringscript for FTP testen og støy scriptet. Videre skriving av rapport.	Kristoffer 7 Timer Øyvind 7 Timer
24.04.2010	Skriving av rapport. Kombinere Støyscript og epost test.	Kristoffer 4,5 Timer Øyvind 7 Timer
25.04.2010	Skriving av rapport	Kristoffer 4 Timer
26.04.2010	Skriving av rapport. Kombinere Støyscript og terminal server test.	Kristoffer 7 Timer Øyvind 7 Timer
27.04.2010	Skriving av rapport. Kombinere Støyscript med MySQL og Apache Bench script.	Kristoffer 7 Timer Øyvind 7 Timer
28.04.2010	Skriving av rapport. Gjort endringer i OpenLDAP scriptet slik at det også kan fungere på Microsoft AD. Har også gjort endringer som gjor at scriptet følger utsende til de andre scriptene. Klargjøring av Terminal test og Microsoft AD test på nativ.	Kristoffer 6 Timer Øyvind 6 Timer
29.04.2010	Skriving av rapport. Klargjøre AD test og OpenLDAP test på serveren.	Kristoffer 7 Timer Øyvind 7 Timer
30.04.2010	Skriving av rapport. Kjøring av Terminal Test på Nativ, XenServer og Vmware ESXi. På grunn av meget dårlig resultater på VMware server ble det satt i gang feilsøking. Ny bios og VMware ESXi modul så ut til å gi noe bedre resultater.	Kristoffer 7 Timer Øyvind 7 Timer
01.05.2010	Skriving av rapport. Kjøring av tester.	Kristoffer 7 Timer Øyvind 7 Timer
02.05.2010	Skriving av rapport. Kjøring av tester	Kristoffer 7 Timer Øyvind 2 Timer
03.05.2010	Skriving av rapport. Kjøring av tester	Kristoffer 7 Timer Øyvind 2 Timer
04-09.05.2010	Overvåking av testscriptene. Igang setting av kjøring av nytt script etter som en og en blir ferdig.Skriving av rapport	Kristoffer 42 timer Øyvind 42 timer
10-15.05.2010	Skriving av rapport. overvåking av test script.	Kristoffer 59 timer Øyvind 59 timer
16-17.05.2010	Skriving av rapport.	Kristoffer 22 Timer Øyvind 22 Timer
13.01-20.05.2010	Bachelor oppgave:	Kristoffer 606,5 Timer Øyvind 666 Timer Totalt: 1272,5 Timer

Tabell 7 – Arbeids logg

E. OPPSETT AV TEST MASKINER OG TEST MILJØ

VMWARE ESXI 4.0

Installasjon av VMware Esxi 4.0 krever en installasjons CD. Installasjons filen er tilgjenglig på VMwares hjemmesider[3] som en .iso fil som er gratis å laste ned men krever registrering. Etter registrering får man tilsendt e-post med serial nøkkel til VMware Esxi. Denne har man 60 dager på å registrere etter produktet er installert.

Etter nedlastning av installsjons filen brennes den til CD som videre brukes til installasjon av Vmware Esxi på serveren.

Ved oppstart av installasjonen blir man presentert med en boot menu. Der valget står mellom å boot fra lokal disk på maskin eller kjøre Esxi installer. Det velges her ” Esxi instllaer”.

Man blir så pressenter med ett valg om man vil installere eller reparere. Det skal velges installasjon og gå videre. Under installasjonen blir det presentert en fremgangs linje som indikerer hvor lang installasjonen har kommet. Dette tar ikke mer enn 5 minutter.

Etter endt installering blir det presentert ett skjermbilde som forteller at installasjonen av vellykket og installasjonen vil operere i evaluerings modus i 60 dager eller frem til en serial nøkkel er registret på installasjon.

Det informeres også om at det må benyttes vSphere Client eller Direct Console User Interface for å administrere serveren. Og at en omstart må foretas før serveren kan tas i bruk.

Etter omstart blir man pressenter med dette skjermbildet. Ved trykking av ”F2” kommer man inn i en enkelt konfigurasjons meny der man har muligheten til å sette passord, brukernavn og netverks konfigurasjoner.

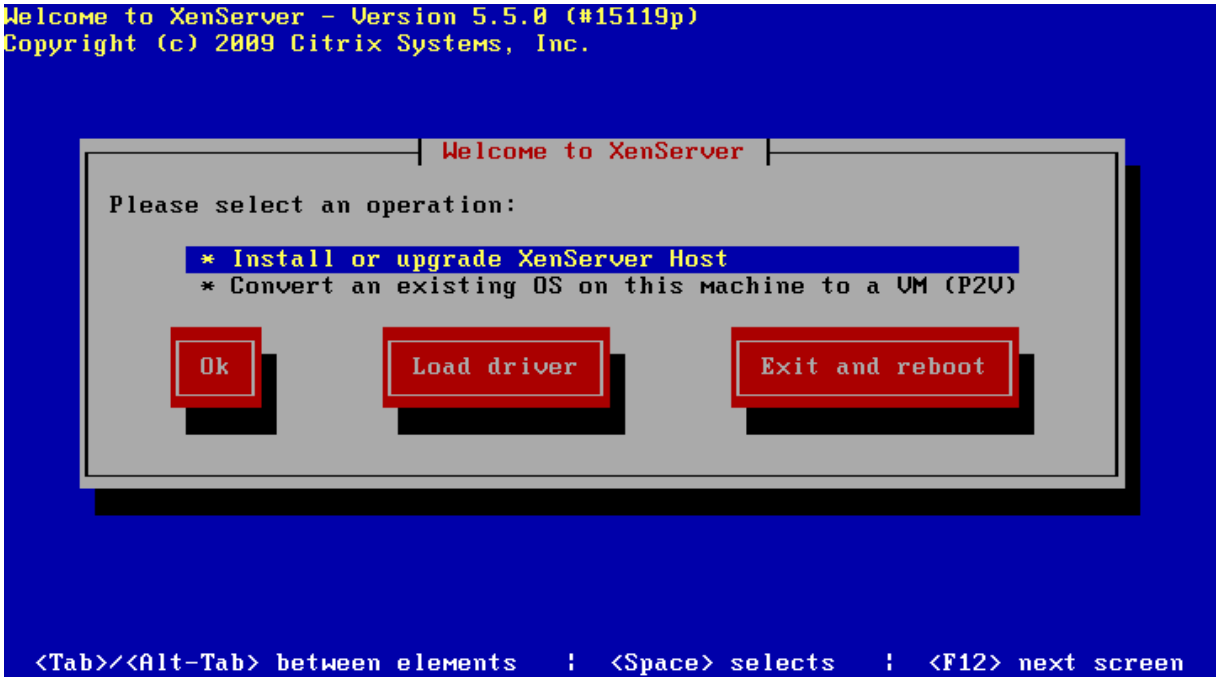
Videre for å administrere og konfigurere serveren installeres vSphere clint på en annen maskin. vSphere er ett enkelt GUI basert program som kobler seg til virtualiserings serveren og lar deg overvåke,opprette, slette og konfigurere virtuelle maskiner. Installasjonen av vSphere er enkel og rett frem. Det er ingen spesielle konfigurasjoner som kreves for å gjennomføre denne installasjonen. Man logger seg på virtualisering serven ved å skrive ip, brukernavn og passord. Passord og brukernavn må settes på serven før man kan logge på med vSphere client.

10 Vedlegg

XEN SERVER 5.5

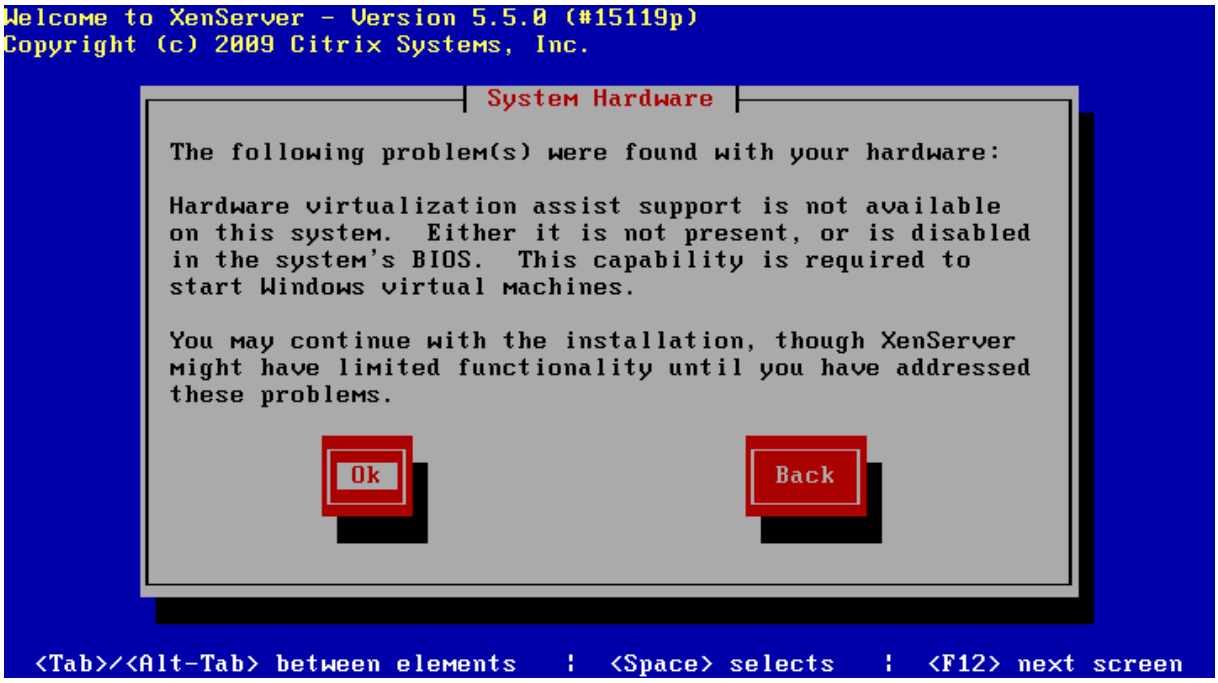
For å få installert Xen Server 5.5 må man først registrere seg Citrix sin hjemme siden for Xen server. Man får da muligheten til å leste ned en installasjons CD i form av en iso fil. Dette er en versjon som krever aktivering etter 30 dager ellers slutter den å fungere.

Etter en oppstartet installasjon av Xen Server 5.5 vil man få to valg. Et for installering/upgradering av Xen Server eller convertering av gjeldene opprativsystem til en virtuell maskin under XenServer som man ser på figur 24.



Figur 24 – Skjermdump fra XenServer install.

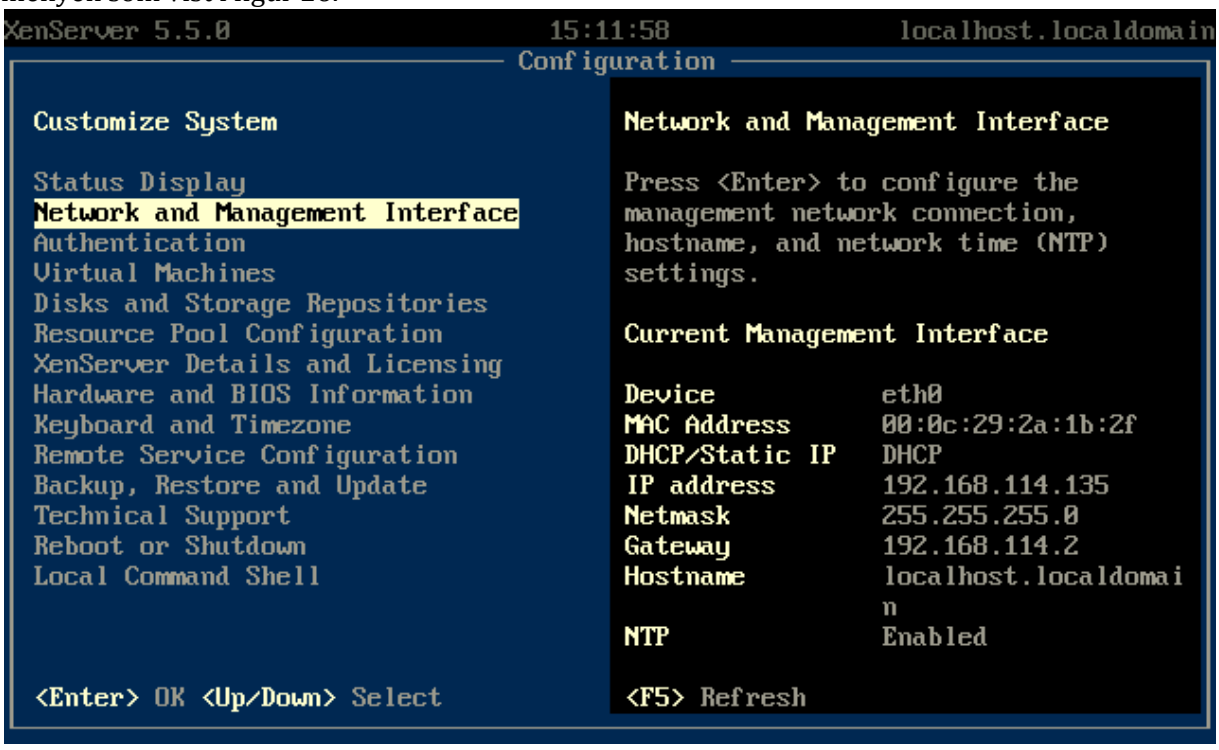
Dersom man har hardware som ikke har virtualisering støtte vil man få opp følgende melding som i figur 25. Man vil fortsatt ha mulighet til å installere hypervisoren men opprativsystemer som Windows vil ikke kunne kjøres som gjeste opprativsystem.



Figur 25 – Skjermdump fra XenServer install medling.

Neste 2 punkter er å sette root passord og bekrefte bruk av harddisk plass nødvendig for hypervisoren.

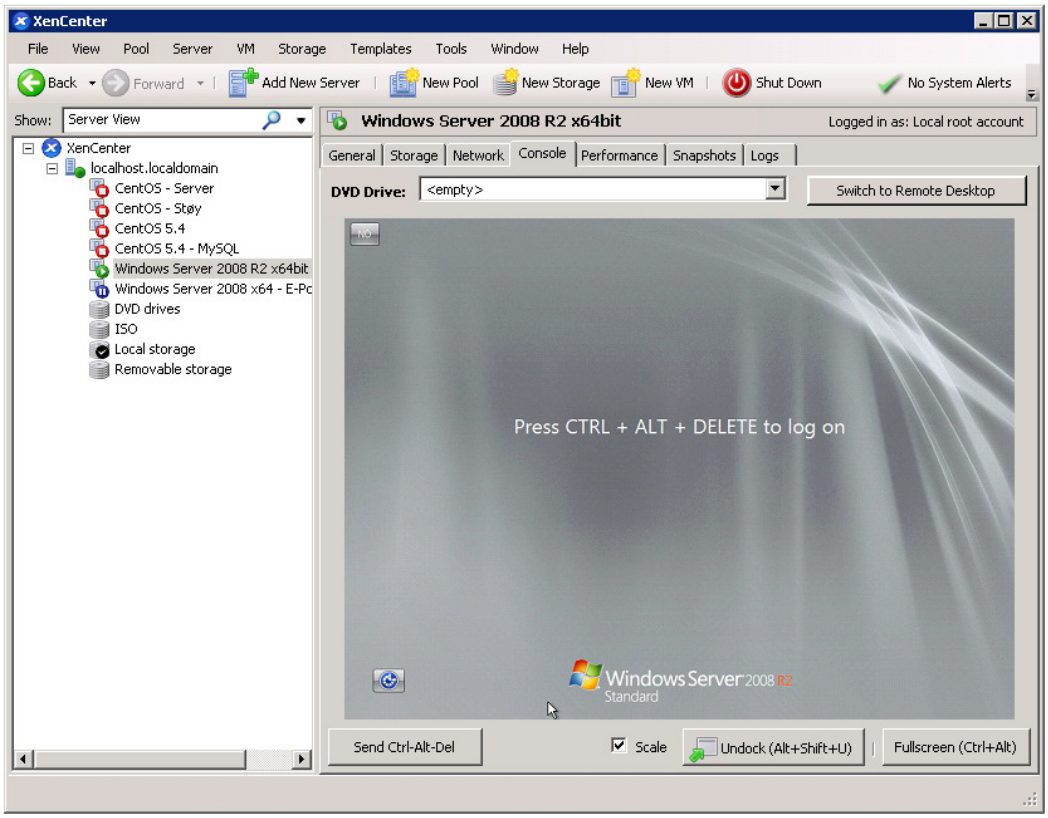
Etter en ferdig installasjon av Xen Server og serveren er i kjørende tilstand vil man få Server menyen som vist i figur 26.



Figur 26 – Skjermdump fra XenServer meny.

Det er flere svært funksjonelle funksjoner som blant annet migrering, backup, krypterings nøkler, bruker kontroll, linux kommando shell og passord som kan gjøres fra denne siden. For oss ble kun notering av ip adresser og oppstarting av gjeste operativsystemer brukt.

Når serveren er i kjørende tilstand slik som på figur x er det mulig å bruke klient applikasjonen Citrix XenCenter 5.5 på en ekstern maskin til å administrere serveren via et grafisk grensesnitt som vist i figur 27.



Figur 27 – Skjermdump av XenCenter.

KONFIGURASJONS OPPSETT FOR LDAP TEST PÅ MICROSOFT ACTIVE DIRACTORY

Kjørende Opprativsystemer

Native:

- CentOS – Test kontroller

Virtuelle maskiner:

- CentOS – Støymaskin
- CentOS – Windows Server 2008

Klargjøring for test miljø

- CentOS – Test kontroller

  Flere av ldap klient scriptene som skal belaste ldap serveren trenger ldap biblotek fil får å kunne kommunisere og behandle ldap data på server.

  ```
  $yum install perl-ldap
  ```

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Porten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

  ```
  line 11: use constant PORT => 1500;
  ```

  Start støyscriptet slik at den er klar for nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

  Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – Windows Server 2008

  Installer Active Diractory

Kjør test Senario 1 eller Senario 2:

Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:

  Start støyscriptet slik at den lytter på nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

- CentOS – Test kontroller

  Legg inn ipaddresse for støymaskin i stoyKlien.conf.
  Andre innstillinger antas å være satt ved konfigurasjon av støyscriptet.

Star test. Parametere for ”ldapTestSenario1.pl” og ”ldapTestSenario2.pl” er like og det blir derfor bare vist eksempler med ”ldapTestSenario1.pl”:

$perl ldapTestSenario1.pl *resultatFil serverIp morDn brukerdn passord antallBrukere maksUndermapper maksSok nyeLdifFiler filnavnBrukere filnavnOrg*
Eller
$perl ldapTestSenario2.pl *resultatFil serverIp morDn brukerdn passord antallBrukere maksUndermapper maksSok nyeLdifFiler filnavnBrukere filnavnOrg*

Eksempel for Windows Active diractory:
$perl ldapTestSenario1.pl resultat.log 192.168.1.10 DC=domene,DC=no CN=Administrator,CN=users,DC=domene,DC=no passord 3000 5 1000 1 brukere3000.ldif org5sub.ldif

Forklaring på parametere:
resultatFil – Navn på logfil
serverIp – IP/dns til OpenLDAP server
morDn – Top organisasjons klassen laget under installering
brukerdn – Brukernavn for innlogging på OpenLDAP server
passord – Passord til bruker for OpenLDAP server
antallbrukere – Antall user klasser som skal lages for testen
maksUndermapper – dypde på organisasjons klasser under hverandre
maksSok – Søk antall for nøkkel og ikke nøkkel atributter
nyeLdifFiler – (1 | 0) Skal nye ldif filer lages. Dette gjøres kun en gang for å få lik database ved kjøring på andre hypervisorer eller nativ.
filnavnBrukere – Ldif fil for user klassen
filnavnOrg – Ldif fil for organisasjons klassen

KONFIGURASJONS OPPSETT FOR LDAP TEST PÅ OPENLDAP

Kjørende Opprativsystemer:

Native:

- CentOS – Test kontroller

Virtuelle maskiner i samme hypervisor:

- CentOS – Støymaskin
- CentOS – OpenLDAP Sever

Klargjøring for test miljø

- CentOS – Test kontroller

  Flere av ldap klient scriptene som skal belaste ldap serveren trenger ldap biblotek fil får å kunne kommunisere og behandle ldap data på server.

  $yum install perl-ldap

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

Alternativ:
Porten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

```
line 11: use constant PORT => 1500;
```

Start støyscriptet slik at den er klar for nettverks kommandoer

```
$perl cpuLastKontroll.pl
```

Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – OpenLDAP Server
  Installer OpenLDAP server

```
$yum install openldap-servers openldap-clients
```

Konfigurer OpenLDAP Server

Rediger ”/etc/openldap/slapd.conf”:

```
suffix "dc=domene,dc=no"
rootdn "cn=root,dc= domene,dc=no"
rootpw passord
```

Kopier standard LDAP oppsett til aktivt oppsett

```
$cp /etc/openldap/DB_CONFIG.example /var/lib/ldap/DB_CONFIG
```

Start OpenLDAP Serveren

```
$ /etc/init.d/ldap start
```

Lag organisasjons klasse for testen ved å lage filen ”mittDomene.ldif” med følgende innhold:

```
dn: dc= domene,dc=no
objectClass: dcObject
objectClass: organization
dc: domene
description: Eksempel
o: Eksempel
```

Kjør så ldap klient for å legge inn organisasjons klassen.

```
$ldapadd -H ldap://localhost -x -D "cn=root,dc=domene,dc=no"  -f mittDomene.ldif -w password
```

Kjør test Senario 1 eller Senario 2:

Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:
  Start støyscriptet slik at den lytter på nettverks kommandoer

$perl cpuLastKontroll.pl

- CentOS – Test kontroller

  Legg inn ipaddresse for støymaskin i stoyKlien.conf
  Andre innstillinger antas å være satt ved konfigurasjon av støyscriptet.

  Start test. . Parametere for "ldapTestSenario1.pl" og "ldapTestSenario2.pl" er like og det blir derfor bare vist eksempler med "ldapTestSenario1.pl":

$perl ldapTestSenario1.pl *resultatFil serverIp morDn brukerdn passord antallBrukere maksUndermapper maksSok nyeLdifFiler filnavnBrukere filnavnOrg*
Eller
$perl ldapTestSenario2.pl *resultatFil serverIp morDn brukerdn passord antallBrukere maksUndermapper maksSok nyeLdifFiler filnavnBrukere filnavnOrg*

Eksempel:
$perl ldapTestSenario1.pl resultat.log 192.168.1.10 DC=domene,DC=no CN=root,DC=domene,DC=no passord 3000 5 1000 1 brukere3000.ldif org5sub.ldif

Forklaring på parametere:
resultatFil – Navn på logfil
serverIp – IP/dns til OpenLDAP server
morDn – Top organisasjons klassen laget under installering
brukerdn – Brukernavn for innlogging på OpenLDAP server
passord – Passord til bruker for OpenLDAP server
antallbrukere – Antall user klasser som skal lages for testen
maksUndermapper – dypde på organisasjons klasser under hverandre
maksSok – Søk antall for nøkkel og ikke nøkkel atributter
nyeLdifFiler – (1 | 0) Skal nye ldif filer lages. Dette gjøres kun en gang for å få lik database ved kjøring på andre hypervisorer eller nativ.
filnavnBrukere – Ldif fil for user klassen
filnavnOrg – Ldif fil for organisasjons klassen

KONFIGURASJONS OPPSETT FOR E-POST TEST:

Kjørende Opprativsystemer

Native:

- CentOS – Test kontroller

Virtuelle maskiner i samme hypervisor:

- CentOS – Støymaskin
- CentOS – Dovecot og postfix

Klargjøring for test miljø

- CentOS – Test kontroller

  For å kunne koble til IMAP server med perl scriptene vi å laste ned følgende bibliotekfiler via yum

```
$yum install perl-Net-IMAP-Simple
```

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Proten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

  ```
  line 11: use constant PORT => 1500;
  ```

  Start støyscriptet slik at den er klar for nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

  Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – Dovecot og postfix

  Installer dovecot, postfix og konfigrasjons verktøy for endring av standard e-post håndterer.

  ```
  $yum install postfix dovecot system-switch-mail system-switch-mail-gnome
  ```

  Alternativ
  Ved mangel av dns server legg til følgende linje i filen "/etc/hosts".
  Rediger og legg til:
  MaskinIp epost.domene.no epost

  Konfigrasjon av postfix

  Rediger "/etc/postfix/main.cf":
  myhostname = epost.domene.no
  mydomain = domene.no
  myorigin = $mydomain
  inet_interfaces = all
  mydestination = $myhostname, localhost.$mydomain, localhost, $mydomain
  mynetworks = (Din IP)/(Din prefix), 127.0.0.0/8
  home_mailbox = Maildir/

  Rediger "/etc/dovecot.conf":
  protocols = imap imaps pop3 pop3s
  mail_location = maildir:~/Maildir
  pop3_uidl_format = %08Xu%08Xv

  Sørg for å bytte ut standard e-post sender med postfix

  ```
  $system-switch-mail
  ```

  Legger til dovecot oprativsystem oppstart

$/sbin/chkconfig --level 345 dovecot on

Start dovecot og gjør en omstart av postfix.

$/etc/init.d/dovecot start
$/etc/init.d/postfix restart

Bruk perl scriptet "lagEpostadr.pl" for det totale antall epost addresser du ønsker å ha med i testen.

$perl lagEpostadr.pl brukernavn password antall
eks: $perl lagEpostadr.pl epostBruker password 10

I dette tilfellet vil epost adressene se slik ut, epostBruker1@domene.no, epostBruker2@domene.no .. og videre

Kjør test Senario 1 eller Senario 2:

Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:

  Start støyscriptet slik at den lytter på nettverks kommandoer

  $perl cpuLastKontroll.pl

- CentOS – Dovecot og postfix

  Sjekk at dovecot og postfix kjører.

- CentOS – Test kontroller

  Alternativ
  Ved mangel av dns server legg til følgende linje i filen "/etc/hosts".
  Rediger og legg til:
  MaskinIp epost.domene.no epost

  Legg inn ipadresse for støymaskin i "stoyKlien.conf"
  Andre innstillinger støytid og støybelastning antas å være satt i scriptet.

  Start test. Parametere for epostTestSenario1.pl og epostTestSenario2.pl er like og det blir derfor bare vist eksempler med " epostTestSenario1.pl".

$perl epostTestSenario1.pl resultatFil domene imapserver smtpServer brukernavn passord antMld mldStr antMapper antbrukere

Eks: $perl epostTestSenario1.pl epost.log domene.no epost.domene.no epost.domene.no epostbruker passord 1000 3000 10 10

resultatFil – Skriv test resultat til denne filen.
Domene – Epost domene
Imapserver – Imap addresen til epost serveren
smtpServer – Smtp addressen til epost serveren
brukernavn – brukernavn brukt ved laging av epost adresser med perl scriptet "lagEpostadr.pl".

passord – Felles passord for brukerne laget i perl scriptet ”lagEpostadr.pl”.
antMld – Antall melding til hver bruker epost
mldStr – Meldings størrelse for tekst innhold i hver epost
antMapper – Antall mappe oppretting i E-postkonto for e-post melding kopier.
antbrukere – Antall epost kontoer som skal belastes. Kan ikke være over e-post antall laget i perl scriptet ”lagEpostadr.pl”.

KONFIGURASJONS OPPSETT FOR FTP TEST:

Kjørende Opprativsystemer:

Native:

- CentOS – Test kontroller og FTP Klient

Virtuelle maskiner:

- CentOS – Støymaskin
- CentOS – FTP Server

Klargjøring for test miljø

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Porten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

  ```
  line 11: use constant PORT => 1500;
  ```

  Start støyscriptet slik at den er klar for nettverks kommandoer.

  ```
  $perl cpuLastKontroll.pl
  ```

  Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – FTP Server

  FTP serveren som brukes I testen er vsftpd og installers på følgende måte fra yum.

  ```
  $yum install vsftpd
  ```

  Automatisk oppstart av vsftpd gjøres på følgende måte.

  ```
  $/sbin/chkconfig --level 345 vsftpd on
  ```

Kopier in perl scriptet “ftpLagfiler.pl”.

Alternativ:
Dersom egendefinerte filstørrelser skal settes må array @kbytes og @antallFiler redigeres i perl scriptet “ftpLagfiler.pl”.

Kjør så scriptfilen ” ftpLagfiler.pl”. Scriptet sørger for at det opprettes filer for nedlasting i mappen /var/ftp. Som standard er disse tilgjenglig via ftp konto ”anonymous”.

```
$perl ftpLagfiler.pl
```

Kjør test Senario 1 eller Senario 2:

Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:

  Start støyscriptet slik at den lytter på nettverks kommandoer.

  ```
  $perl cpuLastKontroll.pl
  ```

- Virtuell CentOS – FTP Server

  Sjekk at vsftpd kjører.

- CentOS – Test kontroller

  Legg inn ipaddresse for støymaskin i stoyKlien.conf
  Andre innstillinger støytid og støybelastning antas å være satt i scriptet.

  Start test. Parametere for ”startFtpTestSenario1.pl” og ”startFtpTestSenario2.pl” er like. Det blir derfor bare vist eksempler med ”startFtpTestSenario1.pl”.

  ```
  $perl startFtpTestSenario1.pl resultatFil server
  ```

  Eks:
  ```
  $perl startFtpTestSenario1.pl ftp.log 192.168.1.10
  ```

  Alternativ:
  Dersom vsftpd er konfigurert med brukernavn og passord må dette sendes med som parametere for starting av testen.
  ```
  $perl startFtpTestSenario1.pl resultatFil server brukernavn passord
  ```

  Eks:
  ```
  $perl startFtpTestSenario1.pl ftp.log 192.168.1.10 ftpBruker passord
  ```

  Forklaring på parametere:
  resultatFil – Angi navn for logfil.
  server – Angi ip/dns for ftpserver.
  brukernavn – Angi brukernavn som gir tilgang til testfilene.
  passord – Angi passord som gir tilgang til testfilene.

KONFIGURASJONS OPPSETT FOR MYSQL TEST

Kjørende Opprativsystemer:

Native:

- CentOS – Test kontroller

Virtuelle maskiner i samme hypervisor:

- CentOS – Støymaskin
- CentOS – MySQL Server

Klargjøring for test miljø

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Porten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

  ```
  line 11: use constant PORT => 1500;
  ```

  Start støyscriptet slik at den er klar for nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

  Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – MySQL Server

  MySQL server må installeres med MySQL Benchmark script.

  ```
  $yum install mysql mysql-bench mysql-server
  ```

Kjør test Senario 1 eller Senario 2:

Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:

  Start støyscriptet slik at den lytter på nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

- Virtuell CentOS – MySQL Server

  Start MySQL server med følgende kommando

$/etc/init.d/mysqld start

Start perl scriptet ”testMysql.pl”.

$perl testMysql.pl

- CentOS – Test kontroller

  Legg inn ipaddresse for støymaskin i stoyKlien.conf
  Andre innstillinger støytid og støybelastning antas å være satt i scriptet.

  Start test. Parametere for ”testMysqlSenario1.pl” og ” ”testMysqlSenario2.pl” er like og det blir derfor bare vist eksempler med ” testMysqlSenario1.pl”:

$perl testMysqlSenario1.pl resultatFil mysqlServer mysqlServerPort testScript
Eksempel:
$perl testMysqlSenario1.pl mysql.log 192.168.1.10 1501 run-all-tests

Forklaring på parametere:
resultatFil – Navn til logfilen
mysqlServer – Ip/dns adresse til mysql server
mysqlServerPort – Nettverks kommando port for kjørende perl script ”testMysql.pl” på test maskin
testScript – Filnavn til Mysql benchmarkscript i mappen “/usr/shared/sql” på test maskin.

KONFIGURASJONS OPPSETT FOR MICROSOFT TERMINAL SERVER TEST

Kjørende Opprativsystemer:

Native:

- CentOS – Test kontroller

Virtuelle maskiner i samme hypervisor:

- CentOS – Støymaskin
- Windows Server 2008 – Terminal Server

Klargjøring for test miljø

- CentOS – Test kontroller

  For å kunne koble opp mot terminal serveren må rdesktop være installert

  $yum install rdesktop

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Porten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

line 11: use constant PORT => 1500;

Start støyscriptet slik at den er klar for nettverks kommandoer

$perl cpuLastKontroll.pl

Bruk terminal eller NC for å finne støy verdier som du ønsker å bruke i testen. Se fremgangs måte for støyscript.

- Virtuell CentOS – Windows Server 2008
  Installer Active Diractory
  Installer Terminal Service
  Installer Perl kompilator

  Sørg for at Perl kan ta i mot nettverks trafikk untenifra ved å konfigurere Windows firewall.

  Kjør "secpol.msc" i run.exe
  Gå til Følgende område: "/Local Policies/user rights assignments/"

  Åpne "allow logon through terminal services" og Legg til gruppen "Remote Desktop Users".

  Kopier over perl scriptet "lagBrukere.pl".

  Lag "logout.bat" under mappen \Users\Default\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\startup

  "logout.bat":
  Shutdown.exe -l

  Dette for automatisk utlogging rett etter innloging.

Kjør test Senario 1 eller Senario 2:
Videre fremgangsmåter forutsetter at stoyKlient.conf er konfigurert med støymengde og tidskjøringer. Se fremgangsmåte for støyscript.

- Virtuell CentOS – Støymaskin:
  Start støyscriptet slik at den lytter på nettverks kommandoer.

  $perl cpuLastKontroll.pl

- Virtuell CentOS – Windows Server 2008
  Start perl scriptet "lagBrukere.pl" som administrator bruker.

- CentOS – Test kontroller
  Legg inn ipaddresse for støymaskin i stoyKlien.conf
  Andre innstillinger støytid og støybelastning antas å være satt i scriptet.

  Start test. Parametere for tsTestSenario1.pl og tsTestSenario2.pl er like.
  Eksempler blir derfor bare vist med " tsTestSenario1.pl":

$perl tsTestSenario1.pl *resultatFil antallBrukere bruker passord domene ipaddresse oppløsning*

Eksempel:
tsTestSenario1.pl resultat.log 10 tsBruker passord domene.no ts.domene.no 800x600

Forklaring på parametere:
resultatFil – Navn på logfil
antallBrukere – Antall brukere som må lages
bruker – Bruker navn (eks tsBruker blir tsBruker1, tsBruker2, tsBruker3…)
passord – Passord for innlogging. Felles for alle brukere
domene – Active Diractory domet
ipaddresse – Ip/DNS til terminal serveren
oppløsning – Skjerm oppløsning for bruker innloggingene

KONFIGURASJONS OPPSETT FOR STØYSCRIPT

Kjørende Opprativsystemer:

Native:

- CentOS – Test kontroller

Virtuelle maskiner i samme hypervisor:

- CentOS – Støymaskin

Klargjøring for test miljø

- Virtuell CentOS – Støymaskin

  Støyscriptet bruker port 1500 som standard og dersom brannmur er på slått må denne porten være åpen for innkommende trafikk i oprativsystemet.

  Alternativ:
  Proten kan endres på linje 11 i perl filen cpuLastKontroll.pl dersom man vil ha en annen lytte port enn 1500.

  ```
  line 11: use constant PORT => 1500;
  ```

  Start støyscriptet slik at den er klar for nettverks kommandoer

  ```
  $perl cpuLastKontroll.pl
  ```

- CentOS – Test kontroller

  Rediger filen ”stoyKlient.conf”, Konfigrasjons punkter:

  stoyServer – Ipaddressen til virtuelle maskinen som kjører støyscriptet
  stoyPort – TCP port for nettverks kommandoer
  kjoreTid – Liste for antall sekunder for hver støy belastning
  stoyString – Liste for stoy lengde for en belastnings mengde
  stoySleep – Liste for sleep mellom hver belastnings mengde i stoyString
  stoyTraader – Antall tråder som skal brukes i alle belastningene

stoyMedIo – {1|0} svakt io belastning av/på i alle støy belastningene
pauseTid – Pause mellom hver testkjøring i test senario

Legg til verdier i stoyServer, stoyPort, kjoreTid, stoyTraader, stoyMedIo, pauseTid.

Eksempel :
stoyServer: 128.39.81.141
stoyPort: 1500
kjoretid: 600 3600
stoyString:
stoySleep:
stoyProsent:
stoyTraader: 3
stoyMedIo: 1
pauseTid: 120

Finn konfigurasjons verdier for støy prosent:

- Virtuell CentOS – Støymaskin:

  Start støyscriptet slik at den lytter på nettverks kommandoer

  $perl cpuLastKontroll.pl

- CentOS – Test kontroller

  Legg inn ipadresse for støymaskin i "stoyKlien.conf".

  Start telenet med ipadresse og port nummer for støyskriptet kjørende på støymaskinen.

  Eksempel:
  telnet 192.168.0.10 1500

  Kommando i telenet:
  telnet: antTraader tid rtxtLengde prosentLast soveTid ioPaa
  eller
  telnet: STOP

  antTraader – Antall tråder som skal være i testen
  tid – {EVIG | antall sekunder} Tid testen skal kjøre
  rtxtLengde – String lengde som genereres for belastning
  prosentLast – CPU last i prosent av den totale ytelsen
  soveTid – sovetid mellom generering av stringverdi
  ioPaa – { 1 | 0} – Generer svak iobelastning på eller av.
  STOP – Dersom støykjøring er satt til evig i parameter "tid" stoppes kjøringen med "STOP" som kommando alene.

  Finn støy verdi med å la "tid" være 0.
  Kommando eks:
  telnet: 3 0 1000 40 1500 1

  Ved ferdig søk returneres kommandoen med oppdatert rtxtLengde og soveTid som passer cpu last via angitt prosentverdi. Fyll søk verdiene inn i filen

stoyKlient.conf. prosentLast til stoyString, soveTid til stoySleep og prosentLast til stoyProsent.

Eksempel:
telnet Mottatt: *3 0 1456 40 2123 1*

”stoyKlient.conf"
Rediger i stoyKlient.conf
stoyString: 1456
stoySleep: 2123
stoyProsent: 40

Eksempel ”stoyKlient.conf" med flere støynivåer:
stoyServer: 128.39.81.141
stoyPort: 1500
kjoretid: 600 3600
stoyString: 1959 6308
stoySleep: 2016 303
stoyProsent: 30 80
stoyTraader: 3
stoyMedIo: 1
pauseTid: 120

F. TEST SENARIOER

WEBSERVER TEST

WEBSERVER SENARIO: BARE-METALL V.S VIRTUALISERING

Test type:
Apache 2.2 web server

Mål med test:
Finne forskjellen fra ”bare metall” til virtualisering.

Operativsystem for test:
Windows server 2008 rett på virtualiserings serveren
Virtuell Windows server 2008 installert på VMware esxi og XenServer

Installerte programmer for mål maskin:
Apace (for tilgang til apache bench og web server)

Generering av belastning:
Apache bench lokalt på maskinen kontrollert av skript.
abTester.pl

Grader av last på målmaskin:

- Max belastning: 500 000 forespørsler fordelt på 50 klienter
- Middels belastning: 50 000 forespørsler fordelt på 10 klienter
- Lav belastning: 10 000 forespørsler fordelt på 5 klienter

Varighet:
Ingen varighet satt. Apache bench skal få kjøre alle tre grader av belastning før den er ferdig.

Oppsett av maskiner rundet testen på virtualisers server:
Ingen andre maskiner. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen:
Ingen andre maskinger. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Målverktøy:
Ved måling av selve webserver tjenesten vil det bli brukt apache bech for generering og samling av data.

Måleverdier av betydning:

- Tiden det tar å utføre X antall forespørsler
- Antall feilede forespørsler
- Gjennomsnittlig tid pr forespørsel
- Antall tilkoblinger
- Antall feilede tilkoblinger

WESBSERVER SENARIO 1: INTERVALLER

Test type: Apache 2.2

Mål med test:
Belaste målmaskin med rollen som webserver med en gitt AB test, samt generere støy på maskiner rundt målmaskinen i intervaller mellom AB testene, for så utfør en ny AB test for å se hvordan dette påvirker test resultat. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Operativsystem for test:
Windows server 2008.

Installerte programmer for mål maskin:
Apace
Skript for starting og stopping av AB og støy skript
Netcat

Generering av belastning:
Apache bench lokalt på maskinen kontrollert av skript og skript for kontroll av støy.
abTestSenario1.pl
stoyKlinet.pm
stoyKlint.conf

Grader av last på målmaskin:

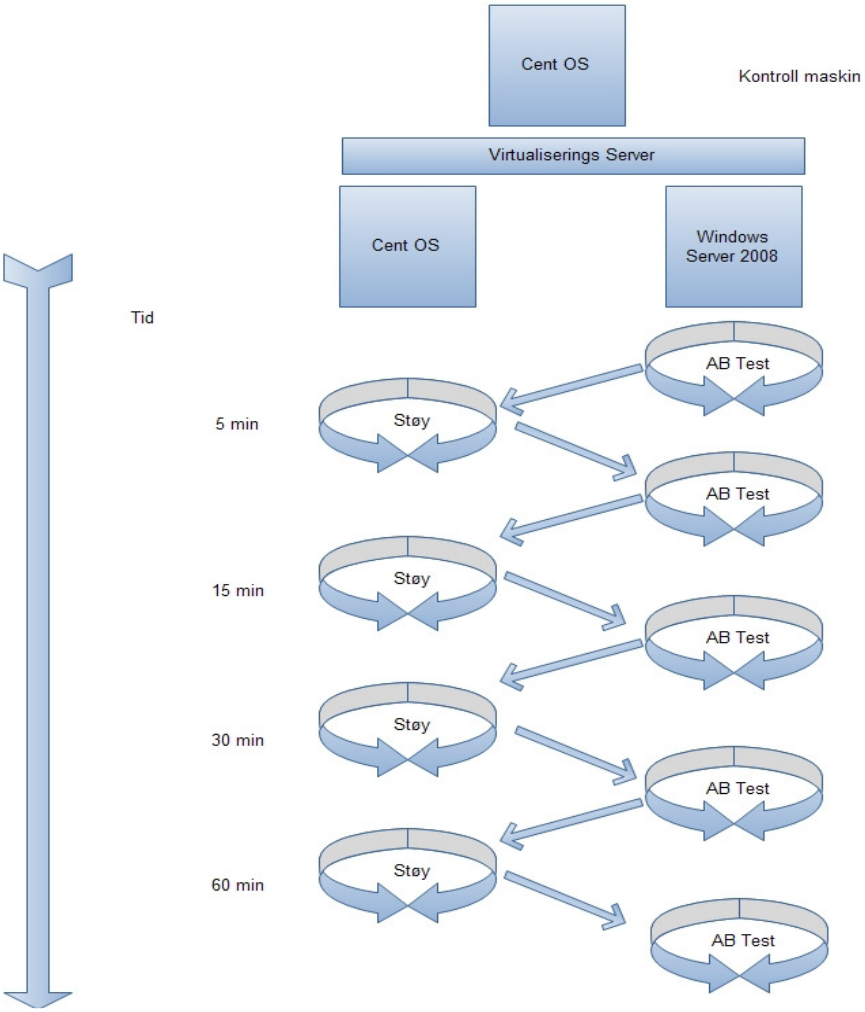
- Max belastning: 500 000 forespørsler fordelt på 50 klienter
- Middels belastning: 50 000 forespørsler fordelt på 10 klienter
- Lav belastning: 10 000 forespørsler fordelt på 5 klienter

Varighet:
Testen vil bli delt inn i 4 intervaller.

- 10 min
- 15 min
- 30 min
- 60 min

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Imellom hver av disse intervallene vil det kjøres AB tester. Men henholdsvis middels belastning, dette er noe som kan endres dersom det er ønskelig.
Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server:
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen:
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.

- Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt.
- Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt.
- Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt.
- Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt.

Man kan kjøre denne testen i to omganger, en gang slik som beskrevet over eller en gang med 80% belasting fra støy maskinen.

Hvordan lasten skal genrers:
Lasten eller støyen på Cent maskinene generert ved kjøring av støy skriptet som er forklart i punkt 5.1.

Målverktøy:
Ved måling av selve webserver tjenesten vil det bli brukt apache bech for generering og samling av data.

Måleverdier av betydning:

- Tiden det tar å utføre X antall forespørsler
- Antall feilede forespørsler
- Gjennomsnittlig tid pr forespørsel
- Antall tilkoblinger
- Antall feilede tilkoblinger

WEBSERVER SENARIO 2: STØY SAMTIDIG

Test type:
Apache 2.2 web server

Mål med test:
Belaste målmaskin med rollen som webserver med en gitt AB test samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test:
Cent OS 5.4

Installerte programmer for mål maskin:
Apace
Skript for starting og stopping av AB og støy skript
Netcat

Generering av belastning:
Apache bench lokalt på maskinen kontrollert av skript og skript for kontroll av støy.
abTestSenario2.pl
stoyKlinet.pm
stoyKlint.conf

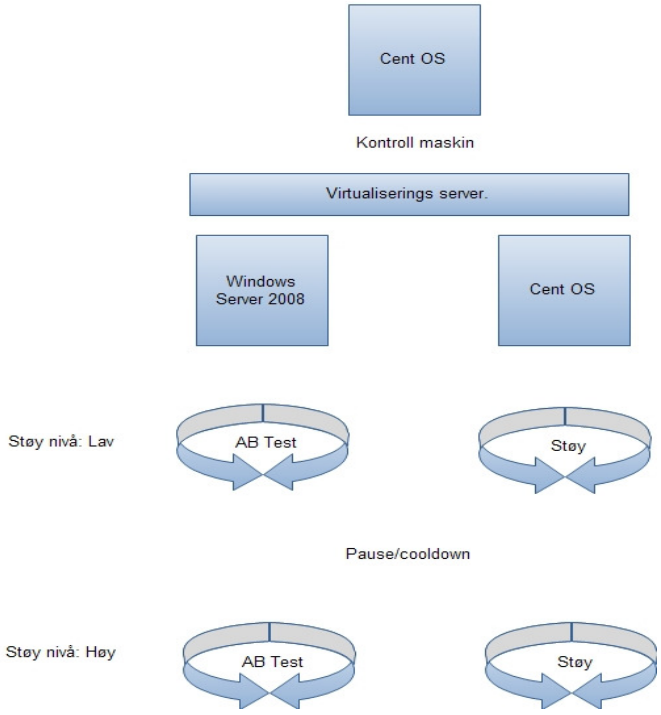
Grader av last på målmaskin:

- Max belastning: 500 000 forespørsler fordelt på 50 klienter
- Middels belastning: 50 000 forespørsler fordelt på 10 klienter
- Lav belastning: 10 000 forespørsler fordelt på 5 klienter

Varighet:
Ingen varighet på test satt. Kun en 10 Minutters pause/cooldown mellom testene.

Oppsett av maskiner rundet testen på virtualisers server:
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.

10 Vedl



Last som skal generer av maskiner rundt målmaskinen:
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last. Men det vil være den gjennomsnittlige belastning maskinen genererer i løp av perioden den skal lage støy.
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Hvordan lasten skal genrers:
Støymaskinen generer støy ved kjøring av støy skriptet som er forklart i punkt 5.1.

Målverktøy:
Ved måling av selve webserver tjenesten vil det bli brukt apache bech for generering og samling av data.

Måleverdier av betydning:
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall forespørsler
- Antall feilede forespørsler
- Gjennomsnittlig tid pr forespørsel
- Antall tilkoblinger
- Antall feilede tilkoblinger

WEBSERVER SENARIO 3: INTERVALLER OG SAMTIDIG

Test type:
Apache 2.2

Mål med test:
Belaste målmaskin med rollen som webserver med en gitt AB test, samt generere støy på maskiner rundt målmaskinen i intervaller mellom AB tester. Og utføre AB test samtidig som det støyes på maskiner rundt. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser når andre maskiner har tilgang på dem.

10 Vedlegg

Operativsystem for test:
Windows server 2008.

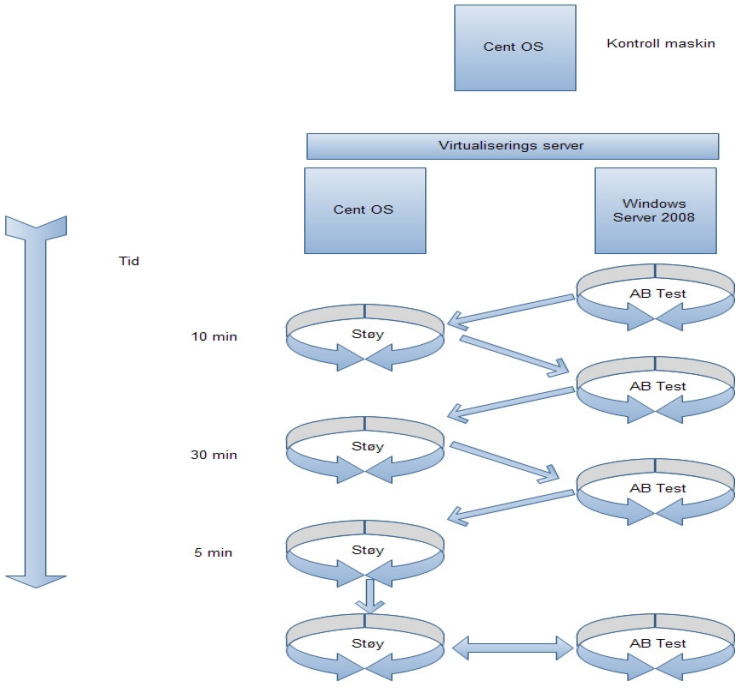
Installerte programmer for mål maskin:
Apace
Skript for starting og stopping av AB og støy skript.
Netcat

Generering av belastning:
Apache bench lokalt på maskinen kontrollert av skript og skript for kontroll av støy.
abTestSenario3.pl
stoyKlinet.pm
stoyKlint.conf

Grader av last på målmaskin:
- Max belastning: 500 000 forespørsler fordelt på 50 klienter.
- Middels belastning: 50 000 forespørsler fordelt på 10 klienter.
- Lav belastning: 10 000 forespørsler fordelt på 5 klienter.

Varighet:
Testen vil bli delt inn i 3 intervaller av støy.
Intervall 1 = 10 min.
Intervall 2 = 30 min.
Intervall 3 = 5 min.
Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen.
De to førest intervallene er for å se hvordan hypersisoren oppfører seg når det er flere maskiner som bytter på resursene.
Det siste er for å tildele støymaskinen resurser før det starts en test samtidig som dette intervallet fortsetter å støye til testen er ferdig.
Testen avsluttes med at det blir kjørt en test samtidig som det støyes fra maskinen rundt. Når AB testen er ferdig vil alt avsluttes.
Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server:
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen:
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last. Men det vil være den gjennomsnittlige belastning maskinen genererer i løp av perioden den skal lage støy.
Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Hvordan lasten skal genrers:
Lasten eller støyen på Cent maskinene generert ved kjøring av støy skriptet som er forklart i punkt 5.1.

Målverktøy:
Ved måling av selve webserver tjenesten vil det bli brukt apache bech for generering og samling av data.

Måleverdier av betydning:
- Tiden det tar å utføre X antall forespørsler
- Antall feilede forespørsler
- Gjennomsnittlig tid pr forespørsel
- Antall tilkoblinger
- Antall feilede tilkoblinger

LDAP OG AD

LDAP OG AD SENARIO: BARE-METALL V.S VIRTUALISERING

Test type
LDAP og AD kontroller.

Mål med test
Finne forskjellen fra ”bare metall” til virtualisering av LDAP/AD kontroller.

Operativsystem for test
Windows server 2008 rett på virtualiserings serveren for testing av AD
Virtuell Windows server 2008 installert på VMware esxi og XenServer for testing av AD
Cent OS 5.4 rett på virtualiserings serveren for testing av LDAP
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer for testing av LDAP

Installerte programmer for mål maskin
Windows server 2008: DHCP, Active Directory kontroller og Script for testing
Cent OS 5.4: Open LDAP, LDAP for perl og Script for testing

Generering av belastning
Følgende skript lokalt på maskinen som kjører spørring mot LDAP/ AD.
drift.ldif
endreAtrbutt.pl
lastOppLdifFil.pl
ldapTest.pl
sokBrukereOgSlett.pl
sokEtterAtributt.pl
sokEtterCn.pl
sokEtterOrgOgSlett.pl

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle spørring mot LDAP/AD før testen er ferdig

Oppsett av maskiner rundet testen på virtualisers server
Ingen andre maskiner. Testen utføres først "bare metall" og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen
Ingen andre maskinger. Testen utføres først "bare metall" og deretter på en enkelt virtuell maskin på virtualiserings servene.

Måleverktøy
Ved måling av LDAP/AD kontroller vil skriptet for generering av last ta tiden det tar å utføre spørringene.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Last opp Org
- Last opp Brukere
- Sok etter CN
- Sok etter Atributter
- Enring av brukere
- Slette alle brukere
- Slette alle Org

LDAP OG AD SENARIO 1: INTERVALLER

Test type
LDAP og AD kontroller.

Mål med test
Belaste målmaskin med rollen som LDAP/AD kontroller med test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av skriptene , for så utfør en ny LDAP/AB test for å se hvordan dette påvirker test resultat. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Operativsystem for test
Windows server 2008.
Cent OS 5.4

Installerte programmer for mål maskin
Windows server 2008: DHCP, Active Directory kontroller og Script for testing
Cent OS 5.4: Open LDAP, LDAP for perl og Script for testing

Generering av belastning
Følgende skript lokalt på maskinen som kjører spørring mot LDAP/ AD og støy på støymaskin.
drift.ldif
endreAtrbutt.pl
lastOppLdifFil.pl
ldapTest.pl
sokBrukereOgSlett.pl
sokEtterAtributt.pl
sokEtterCn.pl
sokEtterOrgOgSlett.pl
ldapTestSenario1.pl
stoyKlient.conf
stoyKlient.pm

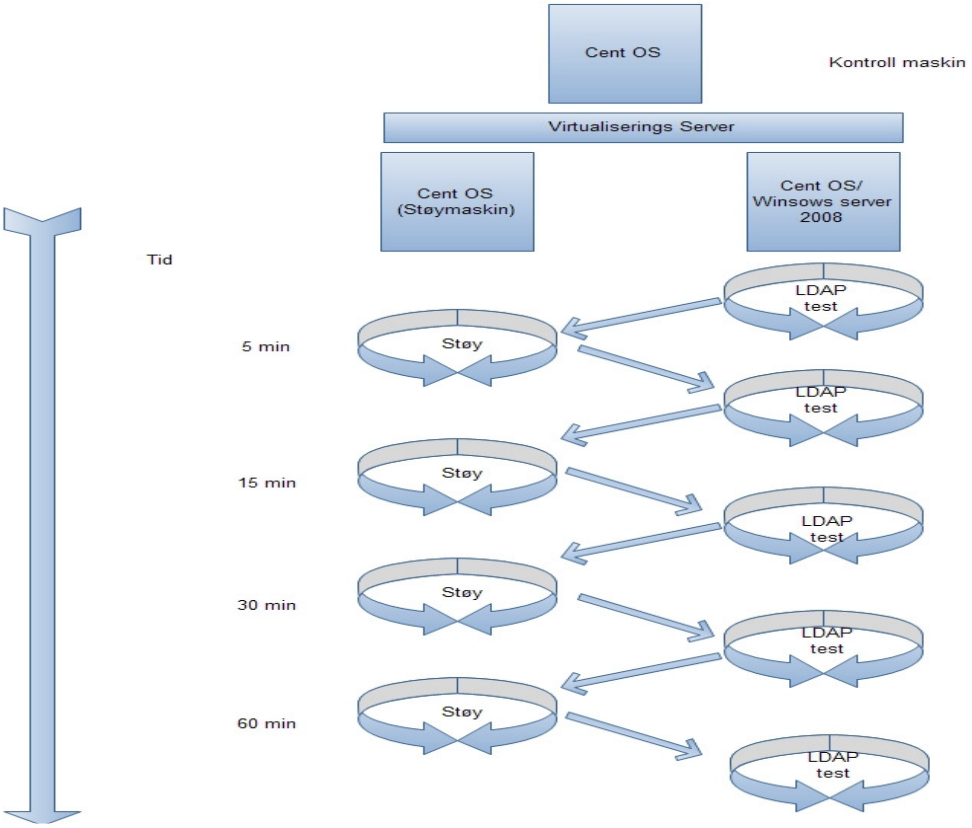
Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Testen vil bli delt inn i 4 intervaller:

- 10 min.
- 15 min.
- 30 min.
- 60 min.

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Imellom hver av disse intervallene vil det kjøres LDAP/AD tester. Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.

- Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt
- Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt

Måleverktøy
Ved måling av LDAP/AD kontroller vil skriptet for generering av last ta tiden det tar å utføre spørringene.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Last opp Org
- Last opp Brukere
- Sok etter CN
- Sok etter Atributter
- Enring av brukere
- Slette alle brukere
- Slette alle Org

LDAP OG AD SENARIO 2: STØY SAMTIDIG

Test type
LDAP og AD kontroller.

Mål med test

Belaste målmaskin med rollen som LDAP/AD kontroller med spørringer gjennom ett skript samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test
Cent OS 5.4
Windows Server 2008

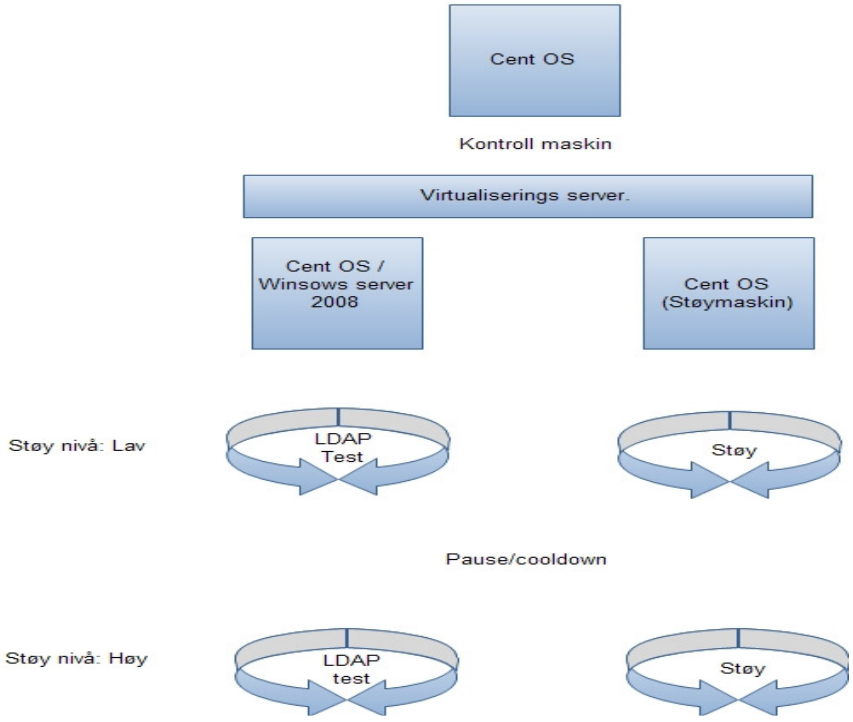
Installerte programmer for mål maskin
Windows server 2008: DHCP, Active Directory kontroller og Script for testing
Cent OS 5.4: Open LDAP, LDAP for perl og Script for testing

Generering av belastning
Følgende skript lokalt på maskinen som kjører spørring mot LDAP/ AD og støy på støymaskin.
drift.ldif
endreAtrbutt.pl
lastOppLdifFil.pl
ldapTest.pl
sokBrukereOgSlett.pl
sokEtterAtributt.pl
sokEtterCn.pl
sokEtterOrgOgSlett.pl
ldapTestSenario2.pl
stoyKlient.conf
stoyKlient.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle spørring mot LDAP/AD før testen er ferdig. Kun en 10 Minutters pause/cooldown mellom testene.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last. Men det vil være den gjennomsnittlige belastning maskinen genererer i løp av perioden den skal lage støy.
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Ved måling av LDAP/AD kontroller vil skriptet for generering av last ta tiden det tar å utføre spørringene.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Last opp Org
- Last opp Brukere
- Sok etter CN
- Sok etter Atributter
- Enring av brukere
- Slette alle brukere
- Slette alle Org

LDAP OG AD SENARIO 3: INTERVALLER OG SAMTIDIG

Test type
LDAP og AD kontroller.

Mål med test
Belaste målmaskin med rollen som LDAP/AD kontroller ved kjøring av skript som utfører spørringer, samt generere støy på maskiner rundt målmaskinen i intervaller mellom LDAP/AD tester.
For så kjøring av støy maskin i en gitt periode før en test starter samtidig som støymaskinen kjører.
Dette for å se hvordan testen påvirkes av at en annen maskin har kontroller over resursene før og under kjøring av test.

Operativsystem for test
Cent OS 5.4
Windows server 2008.

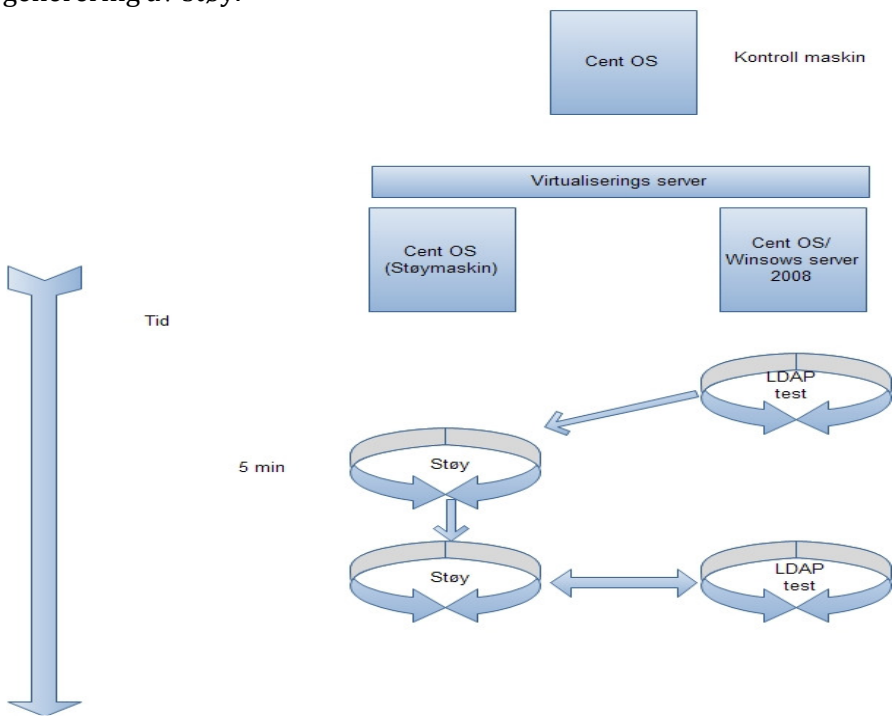
Installerte programmer for mål maskin
Windows server 2008: DHCP, Active Directory kontroller og Script for testing
Cent OS 5.4: Open LDAP, LDAP for perl og Script for testing

Generering av belastning
Følgende skript lokalt på maskinen som kjører spørring mot LDAP/ AD og støy på støymaskin.
drift.ldif
endreAtrbutt.pl
lastOppLdifFil.pl
ldapTest.pl
sokBrukereOgSlett.pl
sokEtterAtributt.pl
sokEtterCn.pl
sokEtterOrgOgSlett.pl
stoyKlient.conf
stoyKlient.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle spørring mot LDAP/AD før testen er ferdig.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen
Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Ved måling av LDAP/AD kontroller vil skriptet for generering av last ta tiden det tar å utføre spørringene.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Last opp Org
- Last opp Brukere
- Sok etter CN
- Sok etter Atributter
- Enring av brukere
- Slette alle brukere
- Slette alle Org

DATABASE

DATABASE SENARIO: BARE-METALL V.S VIRTUALISERING

Test type
Mysql database server

Mål med test
Finne forskjellen fra ”bare metall” til virtualisering.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Mysql
Mysql client
Mysql skript

Generering av belastning
Følgende skript lokalt på maskinen som kjører av database test.
testMysqlKontroll.pl

Grader av last på målmaskin
Det benytte Mysql skript til generering av belastning, det finnes ingen mulighet for å endre graden av denne belastningen.

Varighet
Ingen varighet satt. Mysql skript skal få kjøre til den er ferdig.

Oppsett av maskiner rundet testen på virtualisers server
Ingen andre maskiner. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen
Ingen andre maskinger. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Måleverktøy
Ved måling av selve mysql database serveren benytte skript for generering og samling av data.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Total tid
- Operasjoner i sekundet
- Tid i user og kernel space
- Tid brukt på spesifikke tester
- Antall feilede tilkoblinger

DATABASE SENARIO 1: STØY SAMTIDIG

Test type
Mysql database server

Mål med test
Belaste målmaskin med rollen som database server med gitte test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av skriptene. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Mysql
Mysql client
Mysql skript

Generering av belastning
Følgende skript lokalt på maskinen som kjører av database test og støy på støymaskin.
testMysqlSenario1.pl
stoyKlient.conf
stoyKlient.pm

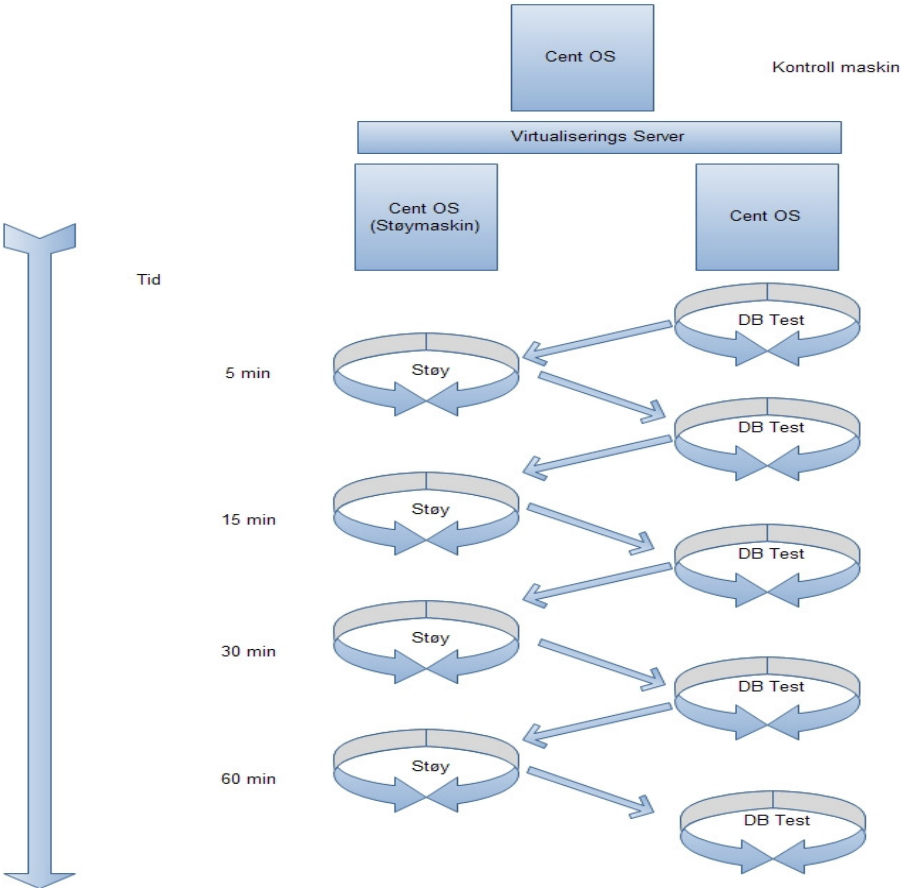
Grader av last på målmaskin
Det benytte Mysql skript til generering av belastning, det finnes ingen mulighet for å endre graden av denne belastningen.

Varighet
Testen vil bli delt inn i 4 intervaller.

- 10 min
- 15 min
- 30 min
- 60 min

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Imellom hver av disse intervallene vil test skriptet kjøres. Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.

- Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt
- Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Ved måling av selve mysql database serveren benytte skript for generering og samling av data.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Total tid
- Operasjoner i sekundet
- Tid i user og kernel space
- Tid brukt på spesifikke tester
- Antall feilede tilkoblinger

DATABASE SENARIO 2: STØY SAMTIDIG

Test type
LDAP og AD kontroller.

Mål med test
Belaste målmaskin med rollen som database server med spørringer gjennom ett skript samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

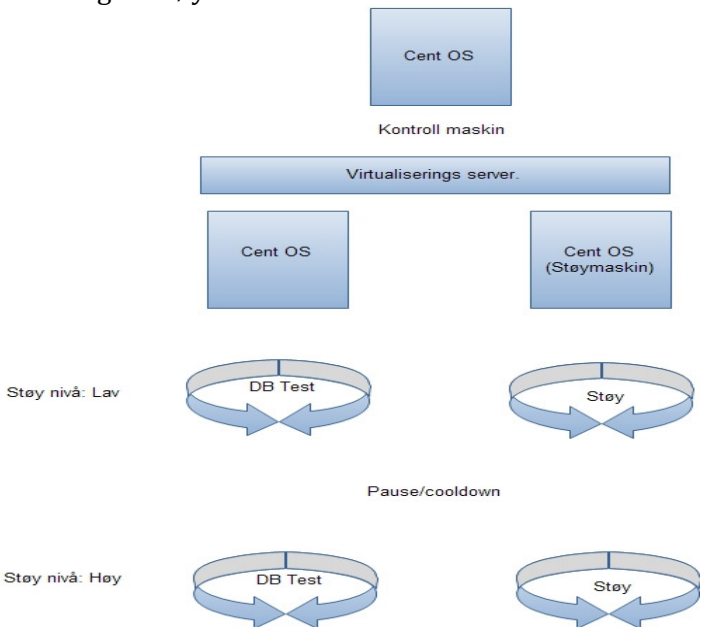
Installerte programmer for mål maskin
Mysql
Mysql client
Mysql skript

Generering av belastning
Følgende skript lokalt på maskinen som kjører av database test og støy på støymaskin.
testMysqlSenario2.pl
stoyKlient.conf
stoyKlient.pm

Grader av last på målmaskin
Det benytte Mysql skript til generering av belastning, det finnes ingen mulighet for å endre graden av denne belastningen.

Varighet
Ingen varighet satt. Mysql skript skal få kjøre til den er ferdig. Mysql test skriptet starter med å kjøre en gang.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last. Men det vil være den gjennomsnittlige belastning maskinen genererer i løp av perioden den skal lage støy.
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Ved måling av database server vil skriptet for generering av last ta tiden det tar å utføre spørringene.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Total tid
- Operasjoner i sekundet
- Tid i user og kernel space
- Tid brukt på spesifikke tester
- Antall feilede tilkoblinger

DATABASE SENARIO 3: INTERVALLER OG SAMTIDIG

Test type
Mysql database server

Mål med test
Belaste målmaskin med rollen som database server med en gittmysql test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av test skript. Og som avlutting kjøre test skriptet samtidig som det støyes på maskiner rundt. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser når andre maskiner har tilgang på dem.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Mysql
Mysql client
Mysql skript

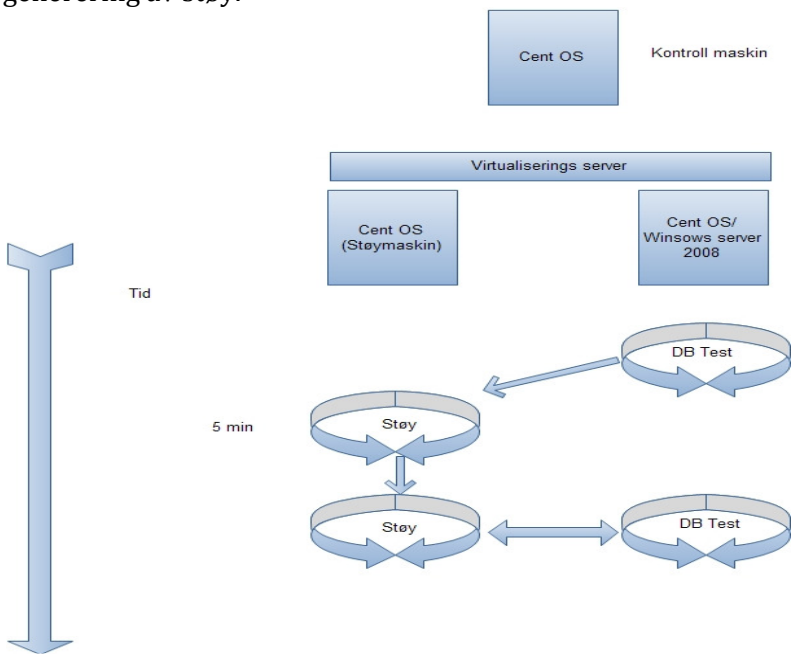
Generering av belastning
Følgende skript lokalt på maskinen som kjører av database test og støy på støymaskin.
testMysqlKontroll.pl
stoyKlient.conf
stoyKlient.pm

Grader av last på målmaskin
Det benytte Mysql skript til generering av belastning, det finnes ingen mulighet for å endre graden av denne belastningen.

Varighet
Ingen varighet satt. Mysql skript skal få kjøre til den er ferdig. Mysql test skriptet starter med å kjøre en gang. For at så skal støyes i ett intervall på 5 minutter før det kjøres en ny test samtidig som støymaskinen støyer.

Oppsett av maskiner rundet testen på virtualisers server

Under denne testen vil det være satt opp en maskin rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen

Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Hvordan lasten skal genrers

Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy

Ved måling av selve mysql database serveren benytte skript for generering og samling av data.

Måleverdier av betydning

Følgende verdier er av betydning for denne testen.

- Total tid
- Operasjoner i sekundet
- Tid i user og kernel space
- Tid brukt på spesifikke tester
- Antall feilede tilkoblinger

E-POST

E-POST SENARIO: BARE-METALL V.S VIRTUALISERING

Test type
Epost server

Mål med test
Finne forskjellen fra ”bare metall” til virtualisering av epost server.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren.
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer.

Installerte programmer for mål maskin
Dovecot(IMAP og POP)
Postfix(SMTP)

Generering av belastning
Følgende skript lokalt på maskinen som kjører av e-post test..
imapKlinetScript.pl
lagEpostadr.pl
sendEpost.pl
epostKontroll.pl

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører til testen er ferdig

Oppsett av maskiner rundet testen på virtualisers server
Ingen andre maskiner. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen
Ingen andre maskinger. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Måleverktøy
Ved testing av epost server vil skriptet for generering av last ta tiden det tar å utføre handlinger og hvor mengden data som er behandlet.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall handlinger
- Gjennomsnittlig spørringer i skundet
- Menge data behandlet

E-POST SENARIO 1: INTERVALLER

Test type
Epost server.

Mål med test
Belaste målmaskin med rollen som e-post server med test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av skriptene , for så utfør en ny e-post test for å se hvordan dette påvirker test resultat. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer.

Installerte programmer for mål maskin
Dovecot(IMAP og POP)
Postfix(SMTP)

Generering av belastning
Følgende skript lokalt på maskinen som kjører av e-post test og støy.
imapKlinetScript.pl
lagEpostadr.pl
sendEpost.pl
epostTestSenario1.pl
stoyKlint.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

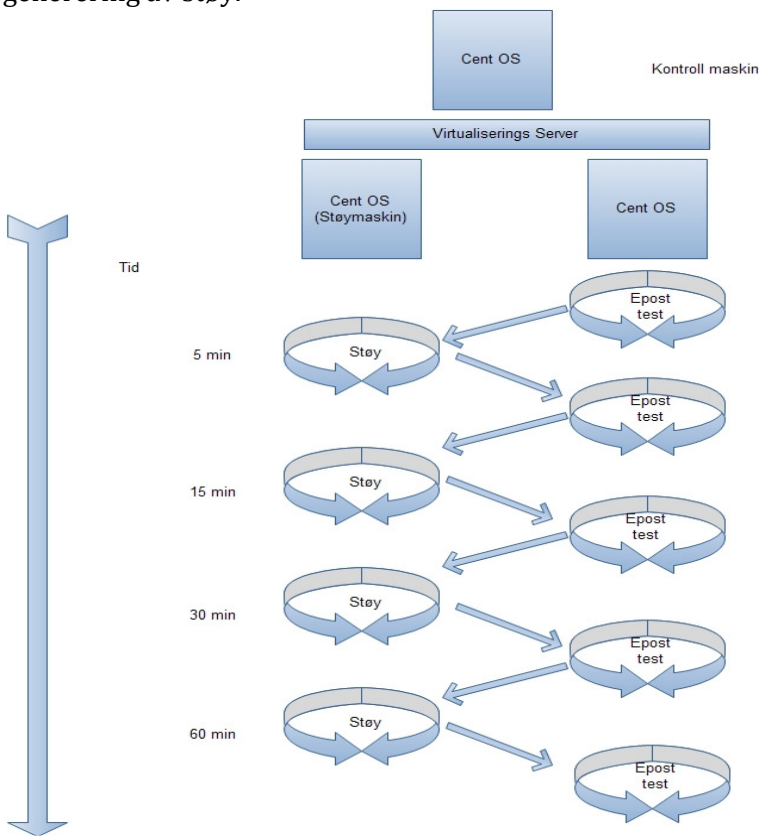
Varighet
Testen vil bli delt inn i 4 intervaller.

- 10 min
- 15 min
- 30 min
- 60 min

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Imellom hvert av disse intervallene vil det kjøres e-post tester. Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server

Under denne testen vil det være satt opp to maskiner rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen

Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.

- Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt.
- Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt.
- Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt.
- Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt.

Måleverktøy

Ved testing av epost server vil skriptet for generering av last ta tiden det tar å utføre handlinger og hvor mengden data som er behandlet.

Måleverdier av betydning

Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall handlinger
- Gjennomsnittlig spørringer i skundet
- Menge data behandlet

E-POST SENARIO 2: STØY SAMTIDIG

Test type
Epost server.

Mål med test
Belaste målmaskin med rollen som e-post server med spørringer gjennom skript samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer.

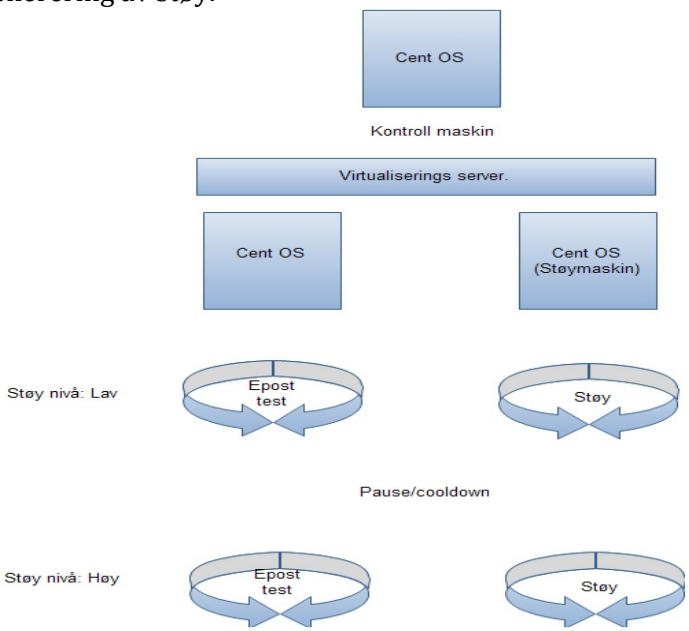
Installerte programmer for mål maskin
Dovecot(IMAP og POP)
Postfix(SMTP)

Generering av belastning
Følgende skript lokalt på maskinen som kjører av e-post test og støy.
imapKlinetScript.pl
lagEpostadr.pl
sendEpost.pl
epostTestSenario2.pl
stoyKlint.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet på test skriptet, dette vil kjøre til testen er ferdig. Det er sett en pause mellom hver test på 5 minutter.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last. Men det vil være den gjennomsnittlige belastning maskinen genererer i løp av perioden den skal lage støy.
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Ved testing av epost server vil skriptet for generering av last ta tiden det tar å utføre handlinger og hvor mengden data som er behandlet.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.
- Tiden det tar å utføre X antall handlinger
- Gjennomsnittlig spørringer i skundet
- Menge data behandlet

E-POST SENARIO 3: INTERVALLER OG SAMTIDIG

Test type
Epost server

Mål med test
Belaste målmaskin med rollen som e-post server ved kjøring av skript som utfører spørringer, samt generere støy på maskiner rundt målmaskinen i intervaller mellom e-post tester.
For så kjøring av støy maskin i en gitt periode før en test starter samtidig som støymaskinen kjører.
Dette for å se hvordan testen påvirkes av at en annen maskin har kontroller over resursene før og under kjøring av test.

Operativsystem for test
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer.

Installerte programmer for mål maskin
Dovecot(IMAP og POP)
Postfix(SMTP)

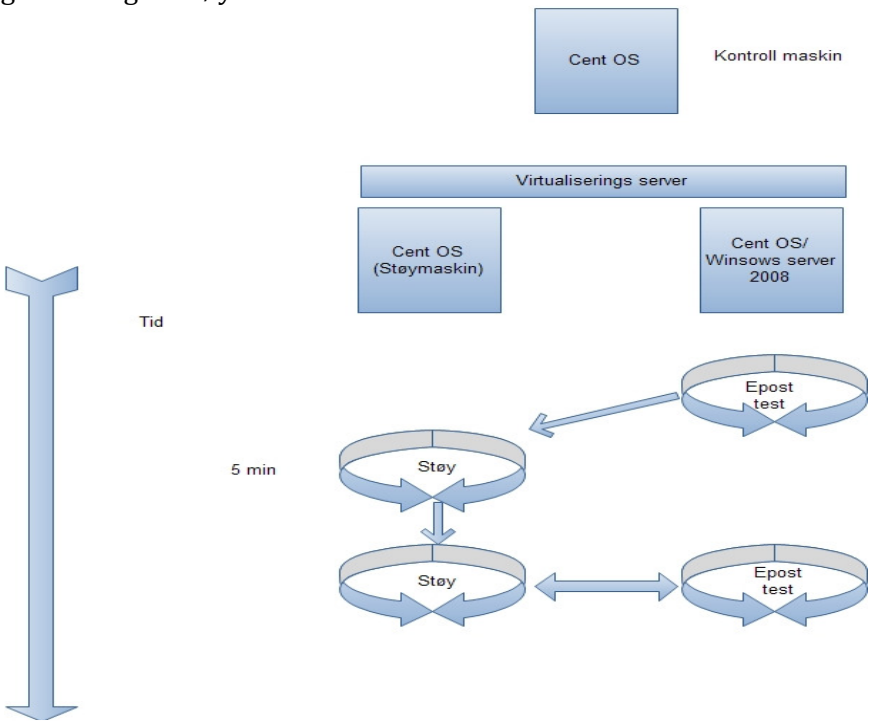
Generering av belastning
Følgende skript lokalt på maskinen som kjører av e-post test og støy.
imapKlinetScript.pl
lagEpostadr.pl
sendEpost.pl
epostTestKontroll.pl
stoyKlint.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet på test skriptet, dette vil kjøre til testen er ferdig.

Oppsett av maskiner rundet testen på virtualisers server

Under denne testen vil det være satt opp en maskin rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen

Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Hvordan lasten skal genrers

Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy

Ved testing av epost server vil skriptet for generering av last ta tiden det tar å utføre handlinger og hvor mengden data som er behandlet.

Måleverdier av betydning

Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall handlinger
- Gjennomsnittlig spørringer i skundet
- Menge data behandlet

FTP

FTP SENARIO: BARE-METALL V.S VIRTUALISERING

Test type
FTP server

Mål med test
Finne forskjellen fra ”bare metall” til virtualisering av FTP server.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Vs FTP D
Skript for generering av filer(filene benyttes til å utføre testene).

Generering av belastning
Følgende skript lokalt på maskinen som kjører av FTP test.
ftpKlient.pl
ftpLagfiler.pl

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett operasjoner som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle operasjoner før testen er ferdig.

Oppsett av maskiner rundet testen på virtualisers server
Ingen andre maskiner. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen
Ingen andre maskinger. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Måleverktøy
Ved testing av FTP server vil skriptet måle og logge verdier av betydning.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre hele testen
- MB/S ved overføring
- Spesifikke tidsmåler for forskjellige typer jobbe
- Mengde data overført

FTP SENARIO 1: INTERVALLER

Test type
FTP server

Mål med test
Belaste målmaskin med rollen som FTP server med test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av skriptene , for så utfør en ny FTP test for å se hvordan dette påvirker test resultat. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Vs FTP D
Skript for generering av filer(filene benyttes til å utføre testene).

Generering av belastning
Følgende skript lokalt på maskinen som kjører av FTP test.
ftpKlient.pl
ftpLagfiler.pl
sterFtpTestSenario1.pl
stoyKlinent.conf
stoyKlinent.pm

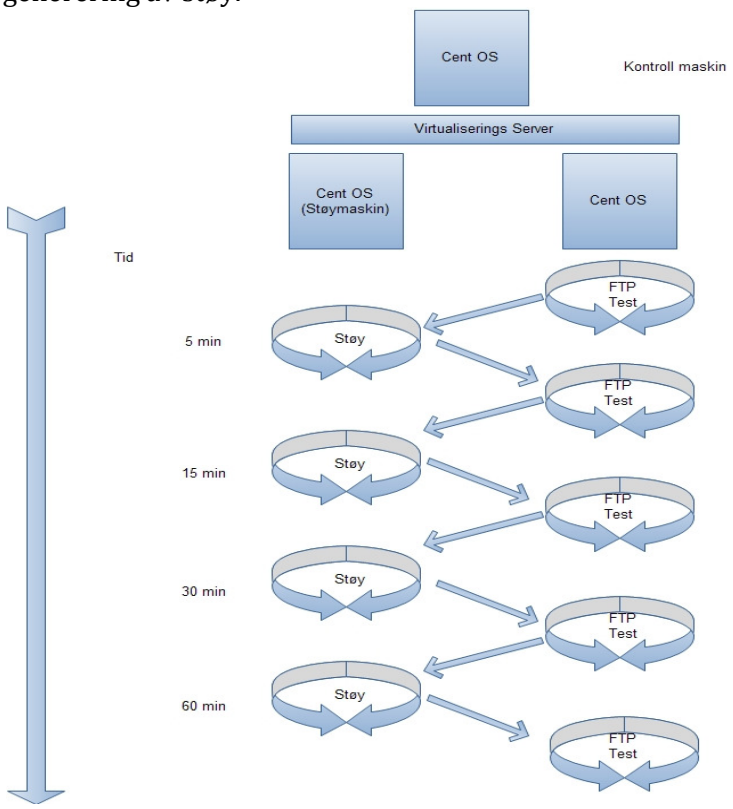
Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett operasjoner som kjøres.

Varighet
Testen vil bli delt inn i 4 intervaller.

- 10 min
- 15 min
- 30 min
- 60 min

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Mellom hver av disse intervallene vil det kjøres FTP tester. Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.

- Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt
- Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt
- Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt

Last som skal generer av maskiner rundt målmaskinen
Støy genereres ved kjøring av støy skript på støy maskinen.

Måleverktøy
Ved testing av FTP server vil skriptet måle og logge verdier av betydning.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre hele testen
- MB/S ved overføring
- Spesifikke tidsmåler for forskjellige typer jobbe
- Mengde data overført

FTP SENARIO 2: STØY SAMTIDIG

Test type
FTP server

Mål med test
Belaste målmaskin med rollen som FTP server med operasjoner gjennom ett skript samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

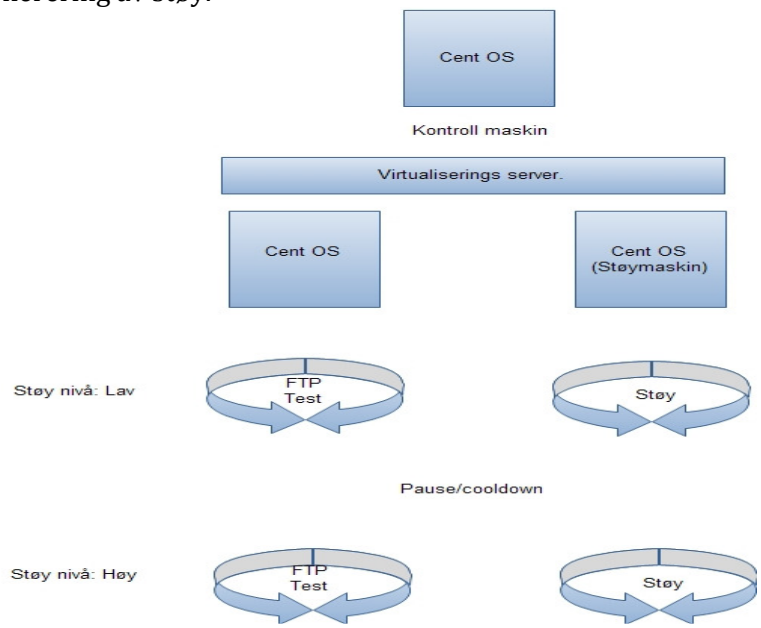
Installerte programmer for mål maskin
Vs FTP D
Skript for generering av filer(filene benyttes til å utføre testene).

Generering av belastning
Følgende skript lokalt på maskinen som kjører av FTP test.
ftpKlient.pl
ftpLagfiler.pl
sterFtpTestSenario2.pl
stoyKlinent.conf
stoyKlinent.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett operasjoner som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle spørring mot FTP serveren før testen er ferdig. Det er satt 5 minutter pause mellom testene.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Last som skal generer av maskiner rundt målmaskinen
Støy genereres ved kjøring av støy skript på støy maskinen.

Måleverktøy
Ved testing av FTP server vil skriptet måle og logge verdier av betydning.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre hele testen
- MB/S ved overføring
- Spesifikke tidsmåler for forskjellige typer jobbe
- Mengde data overført

FTP SENARIO 3: INTERVALLER OG SAMTIDIG

Test type
FTP server

Mål med test
Belaste målmaskin med rollen som FTP server ved kjøring av skript som FTP relatert arbeid, samt generere støy på maskiner rundt målmaskinen i intervaller mellom FTP tester.
For så kjøring av støy maskin i en gitt periode før en test starter samtidig som støymaskinen kjører.
Dette for å se hvordan testen påvirkes av at en annen maskin har kontroller over resursene før og under kjøring av test.

Operativsystem for test
Cent OS 5.4 rett på virtualiserings serveren
Virtuell Cent OS 5.4 installert på VMware esxi og XenServer

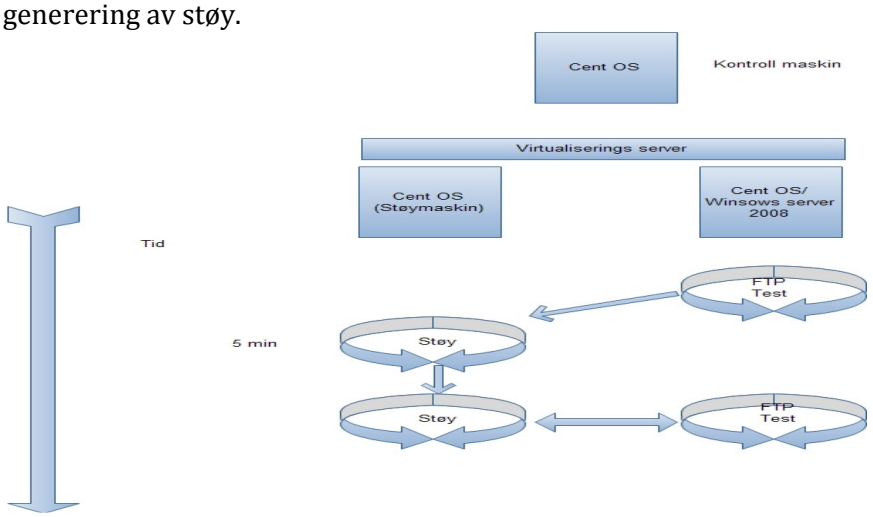
Installerte programmer for mål maskin
Vs FTP D
Skript for generering av filer(filene benyttes til å utføre testene).

Generering av belastning
Følgende skript lokalt på maskinen som kjører av FTP test.
ftpKlient.pl
ftpLagfiler.pl
stoyKlinent.conf
stoyKlinent.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett operasjoner som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører alle operasjoner til det er ferdig. Det er satt varighet på 5 minutter på støyperioden som skal kjøre før og under andre FTP test.

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp to maskiner rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen
Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Last som skal generer av maskiner rundt målmaskinen
Støy genereres ved kjøring av støy skript på støy maskinen.

Måleverktøy
Ved testing av FTP server vil skriptet måle og logge verdier av betydning.

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre hele testen
- MB/S ved overføring
- Spesifikke tidsmåler for forskjellige typer jobbe
- Mengde data overført

TERMINAL

TERMINAL SENARIO: BARE-METALL V.S VIRTUALISERING

Test type
Terminal server

Mål med test
Finne forskjellen fra ”bare metall” til virtualisering av terminal server.

Operativsystem for test
Windows server 2008 rett på virtualiserings serveren
Virtuell Windows server 2008 installert på VMware esxi og XenServer

Installerte programmer for mål maskin
Active Directory
Terminal server service

Generering av belastning
Følgende skript lokalt på maskinen som kjører av Terminal server test.
tsTester.pl

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med oppkoblinger som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører x antall oppkoblinger mot terminal serveren til den er ferdig

Oppsett av maskiner rundet testen på virtualisers server
Ingen andre maskiner. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Last som skal generer av maskiner rundt målmaskinen
Ingen andre maskinger. Testen utføres først ”bare metall” og deretter på en enkelt virtuell maskin på virtualiserings servene.

Måleverktøy
Skript for terminal server testing tar seg av målingen

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall oppkoblinger
- Tiden hver oppkobling tok
- Antall feilede oppkoblinger
- Tid for feilet oppkobling

TERMINAL SENARIO 1: INTERVALLER

Test type
Terminal server.

Mål med test
Belaste målmaskin med rollen som terminal server med test skript, samt generere støy på maskiner rundt målmaskinen i intervaller mellom kjøring av skriptene , for så utfør en ny terminal test for å se hvordan dette påvirker test resultat. Dette er spesielt med tanke på hvordan målmaskinen får tildelt resurser tilbake etter andre maskiner har belastet serveren.

Installerte programmer for mål maskin
Active Directory
Terminal server service

Generering av belastning
Følgende skript lokalt på maskinen som kjører av Terminal server test og støy.
tsTestSenario1.pl
stoyKlient.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

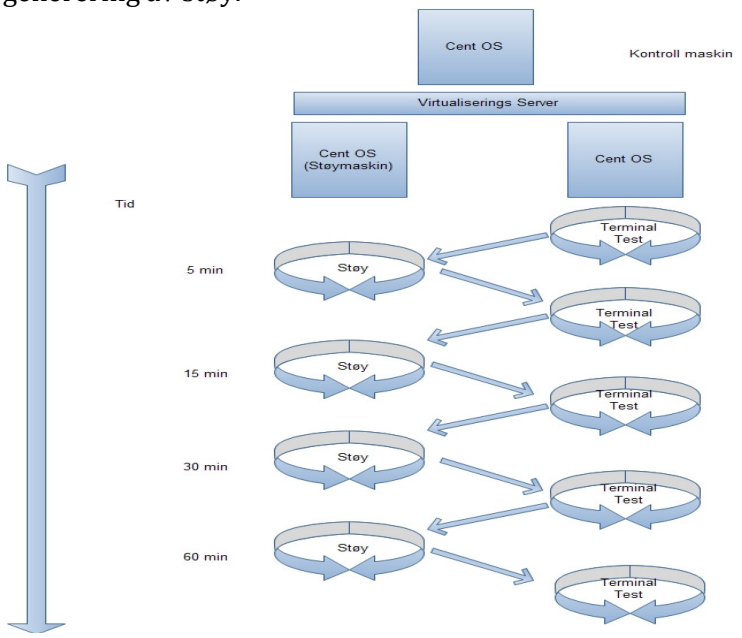
Varighet
Testen vil bli delt inn i 4 intervaller.

- 10 min
- 15 min
- 30 min
- 60 min

Der hvert intervall representer tiden det vil forgå støy på maskinene rundt målmaskinen. Imellom hver av disse intervallene vil det kjøres terminal tester. Det vil i tillegg være en lite pause på 2 minutter etter hver test før ett nytt intervall med støy starter.

Oppsett av maskiner rundet testen på virtualisers server

Under denne testen vil det være satt opp en maskin rundt målmaskinen. AV typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen

Lasten som skal genrers på disse vil ikke være konstant i det tidsintervallet de skal generer last.
Ved Intervall 1: Ca 30% av det Cent maskinene klarer lokalt.
Ved Intervall 2: Ca 30% av det Cent maskinene klarer lokalt.
Ved Intervall 3: Ca 80% av det Cent maskinene klarer lokalt.
Ved Intervall 4: Ca 80% av det Cent maskinene klarer lokalt.

Hvordan lasten skal genrers

Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy

Skript for terminal server testing tar seg av målingen

Måleverdier av betydning

Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall oppkoblinger
- Tiden hver oppkobling tok
- Antall feilede oppkoblinger
- Tid for feilet oppkobling

TERMINAL SENARIO 2: SAMTIDIG STØY

Test type
Terminal server

Mål med test
Belaste målmaskin med rollen som terminal server med x antall oppkoblinger gjennom ett skript samtidig som maskiner rundt målmaskinen generer støy av forskjellig grad.

Operativsystem for test
Virtuell Windows server 2008 installert på VMware esxi og XenServer

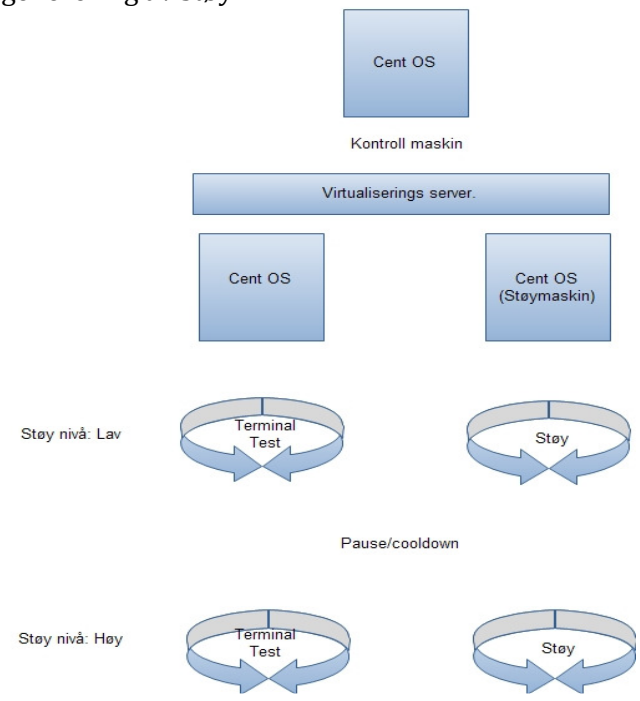
Installerte programmer for mål maskin
Active Directory
Terminal server service

Generering av belastning
Følgende skript lokalt på maskinen som kjører av Terminal server test og støy.
tsTestSenario2.pl
stoyKlient.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører x antall oppkoblinger mot terminal serveren til den er ferdig

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskine sammen med målmaskinen. Av typen Cent OS for generering av støy.



Last som skal generer av maskiner rundt målmaskinen
Ved støy nivå lav skal belastning ligge på rundt 30%.
Ved støy nivå høy skal belastningen ligge på rundt 80%.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Skript for terminal server testing tar seg av målingen

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall oppkoblinger
- Tiden hver oppkobling tok
- Antall feilede oppkoblinger
- Tid for feilet oppkobling

TERMINAL SENARIO 3: INTERVALL OG SAMTIDIG STØY

Test type
Terminal server

Mål med test
Belaste målmaskin med rollen som terminal server ved kjøring av skript som utfører oppkoblinger, samt generere støy på maskiner rundt målmaskinen i intervaller mellom terminal testene.
For så kjøring av støy maskin i en gitt periode før en test starter samtidig som støymaskinen kjører.
Dette for å se hvordan testen påvirkes av at en annen maskin har kontroller over resursene før og under kjøring av test.

Operativsystem for test
Virtuell Windows server 2008 installert på VMware esxi og XenServer

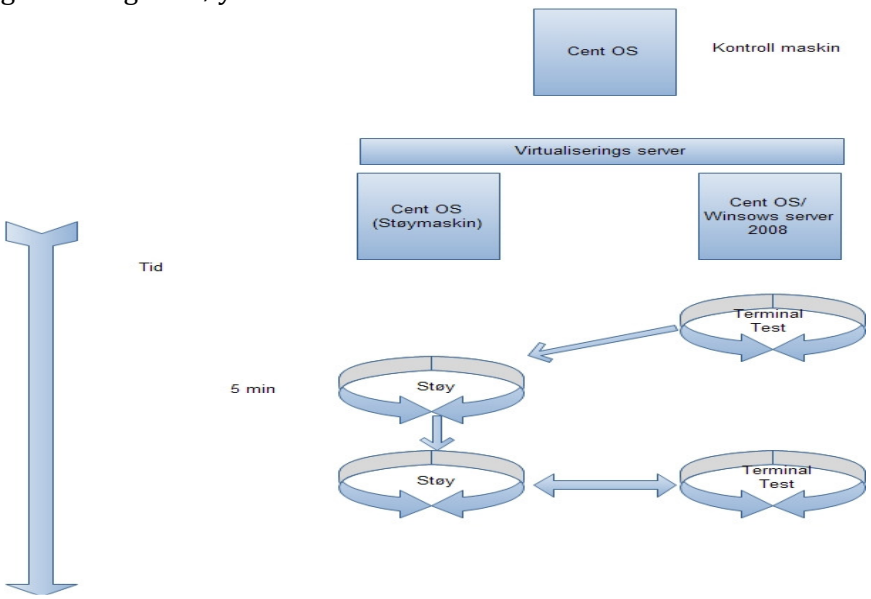
Installerte programmer for mål maskin
Active Directory
Terminal server service

Generering av belastning
Følgende skript lokalt på maskinen som kjører av Terminal server test og støy.
tsTestSenario3.pl
stoyKlient.conf
stoyKlint.pm

Grader av last på målmaskin
Generering av last skjer gjennom kjøring av perl skript. Dette skriptet har en fast sett med spørringer som kjøres.

Varighet
Ingen varighet satt. Skriptet kjører x antall oppkoblinger mot terminal serveren til den er ferdig

Oppsett av maskiner rundet testen på virtualisers server
Under denne testen vil det være satt opp en maskin rundt målmaskinen. Av typen Cent OS for generering av støy.



Last som skal genereres av maskiner rundt målmaskinen
Denne testen kan kjøres i to omganger. I føreste omgang vil belastningen på støymaskinen ligge på 30%. I andre omgang vil støy nivået økes til 80%. Dette kan reguleres etter eget ønske ved bruk av støy skript.

Hvordan lasten skal genrers
Støy genrers gjennom kjøring av støy skript på støy maskinen.

Måleverktøy
Skript for terminal server testing tar seg av målingen

Måleverdier av betydning
Følgende verdier er av betydning for denne testen.

- Tiden det tar å utføre X antall oppkoblinger
- Tiden hver oppkobling tok
- Antall feilede oppkoblinger
- Tid for feilet oppkobling

G. SKRIPT

APACHE

abTeter.pl

```
#!/usr/bin/perl

#Path to ab test
use constant abKatalog => "/usr/bin/ab";

#Ab test maa vaere installert
if (-e abKatalog){
	($webServer,$webfil,$maksKlienter,$klientHopp,$maksSporring,$sporringHopp) =
@ARGV;


	my $filnavn;
	#Okende belasting paa klienter
	for(my $i = 1; $i <= $maksKlienter; $i+=$klientHopp){
		#Okende belasting paa antall side sporringer per klient
			for(my $j = $sporringHopp; $j <= $maksSporring;
$j+=$sporringHopp){
				print "Klienter: $i - AntallSamtidigeSporringer: $j\n";

				system("ab -n $j -c $i http://$webServer/$webfil");
			}


	# Vi onkser å altid teste med en oppkobling forst og ha riktig stigning
	etterpaa
	# eks ved hopp 2 ( 1, 2, 4, 6, 8 osv)
	if($i == 1 && $klientHopp != 1){$i=0;}
	}

}else{
	print "Error: Apache må installeres for aa faa ab (apache benchmark)!\n";
}
```

abTestSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

# Eks:
my ($resultatFil,
$webServer,$webfil,$maksKlienter,$klientHopp,$maksSporring,$sporringHopp) = @ARGV;


#Hent inn konfigurasjon fra fil for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Testlog text skriver
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
```

```
	close RESULTAT;
}

#Slett logfil dersom den finnes fra for
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#testkjoring uten paavirking av stoy
skrivTilResultat "START TEST\n";
skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
#Start test
system("./abTester.pl $webServer $webfil $maksKlienter $klientHopp $maksSporring
$sporringHopp >> $resultatFil");
skrivTilResultat "SLUTT TEST\n";
skrivTilResultat "\n";

print "$pauseTid sek pause for neste kjoring\n";
#Pause for neste test
sleep($pauseTid);

#Gaa gjennom stoy verdier for testing
for(my $i = 0; $i<@stoyString; $i++){
	#Kjorelende i tid for stoyscript
	for(my $t = 0; $t<@stoyKjoreTid; $t++){

			print "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
			#Start stoy
			kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t], $stoyString[$i],
$stoySleep[$i];
			print "Stoy $stoyProsent[$i] ferdig\n";
			skrivTilResultat "START TEST\n";
			skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
			#Start test
			system("./abTester.pl $webServer $webfil $maksKlienter
$klientHopp $maksSporring $sporringHopp >> $resultatFil");
			skrivTilResultat "SLUTT TEST\n";
			skrivTilResultat "\n";

			#siste Runde trenge ingen pause
			if($t != (@stoyKjoreTid-1)){
				print "$pauseTid sek pause for neste kjoring\n";
				#Ta pause
				sleep($pauseTid);
			}

		}

}
```

abTestSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
use threads::shared;

my ($resultatFil,
$webServer,$webfil,$maksKlienter,$klientHopp,$maksSporring,$sporringHopp) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
```

```
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#kjoring uten paavirking av stoy
skrivTilResultat "START TEST\n";
skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
#Start test
system("./abTester.pl $webServer $webfil $maksKlienter $klientHopp $maksSporring
$sporringHopp >> $resultatFil");
skrivTilResultat "SLUTT TEST\n";
skrivTilResultat "\n";

print "$pauseTid sek pause for neste kjoring\n";
#Pause mellom hver test
sleep($pauseTid);

#Kjor alle stoybelastninger sammen med test
for(my $i = 0; $i<@stoyString; $i++){
		skrivTilResultat "START TEST\n";
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";
		#Egen traad for fjernstyring av stoy
		my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i], "evig",
$stoyString[$i], $stoySleep[$i];

		#Start test
		system("./abTester.pl $webServer $webfil $maksKlienter $klientHopp
$maksSporring $sporringHopp >> $resultatFil");

		#Test er ferdig. Stop stoy
		my $stopStoyTraad = new threads \&stopStoy;
		$stopStoyTraad->join;

		#Stoy har avsluttet
		$SamtidigStoy->join;
		skrivTilResultat "SLUTT TEST\n";
		skrivTilResultat "\n";

		#Dersom siste runde av hele testen. ingen pause
		if($i!=(@stoyKjoreTid-1)){
			print "$pauseTid sek pause for neste kjoring\n";
			#Pause mellom hver igjennomfort test
			sleep($pauseTid);
		}

		}
```

E-POST

epostKontroll.pl

```
#!/usr/bin/perl

use Time::HiRes;

#Pause mellom SMTP sending og IMAP Klient oppkobling
my $pause = 60;

#Oppstart parametere brukt ved kjøring av scriptet
#eks: drift.no epost.drift.no epost.drift.no bruker passord 3000 30000 10 10
my ($domene, $imapserver, $smtpServer, $brukernavn, $passord, $antMld,
$mldStr,$antMapper, $antbrukere) = @ARGV;

#Start tid - Totaltid for SMTP og IMAP kjoring
my $startTotal = [ Time::HiRes::gettimeofday( ) ];

print "SMTP TEST: Mld Storrelse: $mldStr - Antall brukere: $antbrukere - Ant.Mld
per bruker: $antMld - Mld.Totalt: ".($antMld*$antbrukere)." \n";

#Start tid - Kjortid for SMTP test
my $start = [ Time::HiRes::gettimeofday( ) ];

#Send Epost til alle brukere samtidig
system("./sendEpost.pl $brukernavn $domene $smtpServer $antMld $mldStr
$antbrukere");
#Tidsforbruk for SMTP Sending
my $ferdig = Time::HiRes::tv_interval( $start );
print "SMTP: $ferdig total Sek - Ca "; printf "%.3f epost mottat i sek\n",
(($antMld*$antbrukere)/$ferdig);

sleep($pause); #La Eposten bli fanget opp av dovecot og klar for IMAP Test

print "IMAP TEST: Antall brukere: $antbrukere - Antall Mapper: $antMapper\n";
#Starter IMAP Klient test
system("./imapKlientScript.pl $imapserver $brukernavn $passord $antMapper
$antbrukere");
#Maaling av totaltid ferdig
my $ferdigTotal = Time::HiRes::tv_interval( $startTotal );

#Trekk fra pausetiden
$ferdigTotal = $ferdigTotal - $pause;
print "Totaltid: $ferdigTotal sek\n";
print "\n";
```

epostTestSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

# Eks: ./epostTestSenario1.pl test.log drift.no epost.drift.no epost.drift.no drift
31pass+tull 2 2
my ($resultatFil, $domene, $imapserver, $smtpServer, $brukernavn, $passord,
```

```
$antMld, $mldStr,$antMapper, $antbrukere) = @ARGV;


#Hent inn konfigurasjon fra fil for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Testlog text skriver
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Slett log fil om den finnes
if(-e $resultatFil){
	unlink($resultatFil);
}

#To testkjoringer uten paavirking av stoy
for(my $i = 0; $i<2; $i++){
	print "StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Kjor testen
	system("./epostKontroll.pl $domene $imapserver $smtpServer $brukernavn
$passord $antMld $mldStr $antMapper $antbrukere >> $resultatFil");
	skrivTilResultat "\n";
	print "$pauseTid sek pause for neste kjoring\n";
	sleep($pauseTid);
}

#Kjor hver stoymengde
for(my $i = 0; $i<@stoyString; $i++){
	#Kjor stoy i en bestemt tidslengde
	for(my $t = 0; $t<@stoyKjoreTid; $t++){

		print "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
		#Kjor stoy
		kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t], $stoyString[$i],
$stoySleep[$i];
		print "Stoy $stoyProsent[$i] ferdig\n";
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";

		#Kjor test
		system("./epostKontroll.pl $domene $imapserver $smtpServer
$brukernavn $passord $antMld $mldStr $antMapper $antbrukere >> $resultatFil");
		skrivTilResultat "\n";

		#Dersom siste runde av hele testen. ingen pause
		if($i!=(@stoyString-1)){
			print "$pauseTid sek pause for neste kjoring\n";
			#Pause mellom hver igjennomfort test

			sleep($pauseTid);
		}

	}

}
```

epostTestSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stykonfig og nettverks styring
use stoyKlient;
use threads;
use threads::shared;


# Eks: ./epostTestSenario2.pl test.log drift.no epost.drift.no epost.drift.no drift
31pass+tull 2 2
my ($resultatFil, $domene, $imapserver, $smtpServer, $brukernavn, $passord,
$antMld, $mldStr,$antMapper, $antbrukere) = @ARGV;

# Hent konfigurasjon fra stoyKlient.conf
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett resultat fil om den finnes fra for
if(-e $resultatFil){
	unlink($resultatFil);
}

#Text til resultatfil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#To kjoringer uten paavirking av stoy
for(my $i = 0; $i<2; $i++){
	print "StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Start test
	system("./epostKontroll.pl $domene $imapserver $smtpServer $brukernavn
$passord $antMld $mldStr $antMapper $antbrukere >> $resultatFil");
	skrivTilResultat "\n";
	print "$pauseTid sek pause for neste kjoring\n";
	#Pause mellom hver test
	sleep($pauseTid);
}


#Kjor alle stoybelastninger sammen med test
for(my $i = 0; $i<@stoyString; $i++){
		print "StoyProsent: $stoyProsent[$i] - StoyKjoreTid: SAMTIDIG\n";

		#Egen traad for fjernstyring av stoy
		my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i], "evig",
$stoyString[$i], $stoySleep[$i];
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";

		#Start test
		system("./epostKontroll.pl $domene $imapserver $smtpServer $brukernavn
$passord $antMld $mldStr $antMapper $antbrukere >> $resultatFil");

		#Test er ferdig. Stop stoy
		my $stopStoyTraad = new threads \&stopStoy;
		$stopStoyTraad->join;

		#Stoy har avsluttet
```

```
                $SamtidigStoy->join;
                skrivTilResultat "\n";

                #Dersom siste runde av hele testen. ingen pause
                if($i!=(@stoyString-1)){
                        print "$pauseTid sek pause for neste kjoring\n";
                        #Pause mellom hver igjennomfort test
                        sleep($pauseTid);
                }

}
```

imapKlientScript.pl

```
#!/usr/bin/perl

#IMAP biblotek filer
use Net::IMAP::Simple;
use threads;
use threads::shared;
use Time::HiRes;

#Navn for opprettet mappe
$MAPPENAVN = "Mappe";
#Navn for mappe for innkommende epost
$INBOX = "INBOX";

#Delte variabler
#Storrelse paa alle brukernes epost meldinger
my $totalstorrelse : shared = 0;
#Total mengde epost til alle brukere
my $totalEpost : shared = 0;

#Oppretting av ny epost mappe
sub lagMapper
{
  my ($antall) = @_;
  if($antall > 0){
         for(my $i = 1; $i<=$antall; $i++){
             if($imap->create_mailbox( $MAPPENAVN.$i )){
              #Få med mappen ved IMAP synkronisering
              $imap->folder_subscribe($MAPPENAVN.$i);
           }

         }
       }
}

#Simulering av en Epost klient
sub jobbMedInbox
{

  my $mappestorrelse = 0;

  my ($mappenavn, $antNyeMapper, $brukernavn) = @_;

  #Åpne mappen
  my $antMld = $imap->select($mappenavn);

  #Total sum av epost til alle brukere
  lock($totalEpost);
  $totalEpost += $antMld;

  #Hvilken mappe står jeg i
```

```
  my $mappeNaa = $imap->current_box;

  #Hvilken funksjoner er i mappen jeg står i
  my $mappeFuksjoner = $imap->flags;

  #Gaa igjennom hver melding
  for(my $i = 1; $i<=$antMld; $i++){

       #Se statusen på om meldingen er lest eller ikke
   if($imap->seen($i)){}

       #Total meldings størrelse
       $meldingsStorrelse = $imap->list($i);
       $mappestorrelse += $meldingsStorrelse;

       #Henter HEAD datainnhold til gitt melding
       my $emner = $imap->top( $i );


       if($INBOX == $mappenavn && $antNyeMapper>0){
              #Spre meldingene mellom mappene
              $tilmappe = $i % $antNyeMapper + 1;
              #kopier melding til neste mappe
              $imap->copy( $i, $MAPPENAVN.$tilmappe)
       }

       #Last ned gjeldende melding
       my $mail = $imap->get( $i );
  }

  #Slett alle meldinger i index. Start med siste til forste
  while($antMld != 0){
       $imap->delete( $antMld );
       $antMld--;
  }
  #Tommer papirkurven for slettet mail
  $imap->expunge_mailbox($mappenavn);

       #Slett alle mapper med epost som er opprettet
       for(my $j = 1; $j<=$antNyeMapper; $j++){
              $imap->folder_unsubscribe($MAPPENAVN.$j);
              $imap->delete_mailbox($MAPPENAVN.$j);
       }

       #Oppdater den totale epost storrelse
       lock($totalstorrelse);
       $totalstorrelse += $mappestorrelse;
}

#Koble til epost konto
sub kobleTil{

       my ($imapserver, $brukernavn, $passord, $antNyeMapper) = @_;
       #Koble til IMAP Server
       $imap = Net::IMAP::Simple->new($imapserver) ||
           die "Fikke koblet til IMAP server\n";


       if(!$imap->login($brukernavn,$passord)){
          print "Fikk ikke logget på IMAP server: " . $imap->errstr . "\n";
          exit(64);
       }

       #Hvilken mappe står jeg i
       $arbMappe = $imap->current_box;

       #Hva kan jeg gjør i mappen
       $imapFuksjoner = $imap->flags;
```

```
	#Hvilken andre mapper finnes
	@mapper = $imap->mailboxes;

	#Lag nye mapper i rot
	lagMapper($antNyeMapper);

	#Åpner gitt mappe. Dersom INDEX vil innhold spres over til mapper laget i
lagMapper
	jobbMedInbox($INBOX, $antNyeMapper, $brukernavn);

	#Log ut
	$imap->quit;
}

#Start parametere
my ($imapserver, $brukernavn, $passord, $antNyeMapper,$antKontoer) = @ARGV;

#Hent start tid
my $start = [ Time::HiRes::gettimeofday( ) ];
for($i=1; $i<=$antKontoer; $i++){
	#Brukernavn
	$bruker = $brukernavn.$i;
	#Oppkobling til epost
	$thr[$i] = new threads \&kobleTil, $imapserver, $bruker, $passord,
$antNyeMapper;
}

#Vent paa at alle epost er ferdig
for($j=1; $j<=$antKontoer; $j++){
	$thr[$j]->join;
}

#Kjoring av hele testen er ferdig
my $ferdig = Time::HiRes::tv_interval( $start );
#Skriv resultat
print "IMAP: $ferdig sek total  - "; printf "%.3f epost les og behandlet i
sek\n",($totalEpost/$ferdig);
print "Alle bruker epost i kb ".($totalstorrelse/1024)." (med epostheadinfo)\n";
print "Totalt antMld behandlet: $totalEpost\n";
```

lagEpostadr.pl

```
#!/usr/bin/perl

use Shell;

#Se om brukeren allerede finnes
sub sokEtterBrukernavn{

	my ($sbrukernavn) = @_;

	my $find = $sbrukernavn."\n";
	@line = ls ("/home");


	for (@line) {
		if ($_ =~ /$find/) {
		#print $_;
		return 1;
```

```
                }
        }

        return 0;
}

#Lag passord for bruker
sub lagPassord{
        my ($setPassord) = @_;
        #perl -e 'print crypt($ARGV[0], "password")' 31pass+tull
        my $kryptPassord = crypt($setPassord, "password");
        return $kryptPassord;
}



($brukernavn,$password, $antall)= @ARGV;
        #Gaa igjennom antall nye brukere
        for($i=1;$i<=$antall;$i++){
                #Se om brukeren finnes
                if(sokEtterBrukernavn $brukernavn.$i){
                        print $brukernavn.$i." Finnes\n";

                }else{
                        #Brukeren finnes ikke.lag bruker med epost
                        print $brukernavn.$i." Finnes ikke \n";
                        #Hent kryptert passord for bruker
                        $settpass = lagPassord $password;

                        #Legg til bruker med passord
                        system("/usr/sbin/useradd -p $settpass $brukernavn$i");
                        #Opprett epost til brukeren som er laget
                        system("mkdir /home/$brukernavn$i/Maildir");
                        system("chown $brukernavn$i:$brukernavn$i
/home/$brukernavn$i/Maildir");
                        system("chmod -R 700 /home/$brukernavn$i/Maildir");


                }
}
```

sendEpost.pl

```
#!/usr/bin/perl

use Thread;
use Shell;

#Antall sendinger gjort av script
$sendere = 1;

#Script parametere fra oppstart
($brukernavn,$domene,$smtpServer,$antallMld,$MldStorrelse,$antallEpost) = @ARGV;

#For alle bruker kontoer
for($i=1; $i<=$antallEpost; $i++){
        #Send epost til brukerkonto
        $thr[$i] = new Thread \&sendEpost ,$i;
}

#Vent til alle epost er sendt
for($j=1; $j<=$antallEpost; $j++){
        $thr[$j]->join;
}
```

```
#Sending av epost via postfix smtp-source
sub sendEpost
{
	($id) = @_;
	$bruker = $brukernavn.$id;

	system("/usr/sbin/smtp-source -c -l $MldStorrelse -t $bruker@$domene -s
$sendere -m $antallMld $smtpServer > /dev/zero");
}
```

FTP

ftpKlient.pl

```
#!/usr/bin/perl
use Net::FTP;
use Time::HiRes;

#Eks: ./ftpKlient.pl 192.168.1.10 anonymous anonymous@anonymous
($server, $brukernavn, $passord) = @ARGV;

#Om ikke brukernavn og passord er satt login med anonymous
if($brukernavn == 0 && $passord == 0){
	$brukernavn = "anonymous";
	$passord = "anonymous@anonymous";
}

#Total storrelse paa alle filer
my $storrelse = 0;
#Total paa hele testen
my $totalTid = 0;

#Koble til FTP Server
$ftp=Net::FTP->new($server,Timeout=>240);
$ftp->login($brukernavn,$passord);

#List katalogen
@filListe=$ftp->dir;
#Kun filnavn
@regFiler;
#Kun Filstorrelse
@regFilerStorrelse;

#Oversikt over hver filstorrelse gruppe
#Storrelse av gjeldende filstorrelse
@filKbListe;
#Tid for Kopiering av filer i samme storrelse
@filTid;
#Total antall for filer i samme storrelse
@filAntall;
#Total storrelse for flere filer av samme storrelse
@totalKb;

#Sett inn verdier for en bestem filstorrelse
sub fyllArrays{

		my ($start, $filStorrelse, $antall) = @_;

		#Tid for kopiering
		$stop = Time::HiRes::tv_interval( $start );
		#Filstorreksen
		push(@filKbListe,$filStorrelse);
		#Totaltid for filstorrelsen
		push(@filTid,$stop);
		#Antall for fil storrelsen
		push(@filAntall,$antall);
		#Den totale filstorrelsen
		push(@totalKb,($antall*$filStorrelse));
		#Legg til tid for tid av hele testen
		$totalTid += $stop;
}

#Hent ut informasjon av filListe
foreach(@filListe) {
	($mappe,undef,undef,undef,$byte,undef,undef,undef,$filnavn) = split / +/,$_;
	#Sorter ut mapper
	$mappe = substr($mappe,0,1);
	if(!($mappe eq "d")){
```

```
                #print "$_\n";
                #Totale storrelse paa alle filer
                $storrelse += $byte;
                #Lagre filnavnet
                push(@regFiler,$filnavn);
                #Lagre storrelse paa fil i kb
                push(@regFilerStorrelse, ($byte/1024));
                }

  }

#Husk siste fil
$sisteFil="";
#Fil teller
$antall=-1;
#Start tid
$start;
for ($i = 0; $i<@regFiler; $i++){
        #Del opp informasjon fra filnavnet
        ($filnavn,$filStorrelse,$filnr) = split /-/,$regFiler[$i];
        #Finn filnr
        $filnr = substr($filnr,0,-4);
        #print "$filnavn,$filStorrelse,$filnr\n";

        #Ny filstorrelse oppdaget
        if(!($filStorrelse eq $sisteFil) && $antall != -1){
                #sleep 1;
                #print "Size: ".($regFilerStorrelse[$i-1]/1024)."\n";
                #Send med Starttid, filstorrelse og antall
                fyllArrays $start, $regFilerStorrelse[$i-1], $antall;
                #Tilbake still antallteller
                $antall=0;
        }

        # Forste kjoring?
        if($antall == -1){
                $antall=1;
        }else{
                $antall++;
        }

        #Sett ny starttid dersom det er ny filstorrelse
        if(!($filStorrelse eq $sisteFil)){
                #print "$regFiler[$i]\n"; #DEBUG
                $start = [ Time::HiRes::gettimeofday( ) ];
        }
        #Husk siste filstorrelse i neste runde
        $sisteFil = $filStorrelse;
        #Hent filen fra ftpserver
        $ftp->get($regFiler[$i]);
}
        #Legg til siste kopierings runde av filstorrelse
        fyllArrays $start, $regFilerStorrelse[@regFilerStorrelse-1], $antall;


        #Skriv ut resultater for fil kopieringen
        for($i=0; $i<@filKbListe; $i++){
                print $filKbListe[$i]."Kb - Ant:".$filAntall[$i]." -
Size:".$totalKb[$i]."kb Totalt - ".$filTid[$i]."sek -
".(($totalKb[$i]/$filTid[$i])/1024)."Mb/sek\n";
        }

        $storrelse = $storrelse/1024; #I Kb
        print "Totaltid: $totalTid - ".(($storrelse/$totalTid)/1024)."Mb/sek
Totalt\n";


        print "Mappe Storrelse: ".($storrelse / 1024)."Mb\n";
```

```
        print "Antall filer: ".@regFiler."\n";
#log ut av ftp server
$ftp->quit;
```

ftpLagfiler.pl

```
#!/usr/bin/perl

#Scriptet må kjøre som root for å kunne bruke dd og skrive til /var/ftp

use constant filnavn => "fildata";
use constant filetternavn => ".dat";
#Hvor filene skal lagres
use constant ftpMappe => "/var/ftp";

#Filstorrelse hvor hver fil i arrayen
@kbytes = (10,250,500,1000,350000,1000000);
#Antall filer av filstorresle.
@antallFiler = (1000, 250, 250, 200, 1 ,1);

#Mappe må finnes
if (-e ftpMappe){
        #Array må ha lik lengde
        if(@kbytes==@antallFiler){
                $antall = 0;
                #Hvor hver filstorrelse som skal lages
                foreach $filstorrelse(@kbytes){
                        #Lag antall filer for en filstorrelse
                        for(my $i=1; $i<=$antallFiler[$antall]; $i++){
                                $filnavn = filnavn."-".$filstorrelse."-".$i.filetternavn;
                                system("dd if=/dev/zero of=".ftpMappe."/$filnavn bs=1024
count=$filstorrelse");
                        }
                        $antall++;
                }
        }else{
                print "Error: Hver fil maa ha en storrelse og hver storrelse maa ha en
fil. Array kbytes og antallFiler har ikke lik storrelse\n";
        }

}else{
        print "Error: Er FTP server installert paa server\n";
}
```

startFtpTestSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

my ($resultatFil,$server, $brukernavn, $passord) = @ARGV;

sub hentKlokkeTid{
        my ($sek,$min,$time) = localtime(time);
        if($min<10){
                $min = "0".$min;
        }
        if($time<10){
                $time = "0".$time;
        }
        if($sek<10){
                $sek = "0".$sek;
        }

        return "Kl $time:$min:$sek";
```

```
}

#Hent stoy konfigurasjon fra stoyKlient.conf
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett rapport fil om den finnes
if(-e $resultatFil){
	unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#To kjoringer uten paavirking av stoyscript
for(my $i=0; $i<2;$i++){
	print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Start test
	system("./ftpKlient.pl $server $brukernavn $passord >> $resultatFil");
	#Slett nedlastet filer
	system("yes | rm fildata-*");
	skrivTilResultat "\n";

	print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
	#Pause til neste kjoring
	sleep($pauseTid);
}

#Gå gjennom stoy verdier for testing
for(my $i = 0; $i<@stoyString; $i++){
	#Kjorelende i tid for stoyscript
	for(my $t = 0; $t<@stoyKjoreTid; $t++){
		print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: $stoyKjoreTid[$t]\n";
		#Start stoy
		kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t], $stoyString[$i],
$stoySleep[$i];
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
		print hentKlokkeTid." - Starter test\n";
		#Start test
		system("./ftpKlient.pl $server $brukernavn $passord >>
$resultatFil");
		#Slett nedlastet filer
		system("yes | rm fildata-*");
		skrivTilResultat "\n";

		#Siste test trenger ikke pause
		if($t != (@stoyKjoreTid-1)){
			print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
			sleep($pauseTid);
		}

	}

}
```

startFtpTestSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
use threads::shared;

#Test input fra shell
my ($resultatFil,$server, $brukernavn, $passord) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Returner klokkeslett
sub hentKlokkeTid{
	my ($sek,$min,$time) = localtime(time);
	if($min<10){
		$min = "0".$min;
	}
	if($time<10){
		$time = "0".$time;
	}
	if($sek<10){
		$sek = "0".$sek;
	}

	return "Kl $time:$min:$sek";
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#To kjoringer uten paavirking av stoy
for(my $i=0; $i<2;$i++){
	print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Start test
	system("./ftpKlient.pl $server $brukernavn $passord >> $resultatFil");
	#Slett nedlastet filer
	system("yes | rm fildata-*");
	skrivTilResultat "\n";

	print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
	#Pause mellom hver test
	sleep($pauseTid);
}

#Kjor alle stoybelastninger sammen med test
for(my $i = 0; $i<@stoyString; $i++){
		print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
```

```
SAMTIDIG\n";
                #Egen traad for fjernstyring av stoy
                my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i], "evig",
$stoyString[$i], $stoySleep[$i];
                skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";

                #Start test
                system("./ftpKlient.pl $server $brukernavn $passord >> $resultatFil");

                #Test er ferdig. Stop stoy
                my $stopStoyTraad = new threads \&stopStoy;
                $stopStoyTraad->join;

                #Stoy har avsluttet
                $SamtidigStoy->join;
                skrivTilResultat "\n";

                #Slett nedlastet filer
                system("yes | rm fildata-*");


                #Dersom siste runde av hele testen. ingen pause
                if($i != (@stoyString-1)){
                                print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
                                #Pause mellom hver igjennomfort test
                                sleep($pauseTid);
                }

}
```

MYSQL

testMysqlKontroll.pl

```
#!/usr/bin/perl
use strict;
use IO::Socket;

#Eks: 192.168.1.10 1501 run-all-tests
my($mysqlServer,$mysqlServerPort,$testScript) = @ARGV;

#Koble til Maskin med MySQL installert
my $socket = IO::Socket::INET->new(PeerAddr => $mysqlServer,
             PeerPort => $mysqlServerPort,
             Proto    => "tcp",
             Type     => SOCK_STREAM)
    or "Feil, port er opptatt eller annen nettverks feil\n";

#Start script paa Maal maskin
print $socket "$testScript\n";

my @resultat;

#Motta resultater
while(<$socket>){
       push(@resultat,$_);
       if($_ eq "$testScript FERDIG\n"){last;}
}
#Skriv resulater til skjerm
print @resultat;

#Lukk port
close($socket);
```

testMysqlSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

# Eks: mysql.log 192.168.1.10 1501 run-all-tests
my ($resultatFil, $mysqlServer,$mysqlServerPort,$testScript) = @ARGV;

#Hent inn konfigurasjon fra fil for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Hent klokketid. Brukes ved status til skjerm
sub hentKlokkeTid{
       my ($sek,$min,$time) = localtime(time);
       if($min<10){
              $min = "0".$min;
       }
       if($time<10){
              $time = "0".$time;
       }
       if($sek<10){
              $sek = "0".$sek;
       }

       return "Kl $time:$min:$sek";
}
```

```
#Testlog text skriver
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Slett logfil dersom den finnes fra for
if(-e $resultatFil){
	unlink($resultatFil);
}

#To testkjoringer uten paavirking av stoy
for(my $i = 0; $i<2; $i++){
	print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "START TEST\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Start test
	system("./testMysqlKontroll.pl $mysqlServer $mysqlServerPort $testScript >>
$resultatFil");
	skrivTilResultat "SLUTT TEST\n";
	skrivTilResultat "\n";
	print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
	#Pause for neste test
	sleep($pauseTid);
}

#Gå gjennom stoy verdier for testing
for(my $i = 0; $i<@stoyString; $i++){
	#Kjorelende i tid for stoyscript
	for(my $t = 0; $t<@stoyKjoreTid; $t++){
			print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: $stoyKjoreTid[$t]\n";
			#Start stoy
			kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t], $stoyString[$i],
$stoySleep[$i];
			print hentKlokkeTid." - Stoy $stoyProsent[$i] ferdig - Starter
test\n";
			skrivTilResultat "START TEST\n";
			skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
			#Start test
			system("./testMysqlKontroll.pl $mysqlServer $mysqlServerPort
$testScript >> $resultatFil");
			skrivTilResultat "SLUTT TEST\n";
			skrivTilResultat "\n";

			#siste Runde trenge ingen pause
			if($t != (@stoyKjoreTid-1)){
				print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
				sleep($pauseTid);
			}

		}

}
```

testMysqlSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
```

```
use threads::shared;

#Test input fra shell
my ($resultatFil, $mysqlServer,$mysqlServerPort,$testScript) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Returner klokkeslett
sub hentKlokkeTid{
	my ($sek,$min,$time) = localtime(time);
	if($min<10){
		$min = "0".$min;
	}
	if($time<10){
		$time = "0".$time;
	}
	if($sek<10){
		$sek = "0".$sek;
	}

	return "Kl $time:$min:$sek";
}

#To kjoringer uten paavirking av stoy
for(my $i = 0; $i<2; $i++){
	print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
	skrivTilResultat "START TEST\n";
	skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
	#Start test
	system("./testMysqlKontroll.pl $mysqlServer $mysqlServerPort $testScript >>
$resultatFil");
	skrivTilResultat "SLUTT TEST\n";
	skrivTilResultat "\n";
	print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
	#Pause mellom hver test
	sleep($pauseTid);
}


#Kjor alle stoybelastninger sammen med test
for(my $i = 0; $i<@stoyString; $i++){
		print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";
		skrivTilResultat "START TEST\n";
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";
		#Egen traad for fjernstyring av stoy
		my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i], "evig",
$stoyString[$i], $stoySleep[$i];
```

```
                #Start test
                system("./testMysqlKontroll.pl $mysqlServer $mysqlServerPort
$testScript >> $resultatFil");

                #Test er ferdig. Stop stoy
                my $stopStoyTraad = new threads \&stopStoy;
                $stopStoyTraad->join;

                #Stoy har avsluttet
                $SamtidigStoy->join;
                skrivTilResultat "SLUTT TEST\n";
                skrivTilResultat "\n";

                #Dersom siste runde av hele testen. ingen pause
                if($i!=(@stoyKjoreTid-1)){
                        print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
                        #Pause mellom hver igjennomfort test
                        sleep($pauseTid);
                }

                }
```

LDAP

drift.ldif

```
dn: dc=drift,dc=no
objectClass: dcObject
objectClass: organization
dc: drift
description: The Example Corporation
o: drif Org
```

endreAtributt.pl

```
#!/usr/bin/perl

use strict;
#Bibliotekfiler for ldap
use Net::LDAP;
use Net::LDAP::LDIF;
use Time::HiRes;

#Start paramentere
my ($server,$brukerdn,$passord,$ldifForBrukere) = @ARGV;

my $ldap;

my $totalt = 0;

#Ferdig liste for eksisterende brukere, kun lese
my $ldif = Net::LDAP::LDIF->new( $ldifForBrukere, "r");
#Start tid for test
my $startTid = [ Time::HiRes::gettimeofday( ) ];
#Saa lenge det er brukere
while( not $ldif->eof ( ) ) {
       #Hent bruker
       my $entry = $ldif->read_entry ( );
       #Feil ved lesing av bruker
       if ( $ldif->error ( ) ) {
     print "Error melding: ", $ldif->error ( ), "\n";
     print "Error linje:\n", $ldif->error_lines ( ), "\n";
   } else {

              #Opp kobling med skrive rettigheter. ny oppkobling for hver endring
              $ldap = Net::LDAP->new($server)
                 or die "Could not connect!";
              $ldap->bind($brukerdn, password=>$passord);

              #tell antall brukere endret
              $totalt++;

              #Tre endringer.Erstatt, legg til og slett artibutt
              $ldap->modify( $entry, replace => { 'postalAddress' =>
('Bruker'.$totalt.'Postadresse') } );
              $ldap->modify( $entry, add => { 'mail' =>
('Bruker'.$totalt.'@vmdrift.no') } );
              $ldap->modify( $entry, delete => { 'title' => ('TittelBruker'.$totalt)
} );

              #koble fra.
              $ldap->unbind;



   }
 }
#Ferdig med aa lese ldif for brukere
$ldif->done ( );
```

```
#Lagre tids forbruk
my $ferdig = Time::HiRes::tv_interval( $startTid );

#Skriv total brukere endret og og tid for test
print "$totalt $ferdig\n";
```

lagLdifFil.pl

```
#!/usr/bin/perl

#print "-----------------------------------------------------------\n";
# Ved aa ta "formange" submapper vil antall brukere gaa over bestemt antall
# Eks: ./lagLdifFil.pl 100 2 DC=vmdrift,DC=no ldifForBrukere.ldif ldifForOrg.ldif
my ($brukereIgjen, $maksUndermapper,$morDn, $filnavnBrukere, $filnavnOrg)=@ARGV;

#Navn for filene
open(LDIF_BRUKERE,">".$filnavnBrukere);
open(LDIF_ORG,">".$filnavnOrg);

my $randomUndermapper;
my $randomBrukere = 0;
my $totaltBrukere = 0;
my $totaltGrupper = 0;

#Bestemm antall undermapper og brukere for en org klasse
sub settRandom{
	#Hvor lavt skal undermapper gaa. Ikke høyre en maks lengde
	$randomUndermapper = int(rand($maksUndermapper)+1);
	#Hvor mange brukere skal tildelses mappen
	$randomBrukere = int(rand(($brukereIgjen/2))+1);
	#Det maa være nok brukere igjen til alle undermapper
	if($randomBrukere<$randomUndermapper && $brukereIgjen>=$randomUndermapper){
		$randomBrukere=$randomUndermapper;
	#Hvis ikke maa det bli faare undermapper
	}elsif ($randomUndermapper>$brukereIgjen && $brukereIgjen != 
0){$randomUndermapper=$randomBrukere;}
}

#Lag ny org klasse med riktig dn og navn
sub lagOrg{
	my($dn, $gruppenr) = @_;
	print LDIF_ORG "dn: $dn$morDn
objectClass: organizationalUnit
objectClass: top
description: Beskrivelse av Gruppe$gruppenr
ou: Gruppe$gruppenr
postalAddress: Gruppe$gruppenrAdresse
postalCode: $gruppenr
street: GateGruppe$gruppenr
telephoneNumber: $gruppenr\n\n";


}

#lag ny posix bruker. Med riktig dn og navn
sub lagBruker{
	my($dn, $brukernr) = @_;
	print LDIF_BRUKERE "dn: $dn$morDn
objectClass: inetOrgPerson
objectClass: organizationalPerson
objectClass: posixAccount
objectClass: person
objectClass: top
carLicense: $brukernr
cn: Bruker$brukernr
description: Beskrivelse av Bruker$brukernr
```

```
displayName: Bruker$brukernr Bruker$brukernr
employeeNumber: $brukernr
gidNumber: ".$brukernr."00
givenName: Bruker$brukernr
homeDirectory: /home/Bruker$brukernr
mobile: $brukernr
postalAddress: Bruker$brukernrAdresse
postalCode: $brukernr
preferredLanguage: Norsk
roomNumber: R$brukernr
sn: Bruker$brukernr
telephoneNumber: $brukernr
title: TittelBruker$brukernr
uid: Bruker$brukernr
uidNumber: 100$brukernr\n\n";
}

#Forsatt brukere igjen
while($brukereIgjen!=0){
	#Sett antall brukere og org klasser til mappen
	settRandom;
	#Trekk fra random brukere brukt i org klassen
	$brukereIgjen -= $randomBrukere;

	#Antall submapper laget for klassen
	$subBrukereIgjen = $randomBrukere;
	#Navn for org klasse
	$subOu = "";

	#lag org med alle under org klasser
	for(my $i=1; $i<=$randomUndermapper; $i++){


		#Så lenge det er forsatt undermapper og brukere
		if($i!=$randomUndermapper && $subBrukereIgjen != 0){
			#Sett antall brukere for mappen
			$subBrukere = int(rand($subBrukereIgjen)+1);

		}else{
		 #Bruk opp resten av brukerne
		 $subBrukere = $subBrukereIgjen
		}

		#Alle undermapper maa minst ha en bruker
		if(($subBrukereIgjen-1) <= ($randomUndermapper-$i)){
			$subBrukere = 1;
		}
		#Alle org klasser maa ha minst en bruker
		if($subBrukere>=($randomUndermapper-$i)){
			$subBrukere = $subBrukere - ($randomUndermapper-$i);
			if($subBrukere == 0){$subBrukere = 1;} #Ved 2 minus 2
		}

		#Tell total org klasser
		$totaltGrupper++;
		#trekk fra oppbrukte brukere for mappen
		$subBrukereIgjen -= $subBrukere;
		#Lag navn for orgklassen
		$subOu = "ou=Gruppe".$totaltGrupper.",".$subOu;
		#Lag org klassen
		lagOrg $subOu, $totaltGrupper;

		#Lag brukere for orgmappen
		for(my $j=1; $j<=$subBrukere; $j++){
			#Registerer total brukere
			$totaltBrukere++;
			#Lag ldap omraade for bruker
			$brukerDn = "cn=Bruker".$totaltBrukere.",".$subOu;
```

```
                    #Lag bruker
                    lagBruker $brukerDn,$totaltBrukere;
                }



        }


}

#Skriv ut total org klasser og total postfix bruker klasser
print "$totaltGrupper $totaltBrukere\n";

#Lukk filene
close(LDIF_BRUKERE);
close(LDIF_ORG);
```

lastOppLdifFil.pl

```
#!/usr/bin/perl

use strict;
#Bibliotek fil for ldap oppkobling
use Net::LDAP;
use Net::LDAP::LDIF;
use Time::HiRes;

my $nettMottatt = 0;
my $nettSendt = 0;

#eksempel: localhost CN=root,DC=vmdrift,DC=no 31pass+tull ldifForOrg.ldif
my ($server,$brukerdn,$passord,$ldifFil) =@ARGV;

#Oppkobling til server
my $ad = Net::LDAP->new($server)
                or die "Fikk ikke koblet til!";
$ad->bind($brukerdn, password=>$passord);

my $totalt = 0;

#Les ldif fil for ldap
my $ldif = Net::LDAP::LDIF->new( $ldifFil, "r");
#Noter start tid for test
my $startTid = [ Time::HiRes::gettimeofday( ) ];

#Dersom det er mer data i fil
while( not $ldif->eof ( ) ) {
       #Hent ut Klasser
       my $entry = $ldif->read_entry ( );
   if ( $ldif->error ( ) ) {
     print "Error melding: ", $ldif->error ( ), "\n";
     print "Error linje:\n", $ldif->error_lines ( ), "\n";
   } else {

            #Legg inn uthentet klasse på ldap server
            my $result = $ad->add( $entry );
    warn $result->error(  ) if $result->code(  );
            #Tell antall innlegginger
            $totalt++;

   }
 }
#Lukk ldif fil lesing
$ldif->done ( );
#Koble fra LDAP server
$ad->unbind;
```

```
#Noter tid ferdig
my $ferdig = Time::HiRes::tv_interval( $startTid );


#Skriv ut antall innlegg og totaltid
print "$totalt $ferdig\n";
```

ldapTest.pl

```
#!/usr/bin/perl

# Eks: ./ldapTest.pl localhost DC=vmdrift,DC=no CN=root,DC=vmdrift,DC=no
31pass+tull 1000 4 500 1
my
($server,$morDn,$brukerdn,$passord,$antallBrukere,$maksUndermapper,$maksSok,$nyeLdi
fFiler, $filnavnBrukere, $filnavnOrg) = @ARGV;

#Husk antall undermapper
my $antallUndermapper;

#Hent ut skrift ved kjoring
sub kjorScript{
	my($scriptNavn) = @_;
	#Lagre tekst fra skjerm
	my $paaSkjerm = "";
	open(TEKST,"./".$scriptNavn." |") || die "Failed: $!\n";
	while ( <TEKST> )
		{
			$paaSkjerm .= $_;

		}
	close TEKST;
	#Returner tekst fra skjerm
	return $paaSkjerm;

}

#Skal det lages nye ldif filer for bruker og org. Hvis filene ikke finnes. lag dem
uansett.
if($nyeLdifFiler == 1 || (!(-e $filnavnBrukere) && !(-e $filnavnOrg))){
	#Lag ldif filer for bruker og org
	($antallUndermapper, $antallBrukere) = split /\W/, kjorScript ("lagLdifFil.pl
$antallBrukere $maksUndermapper $morDn $filnavnBrukere $filnavnOrg");
	print "Submapper $nyeLdifFiler : $antallUndermapper Brukere:
$antallBrukere\n";
}

my $tidLagOrg;
#Last opp ldif data til ldapserver for org. ta tid og antall
($antallUndermapper,$tidLagOrg) = split /\s/, kjorScript ("lastOppLdifFil.pl
$server $brukerdn $passord $filnavnOrg");
printf "%-25s", "Last opp Org ($antallUndermapper): ";
printf "$tidLagOrg - %.3f i sek\n",($antallUndermapper/$tidLagOrg);


my $tidLagBrukere;
#Last opp ldif data til ldapserver for brukere. ta tid og antall
($antallBrukere,$tidLagBrukere) = split /\s/, kjorScript ("lastOppLdifFil.pl
$server $brukerdn $passord $filnavnBrukere");
printf "%-25s", "Last opp Brukere ($antallBrukere): ";
printf "$tidLagBrukere - %.3f i sek\n",($antallBrukere/$tidLagBrukere);

printf "%-25s", "Sok etter CN: ";
#Sok etter nokkel atributter etter antall. ta tiden
my(undef,$tidSokCn) = split /\s/, kjorScript ("sokEtterCn.pl $server $brukerdn
```

```
$passord $morDn $antallBrukere $maksSok");
printf "$tidSokCn - %.3f i sek\n",($maksSok/$tidSokCn);


printf "%-25s", "Sok etter Atributter: ";
#Sok etter ikke nokkel atributter etter antall. ta tiden.
my(undef,$tidSokAtributt) = split /\s/,kjorScript ("sokEtterAtributt.pl $server
$brukerdn $passord $morDn $antallBrukere $maksSok");
printf "$tidSokAtributt - %.3f i sek\n",($maksSok/$tidSokAtributt);

printf "%-25s", "Enring av brukere: ";
#Utfor endringer erstatt, legg til og fjern paa alle brukere
my(undef,$tidEndreBrukere) = split /\s/,kjorScript ("endreAtributt.pl $server
$brukerdn $passord $filnavnBrukere");
printf "$tidEndreBrukere - %.3f i sek\n",($antallBrukere/$tidEndreBrukere);

printf "%-25s", "Slette alle brukere: ";
#Slett alle brukere ved a bruke ldif fil
my(undef,$tidSlettBrukere) = split /\s/,kjorScript ("sokBrukereOgSlett.pl $server
$brukerdn $passord $morDn");
printf "$tidSlettBrukere - %.3f i sek\n",($antallBrukere/$tidSlettBrukere);

#Slett alle org ved aa bruke ldif fil
printf "%-25s", "Slette alle Org: ";
my(undef,$tidSlettOrg) = split /\s/,kjorScript ("sokEtterOrgOgSlett.pl $server
$brukerdn $passord $morDn");
printf "$tidSlettOrg - %.3f i sek\n",($antallUndermapper/$tidSlettOrg);
```

ldapTestSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

#Hent klokketid. Brukes ved status til skjerm
sub hentKlokkeTid{
	my ($sek,$min,$time) = localtime(time);
	if($min<10){
		$min = "0".$min;
	}
	if($time<10){
		$time = "0".$time;
	}
	if($sek<10){
		$sek = "0".$sek;
	}

	return "Kl $time:$min:$sek";
}

# Eks (AD): ./ldapTestSenario1.pl testSenario1.log 128.39.80.83 DC=drift,DC=no
CN=administrator,cn=users,DC=drift,DC=no 31pass+tull 100 2 50 1 brukere.ldif
org.ldif
# Eks (OpenLDAP):
my ($resultatFil,
$server,$morDn,$brukerdn,$passord,$antallBrukere,$maksUndermapper,$maksSok,$nyeLdif
Filer, $filnavnBrukere, $filnavnOrg) = @ARGV;

#Skriv ut konfigurasjon for test
print "\nResultatFil: $resultatFil\n";
print "LDAP serverIP: $server\n";
print "LDAP Rot: $morDn\n";
print "Brukernavn: $brukerdn\n";
print "Passord: $passord\n";
print "AntallBrukere: $antallBrukere\n";
```

```
print "Maks submappedybde (i Orgclass): $maksUndermapper\n";
print "Antall sok: $maksSok\n";
print "Lage nye Ldif-filer (Bool): $nyeLdifFiler\n";
print "Brukere LDIF: $filnavnBrukere\n";
print "Org LDIF: $filnavnOrg\n\n";


#Hent inn konfigurasjon fra fil for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Testlog text skriver
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Slett logfil dersom den finnes fra for
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Dersom ldif for brukere og org finnes eller skal lages
if((-e $filnavnBrukere) && (-e $filnavnOrg) || $nyeLdifFiler == 1){

#To testkjoringer uten paavirking av stoy
	for(my $i = 1; $i<=2; $i++){
		print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
		skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";

		#Start test
		system("./ldapTest.pl $server $morDn $brukerdn $passord $antallBrukere
$maksUndermapper $maksSok $nyeLdifFiler $filnavnBrukere $filnavnOrg >>
$resultatFil");
		skrivTilResultat "\n";

		print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
		#Pause til neste kjoring
		sleep($pauseTid);
	}

#Gå gjennom stoy verdier for testing
	for(my $i = 0; $i<@stoyString; $i++){ #i= 0
	#Kjorelende i tid for stoyscript
			for(my $t = 0; $t<@stoyKjoreTid; $t++){ #t = 0

				print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: $stoyKjoreTid[$t]\n";
				#Start stoy
				kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t],
$stoyString[$i], $stoySleep[$i];
				print hentKlokkeTid." - Stoy $stoyProsent[$i] ferdig\n";

				skrivTilResultat "StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: $stoyKjoreTid[$t]\n";
				print hentKlokkeTid." - Starter Test\n";
				#Start test
				system("./ldapTest.pl $server $morDn $brukerdn $passord
$antallBrukere $maksUndermapper $maksSok 0 $filnavnBrukere $filnavnOrg >>
$resultatFil");
				skrivTilResultat "\n";
				#siste Runde trenge ingen pause
				if($t != (@stoyKjoreTid-1)){
```

```
                                print hentKlokkeTid." - $pauseTid sek pause for
neste kjoring\n";
                                sleep($pauseTid);
                        }

                }

        }
}else{
        print "Error: Finner ikke gitt LDIF filer\n";
}
```

ldapTestSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
use threads::shared;

#Returner klokkeslett
sub hentKlokkeTid{
        my ($sek,$min,$time) = localtime(time);
        if($min<10){
                $min = "0".$min;
        }
        if($time<10){
                $time = "0".$time;
        }
        if($sek<10){
                $sek = "0".$sek;
        }

        return "Kl $time:$min:$sek";
}

# Eks (MS AD): ./ldapTestSenario2.pl ldapSenario2-3000b-5sub-1000sok-t1-bootet.log
128.39.80.83 DC=drift,DC=no CN=administrator,cn=users,DC=drift,DC=no 31pass+tull
3000 5 1000 0 brukere3000.ldif org5sub.ldif

my ($resultatFil,
$server,$morDn,$brukerdn,$passord,$antallBrukere,$maksUndermapper,$maksSok,$nyeLdif
Filer, $filnavnBrukere, $filnavnOrg) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
        unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
        my($tekst) = @_;
        open RESULTAT,">>$resultatFil";
        print RESULTAT $tekst;
        close RESULTAT;
}
#Dersom ldif for brukere og org finnes eller skal lages
if((-e $filnavnBrukere) && (-e $filnavnOrg) || $nyeLdifFiler == 1){
        print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
```

```
        skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
        #Kjor test uten samtidig stoy
        system("./ldapTest.pl $server $morDn $brukerdn $passord $antallBrukere
$maksUndermapper $maksSok $nyeLdifFiler $filnavnBrukere $filnavnOrg >>
$resultatFil");
        skrivTilResultat "\n";

        print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
        #Pause for teste kjoring
        sleep($pauseTid);

        #Kjor alle stoybelastninger sammen med test
        for(my $i = 0; $i<@stoyString; $i++){
                        print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: SAMTIDIG\n";
                        #Start stoy traad for samtidig kjoring
                        my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i],
"evig", $stoyString[$i], $stoySleep[$i];
                        skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";

                        #Start test
                        system("./ldapTest.pl $server $morDn $brukerdn $passord
$antallBrukere $maksUndermapper $maksSok 0 $filnavnBrukere $filnavnOrg >>
$resultatFil");

                        #Stop stoy, test er ferdig
                        my $stopStoyTraad = new threads \&stopStoy;
                        $stopStoyTraad->join;

                        #Stoy er ferdig
                        $SamtidigStoy->join;
                        skrivTilResultat "\n";

                        #Dersom siste runde av hele testen. ingen pause
                        if($i != (@stoyString-1)){
                                print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
                                #Pause mellom hver igjennomfort test
                                sleep($pauseTid);
                                }

                }

}else{
        print "Error: Finner ikke gitt LDIF filer\n";
}
```

sokBrukereOgSlett.pl

```
#!/usr/bin/perl

use strict;
#Bibliotek for kontakt med ldapserver
use Net::LDAP;
#For lesing av ldif filer
use Net::LDAP::LDIF;
use Time::HiRes;

my ($server,$brukerdn,$passord,$baseDn) = @ARGV; #localhost
CN=root,DC=vmdrift,DC=no 31pass+tull DC=vmdrift,DC=no 1000
my $text;
#Tell antall for slettinger
my $count;

my $ldap = Net::LDAP->new($server)
                or die "Kunne ikke kobletil!";
```

```
$ldap->bind($brukerdn, password=>$passord);

my $antall = 1;
do{ # Finn ut hvor mange brukere det er for sletting
	$text = $ldap->search(filter=>'cn=Bruker'.$antall,base=>$baseDn);
	$count = $text->count;
	if($count == 0){
		$antall--;
	}else{
		$antall += $count;
	}
}while($count != 0);

#Tell antall slettet
my $antSlettet = 0;
#Noter startid
my $startTid = [ Time::HiRes::gettimeofday( ) ];
#For hver bruker som skal slettes
for (my $j=$antall; $j>0; $j--) {
	#Sok etter bruker
	$text = $ldap->search(filter=>'cn=Bruker'.$j,base=>$baseDn);
	#Hvor mange funn med anitt brukernavn
	$count = $text->count;



	#For hver bruker. (Skal kun vaere en) slett bruker
	for (my $i=0; $i<$count; $i++) {
		my $entry = $text->entry($i);
			#Slett bruker
			$ldap->delete($entry);
			#Tell antall funnet
			$antSlettet++;
		}
}
#Koble fra ldapserver
$ldap->unbind;

#Noter total tid
my $ferdig = Time::HiRes::tv_interval( $startTid );

#Print antall brukere slettet og total tid for sletting
print "$antSlettet $ferdig\n";
```

sokEtterAtributt.pl

```
#!/usr/bin/perl

use strict;
#Bibliotek for LDAP tilkobling
use Net::LDAP;
use Time::HiRes;

#Start parametere
my ($server,$brukerdn,$passord,$baseDn,$antall,$antallSok) = @ARGV;



my $ldap;
my $funn= 0;
#Noter start tid
my $startTid = [ Time::HiRes::gettimeofday( ) ];
#Utfor antall sok
for (my $j=1; $j<=$antallSok; $j++) {
	#Koble til ldap server
	$ldap = Net::LDAP->new($server)
		or die "Kunne ikke koble til!";
```

```
        $ldap->bind($brukerdn, password=>$passord);

        #Sok paa tilfeldig bruker
        my $search  = int((rand($antall)+1));
        #Sok paa ikke nokkel atributt beskrivelse
        my $text = $ldap->search(filter=>'description=Beskrivelse av
Bruker'.$search,base=>$baseDn);

        #Antall funnet. Bor bare vaere 1
        my $count = $text->count;
        $funn += $count;

        #For vaert funn
        for (my $i=0; $i<$count; $i++) {
                        #hent ut tekst i funn
          my $entry = $text->entry($i)
        }
        #Koble fra ldapserver
        $ldap->unbind;
}

#Noter totaltid
my $ferdig = Time::HiRes::tv_interval( $startTid );

#Skriv ut antall funn og tid ferdig
print "$funn $ferdig\n";
```

sokEtterCn.pl

```
#!/usr/bin/perl

use strict;
#Bibloktek for tilkobling til LDAP server
use Net::LDAP;
use Time::HiRes;

#Start parametere
my ($server,$brukerdn,$passord,$baseDn,$antall,$antallSok) = @ARGV;

#LDAP server objekt
my $ldap;
my $funn= 0;

#Noter start tid
my $startTid = [ Time::HiRes::gettimeofday( ) ];

#Utfor antall sok
for (my $j=1; $j<=$antallSok; $j++) {
        #Koble til LDAP server
        $ldap = Net::LDAP->new($server)
                  or die "Kunne ikke koble til LDAP Server!";
        $ldap->bind($brukerdn, password=>$passord);

        #Finn tilfeldig bruker aa soke paa
        my $search  = int((rand($antall)+1));
        #Utfor ldap sok på nokkel atributt
        my $text = $ldap->search(filter=>'cn=Bruker'.$search,base=>$baseDn);

        #Noter antall funn
        my $count = $text->count;

        #Lagre antall funn
        $funn += $count;

        for (my $i=0; $i<$count; $i++) {
                #Hent ut tekt i funnet bruker klasse
          my $entry = $text->entry($i);
```

```
        }
        #Koble fra LDAP server
        $ldap->unbind;
}

#Noter total kjoretid
my $ferdig = Time::HiRes::tv_interval( $startTid );

#Skriv antall funn og kjoretid
print "$funn $ferdig\n";
```

sokEtterOrgOgSlett.pl

```
#!/usr/bin/perl

use strict;
#Bibliotek fil for LDAP Server oppkobling
use Net::LDAP;
#For haatering av LDIF filer
use Net::LDAP::LDIF;
use Time::HiRes;

#Start parametere
my ($server,$brukerdn,$passord,$baseDn) = @ARGV;# localhost
CN=root,DC=vmdrift,DC=no 31pass+tull DC=vmdrift,DC=no 39

my $count;
my $text;

#Koble til LDAP Server
my $ldap = Net::LDAP->new($server)
     or die "Kunne ikke koble til server!";
$ldap->bind($brukerdn, password=>$passord);

my $antall = 1;
do{ # Finn ut hvor mange grupper det er for sletting
        $text = $ldap->search(filter=>'ou=Gruppe'.$antall,base=>$baseDn);
        $count = $text->count;
        if($count == 0){
                $antall--;
        }else{
                $antall += $count;
        }
}while($count != 0);

#Slette teller
my $antSlettet = 0;
#noter starttid for test
my $startTid = [ Time::HiRes::gettimeofday( ) ];
#Gaa igjennom antall funnet i sok
for (my $j=$antall; $j>0; $j--) {
        #Org klasse som skal sokes på
        $text = $ldap->search(filter=>'ou=Gruppe'.$j,base=>$baseDn);

        #Antall funnet
        $count = $text->count;



        #For antall funn
        for (my $i=0; $i<$count; $i++) {
                #Hent ut sok verdi funnet
          my $entry = $text->entry($i);
                #Slett org klasse
                        $ldap->delete($entry);
                #Tell antall slettet
```

```
                    $antSlettet++;
                }
}
#koble fra LDAP server
$ldap->unbind;
#Noter tid ferdig for test
my $ferdig = Time::HiRes::tv_interval( $startTid );

#Skriv antall slettet org klasser og total tid for sletteing
print "$antSlettet $ferdig\n";
```

TERMINAL

tsTester.pl

```
#!/usr/bin/perl

use strict;
use threads;
use threads::shared;
use Time::HiRes;
use Shell;
use IO::Socket;

#Port for perscript kjorende paa Windows Server
use constant WINDOWSSERVERPORT => "1500";

#Oppkoblinger som maa gjentas
my $feiletTotal : shared = 0;
#liste med tid for feilet opp kobling til velykket oppkobling
my @tidFeilet : shared;
#Liste for hvilken bruker som fikk feil oppkobling
my @tsFeilet : shared;
#tid for innloging for hver bruker
my @tkTider : shared;

sub kobleTilRdp{
	my($bruker,$passord,$domene,$ipaddresse,$opplosning,$brukernr) = @_;

	#Se feilmeldinger slik at oppkobling kan gjøres paa nytt
	$Shell::capture_stderr = 1;
	#Feil for denne oppkoblingen
	my $feilet = 0;
	#Total tid hvor denne oppkoblinen har feilet
	my $totalFeiltid = 0;
	#Tekst gitt av rdesktop
	my $text;
	#Ta tid for oppkobling
	my $termStart = [ Time::HiRes::gettimeofday( ) ];
	do{
		#Ta tid for eventuel feil oppkobling
		my $startFeiltid = [ Time::HiRes::gettimeofday( ) ];
		#Start oppkobling til terminal server
		$text = rdesktop ("-u $bruker -d $domene -p $passord -k no -g
$opplosning $ipaddresse");
		($text, undef) = split / /,$text;

		#Finn ut om oppkobling har feilet
		if($text eq "ERROR:"){
			#Tell antall feil oppkoblingen får
			$feilet++;
			#Legg sammen tid hvor oppkobling har feilet
			$totalFeiltid += Time::HiRes::tv_interval( $startFeiltid
);
		}
	#Prov paa nytt ved feiling
	}while($text eq "ERROR:");

	#Tid for ferdig kjort test
	my $termTid = Time::HiRes::tv_interval( $termStart );


	#Lock felles resultats variabler for oppdatering
	lock($feiletTotal);
	lock(@tidFeilet);
	lock(@tkTider);
	lock(@tsFeilet);
```

```
        #Antall brukerens oppkoblinger har feilet
        $tsFeilet[$brukernr] = $feilet;
        #Tid for gjeldende oppkobling
        $tkTider[$brukernr] = $termTid;
        #Oppdater total feil oppkoblinger
        $feiletTotal += $feilet;

        #Forsoks tid for velykket oppkobling
        $tidFeilet[$brukernr] = $totalFeiltid;


}

#Oppretting av brukere paa Windows Server
sub windowsBrukere{

        my ($ipaddresse,$brukernavn, $passord, $antall, $fjern) = @_;

        #Prov aa koble til server
        my $socket = IO::Socket::INET->new(PeerAddr => $ipaddresse,
                                PeerPort => WINDOWSSERVERPORT,
                                Proto    => "tcp",
                                Type     => SOCK_STREAM)
        or die "Kunne ikke koble til ".WINDOWSSERVERPORT;

        #Send brukernavn (prefix), passord, og fjern (bool)
        print $socket "$brukernavn $passord $antall $fjern\n";

        #Vent til alle brukere er laget eller slettet
        my $motta = <$socket>;

        #Lukk oppkoblingen
        close($socket);

        #Server har klart aa lage brukere eller fjernet dem
        if($motta eq "ferdig\n"){
                return 1;
        }else{
        #Fjerning/laging av brukere mislykkes
                return 0;
        }

}

#rdesktop katalog paa CentOS 5.4
my $rdpKatalog = "/usr/bin/rdesktop";

my ($Oppkoblinger, $bruker, $passord, $domene, $ipaddresse, $opplosning) = @ARGV;

#Lag en bruker og ta vare på servertilbakemld (Laget eller ikke)
my $brukereLaget = windowsBrukere $ipaddresse, $bruker, $passord, $Oppkoblinger, 0;

#Sjekker om rdesktop er installert.
if(-e $rdpKatalog && $brukereLaget == 1){
        my @rdpKlienter;
                print "Ant tsKlienter: $Oppkoblinger\n";
                #Ta den totale tiden
                my $start = [ Time::HiRes::gettimeofday( ) ];
                #Start oppkoblinger
                for(my $j = 1; $j<= $Oppkoblinger; $j++){
                        #Koble til terminal server
                        $rdpKlienter[$j-1] = new threads \&kobleTilRdp, "$bruker$j",
$passord, $domene, $ipaddresse, $opplosning,$j;
                }
                # Vent til at alle er koblet fra
                foreach my $rdpKlient(@rdpKlienter){$rdpKlient->join;}

                #Noter totaltid brukt
```

```
                my $totaltid = Time::HiRes::tv_interval( $start );

                #Skriv ut resultat
                for(my $j = 1; $j<= $Oppkoblinger; $j++){
                        #Skriv resultat for hver bruker
                        print "$bruker$j: $tkTider[$j] - Feilet: $tsFeilet[$j] -
Feilettid: $tidFeilet[$j]\n";
                }

                #Skriv totall oppkoblinger og totaltid for alle oppkoblinger
                print "TID for $Oppkoblinger oppkobling: $totaltid\n";
                #Skriv total oppkobling som maate igjentas
                print "Feilet total: $feiletTotal\n";


                # Fjern brukerne fra serveren og deres hjemmeområde med profildata
                windowsBrukere $ipaddresse,$bruker, $passord, $Oppkoblinger, 1;




}else{
        print "Error: rdesktop maa vaere installert og server maa ha brukere\n";
}
```

tsTestSenario1.pl

```
#!/usr/bin/perl
use strict;
#Styring og innhenting av konfigurasjon til stoyscript
use stoyKlient;

#Hent klokketid. Brukes ved status til skjerm
sub hentKlokkeTid{
        my ($sek,$min,$time) = localtime(time);
        if($min<10){
                $min = "0".$min;
        }
        if($time<10){
                $time = "0".$time;
        }
        if($sek<10){
                $sek = "0".$sek;
        }

        return "Kl $time:$min:$sek";
}

my ($resultatFil, $Oppkoblinger, $bruker, $passord, $domene, $ipaddresse,
$opplosning) = @ARGV;

#Hent inn konfigurasjon fra fil for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent, $pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett logfil dersom den finnes fra for
if(-e $resultatFil){ #Slett rapport fil om den finnes
        unlink($resultatFil);
}

#Testlog text skriver
sub skrivTilResultat{
```

```
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#testkjoringer uten paavirking av stoy
print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
#Start test
system("./tsTester.pl $Oppkoblinger $bruker $passord $domene $ipaddresse
$opplosning >> $resultatFil");
skrivTilResultat "\n";
print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
#Pause til neste kjoring
sleep($pauseTid);

#Gå gjennom stoy verdier for testing
for(my $i = 0; $i<@stoyString; $i++){
	#Kjorelende i tid for stoyscript
	for(my $t = 0; $t<@stoyKjoreTid; $t++){
		print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: $stoyKjoreTid[$t]\n";
		#Start stoy
		kjorStoy $stoyProsent[$i], $stoyKjoreTid[$t], $stoyString[$i],
$stoySleep[$i];

		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
$stoyKjoreTid[$t]\n";
		#Start test
		system("./tsTester.pl $Oppkoblinger $bruker $passord $domene
$ipaddresse $opplosning >> $resultatFil");
		skrivTilResultat "\n";

		#Siste Runde trenge ingen pause
		if($t!=(@stoyString-1)){
			print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
			#Pause til neste kjoring
			sleep($pauseTid);
		}

	}

}
```

tsTestSenario2.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
use threads::shared;

#Returner klokkeslett
sub hentKlokkeTid{
	my ($sek,$min,$time) = localtime(time);
	if($min<10){
		$min = "0".$min;
	}
	if($time<10){
		$time = "0".$time;
	}
	if($sek<10){
		$sek = "0".$sek;
```

```
	}

	return "Kl $time:$min:$sek";
}

my ($resultatFil, $oppkoblinger, $bruker, $passord, $domene, $ipaddresse,
$opplosning) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Kjoringer uten paavirking av stoy
print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
#Start test
system("./tsTester.pl $oppkoblinger $bruker $passord $domene $ipaddresse
$opplosning >> $resultatFil");
skrivTilResultat "\n";
print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
#Pause mellom hver test
sleep($pauseTid);

#Kjor alle stoybelastninger sammen med test
for(my $i = 0; $i<@stoyString; $i++){
		print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";
		#Egen traad for fjernstyring av stoy
		my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i], "evig",
$stoyString[$i], $stoySleep[$i];
		skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG\n";

		#Start test
		system("./tsTester.pl $oppkoblinger $bruker $passord $domene
$ipaddresse $opplosning >> $resultatFil");

		#Test er ferdig. Stop stoy
		my $stopStoyTraad = new threads \&stopStoy;
		$stopStoyTraad->join;

		#Stoy har avsluttet
		$SamtidigStoy->join;
		skrivTilResultat "\n";

		#Dersom siste runde av hele testen. ingen pause
		if($i!=(@stoyString-1)){
			print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
			#Pause mellom hver igjennomfort test
			sleep($pauseTid);
		}
```

```
        }
```

teTestSenario3.pl

```
#!/usr/bin/perl
use strict;
#Stoy konfigurasjoner og nettverskstyring
use stoyKlient;
use threads;
use threads::shared;

#Returner klokkeslett
sub hentKlokkeTid{
	my ($sek,$min,$time) = localtime(time);
	if($min<10){
		$min = "0".$min;
	}
	if($time<10){
		$time = "0".$time;
	}
	if($sek<10){
		$sek = "0".$sek;
	}

	return "Kl $time:$min:$sek";
}

my ($resultatFil, $oppkoblinger, $bruker, $passord, $domene, $ipaddresse,
$opplosning) = @ARGV;

#Last inn konfigrasjoner for stoyscript
my($stoyKjoreTid, $stoyString, $stoySleep,$stoyProsent,$pauseTid)= lesStoyConf;
my @stoyKjoreTid = @$stoyKjoreTid;
my @stoyString = @$stoyString;
my @stoySleep = @$stoySleep;
my @stoyProsent = @$stoyProsent;

#Slett gammel logfil
if(-e $resultatFil){ #Slett rapport fil om den finnes
	unlink($resultatFil);
}

#Skriv resultat til fil
sub skrivTilResultat{
	my($tekst) = @_;
	open RESULTAT,">>$resultatFil";
	print RESULTAT $tekst;
	close RESULTAT;
}

#Kjoringer uten paavirking av stoy
print hentKlokkeTid." - StoyProsent: 0 - StoyKjoreTid: 0\n";
skrivTilResultat "StoyProsent: 0 - StoyKjoreTid: 0\n";
#Start test
system("./tsTester.pl $oppkoblinger $bruker $passord $domene $ipaddresse
$opplosning >> $resultatFil");
skrivTilResultat "\n";
print hentKlokkeTid." - $pauseTid sek pause for neste kjoring\n";
#Pause mellom hver test
sleep($pauseTid);

#Prov med alle alle stoybelastninger
for(my $i = 0; $i<@stoyString; $i++){
	#Kjor alle stoy lengde tider
	for(my $t = 0; $t<@stoyKjoreTid; $t++){
```

```
                    print hentKlokkeTid." - StoyProsent: $stoyProsent[$i] -
StoyKjoreTid: SAMTIDIG ETTER $stoyKjoreTid[$t] SEK STOY KJORING ALENE\n";

                    #Start stoy
                    my $SamtidigStoy = new threads \&kjorStoy, $stoyProsent[$i],
"evig", $stoyString[$i], $stoySleep[$i];

                    #La stoy kjore alene i angitt tid
                    sleep ($stoyKjoreTid[$t]);

                    print hentKlokkeTid." - Starter Test nå etter $stoyKjoreTid[$t]
sek STOY\n";
                    skrivTilResultat "StoyProsent: $stoyProsent[$i] - StoyKjoreTid:
SAMTIDIG ETTER $stoyKjoreTid[$t] SEK STOY KJORING ALENE\n";

                    #Start saa test kjor sammen med stoy
                    system("./tsTester.pl $oppkoblinger $bruker $passord $domene
$ipaddresse $opplosning >> $resultatFil");

                    #Test er ferdig, stop stoy
                    my $stopStoyTraad = new threads \&stopStoy;
                    $stopStoyTraad->join;

                    #Stoy er stoppet
                    $SamtidigStoy->join;
                    skrivTilResultat "\n";

                    #Ved siste runde trengs ingen pause
                    if($i!=(@stoyString-1)){
                          print hentKlokkeTid." - $pauseTid sek pause for neste
kjoring\n";
                          #Pause for teste runde med kjoring
                          sleep($pauseTid);
                    }
             }
}
```

STØY

cpuLastKontroll.pl

```
#!/usr/bin/perl
#Koden innholder kilde fra:
# http://www.guymal.com/mycode/generate_random_string/
# http://www.pil.sdu.dk/1/until2039-12-31/generate_2009-09-07.pl.html

use strict;
#Sending av data over nettverk
use IO::Socket;
#Tid for pause
use Time::HiRes qw( usleep nanosleep );
use threads;
use threads::shared;
use Time::HiRes;


use constant PROSENTAVVIK => 3; # +/- Avik fra CPU belastning i %
use constant PORT => 1500; # Nettverksport for styring
use constant STABIL => 7; # Hvor mange treff forhaand til % før den regnes som
stabil

my $antTraader = 0;
my $tid; # i sekunder

my $prosentLast = 0;

#Forteller naar stoy skal slutte med aa bli 1
my $avsluttTraader : shared = 0;
#Tekst lengde paa string
my $rtxtLengde : shared = 0;
#pasue mellom kjoring
my $soveTid : shared = 2;
#Array med liste over traader
my @cpuTraader;
#Trad for stoy
my $ioTraad;
#skal io være med i belastningen
my $ioPaa = 0;

#Funksjon for uthenting av totalcpulast
sub cpuLoad{

	my ( $userdiff, $idlediff, $totaldiff, $sysdiff);
	for ( my $i = 0; $i<2; $i++)
	{	#Avlesing av prosses status
		open STAT, "</proc/stat";
		while (<STAT>)
	{
		#Vi onsker kun infomasjon om cpu
	  if ( $_ =~ m/cpu\s/ )
		{
			#Les ut cpu belastninger
		my ( undef, $user, $nice, $sys, $idle, undef ) = split /\W+/, $_;
		$userdiff = ($user + $nice) - $userdiff;
		$idlediff = $idle - $idlediff;
		$sysdiff = $sys - $sysdiff;
		$totaldiff = ( $user + $nice + $sys + $idle ) - $totaldiff;
		}
	}
	close STAT;
	#Vent 1 sek til teste belastnings avlesing
	sleep 1 if $i == 0;
	}

	#Finn forbruk i prosent
```

```
	my $userp = $userdiff * 100.0 / $totaldiff;
	my $idlep = $idlediff * 100.0 / $totaldiff;
	my $sysp = $sysdiff * 100.0 / $totaldiff;

	#Total cpulast
	my $totalLast = int($userp+$sysp);

	#Hvor mye avik i prosent last godkjennes
	my $avvik = PROSENTAVVIK;

	#Justerings opp for belastning
	my $faktor = $prosentLast / ( $totalLast + 0.1 );

	#CPU prosent for hoy. belastning maa justeres ned
	if($totalLast > ($prosentLast+$avvik)){
		#Juster random string legde
		$rtxtLengde -= int( 50/$faktor );
		#Juster pause mellom hver generering av string
		$soveTid += int (20/$faktor);
		print "Høy $totalLast - $rtxtLengde - $soveTid - $faktor\n";
	}

	#CPU prosent for lav. belastning maa justeres opp
	if($totalLast < ($prosentLast-$avvik)){
		#Juster random string legde
		$rtxtLengde += int( 50/$faktor );
		#Juster pause mellom hver generering av string
		$soveTid -= int (20/$faktor);
		print "lav $totalLast - $rtxtLengde - $soveTid - $faktor\n"
	}

	#CPU prosent er innenfor godkjent min/maks
	if(($totalLast > ($prosentLast- PROSENTAVVIK )) && ($totalLast <
($prosentLast+3))){
		print "Stabil $totalLast - $rtxtLengde - $soveTid\n";
		#funn
		return 1;
	}

	#Maa justeres
	return 0;
}

#Generer string verdi
#Kilde: http://www.guymal.com/mycode/generate_random_string/
sub generate_random_string
{
	#Lengde for string
	my $length_of_randomstring=shift;

	#Innhold i string
	my @chars=('a'..'z','A'..'Z','0'..'9','_');
	#String som generes
	my $random_string;

	foreach (1..$length_of_randomstring)
	{
		#Finn tilfeldig verdi
		$random_string.=$chars[rand @chars];

	}
	#Returner funnet stringverdi
	return $random_string;
}

#Generer cpu last, denne regulerers med $rtxtLengde og $soveTid
sub cpulast{
	print "Traad $_[0] Startet\n";
```

```
        #Saa lenge stop ikke er satt. Generer last
        while($avsluttTraader != 1){
                #Lag en tilfeldig string etter angitt lengde
                generate_random_string($rtxtLengde);

                #Pause mellom hver last kjoring
                usleep($soveTid);

        }

        print "Traad $_[0] Avslutter\n";


}

#Skriv randomdata til fil
sub ioLast{
        print "Traad IO last Starter\n";
        #Saa lenge belastning ikke skal avsluttes
        while($avsluttTraader != 1){

#Noe kode kilde fra:
# http://www.pil.sdu.dk/1/until2039-12-31/generate_2009-09-07.pl.html

                my $linjeData = 11;
                my $mengdeIByte = 0;
                my $mengdeIMb = 0;

                #Skriv til fil data.dat
                open (DATAFIL, '>data.dat');
                #Fil overgaar ikke 20 og stoy skal ikke avslutte
                while ($mengdeIMb<20 && $avsluttTraader != 1){
                        my $linje = "";
                        for (my $i = 0; $i < $linjeData; $i++){
                                my $tTall = rand 10;

                                my $tTallStr = "$tTall ";

                                $linje = $linje . $linje . $tTallStr;
                        }

                        $linje = $linje . "\n";
                        my $linjeLength    = length( $linje);
                        $mengdeIByte     += $linjeLength;
                        #Regn ut datamegde i mb
                        $mengdeIMb = $mengdeIByte / 1024 / 1024;

                        #Skriv data til fil
                        print DATAFIL $linje;
                        #Lav stoy ved harddisk skriving
                        usleep(7000);

                        }
                        #Lukk fil for ny kjoring eller avslutting
                        close (DATAFIL, '>data.dat');
                }
        print "Traad IO last Slutter\n"
}

sub finnBelastning{
        #antall riktig cpu prosent som maa treffes for a regnes som stabil
        my $stabilTeller = STABIL;
        #Total randomstringlengde funnet ved stabile treff
        my $txtlengde_stbl = 0;
        #Total sovetid funnet paa stabile treff
        my $sovetid_stbl = 0;

        #Skal harddisk skriving vaere med i testen
```

```
        if($ioPaa == 1){
                #Start harddisk skriving
                $ioTraad = new threads \&ioLast;
        }

        #Antall traader for cpu belastning
        for(my $i=1; $i<=$antTraader; $i++){
                #Start cpu belastnings traad
                $cpuTraader[$i] = new threads \&cpulast, $i;

        }

        #Forsett saa lenge det er behov for flere stabile treff
        while($stabilTeller != 0){

                #Stabil treff funnet
                if(cpuLoad == 1){
                        #Legg sammen tekst lengde
                        $txtlengde_stbl += $rtxtLengde;
                        #Legg sammen sovetid
                        $sovetid_stbl   += $soveTid;

                        #Ferre stabile treff aa finne
                        $stabilTeller--;
                        }
        }

        #Be alle belastnings traader om aa avslutte
        $avsluttTraader = 1;
        print "Regnet som stabil\n";

        #Avslutter alle tradene
        for(my $i=1; $i<=$antTraader; $i++){
                $cpuTraader[$i]->join;
        }
        if($ioPaa == 1){
                $ioTraad->join;
        }

        #Alle traader avsluttet. Ved flere kjoring skal ikke stoy stoppes
        $avsluttTraader = 0;

        $stabilTeller = STABIL;

        #Finn gjennomsnitt for cpu belastnings verdier
        my $snitt_txt = int ( $txtlengde_stbl / STABIL );
        my $snitt_sov = int ( $sovetid_stbl / STABIL );
        return ( $snitt_txt, $snitt_sov );

}

#Kun kjoring av stoy med bestemte stoy verdier
sub startBelastning{
        my($tidBestemt) = @_;

        #Hvis valg for testen
        if($ioPaa == 1){
                #Start io
                $ioTraad = new threads \&ioLast;
        }

        #Start alle stoytraader for testen
        for(my $i=1; $i<=$antTraader; $i++){
                #Start stoy traad
                $cpuTraader[$i] = new threads \&cpulast, $i;
        }

        #Hvis tid der satt
```

```
        if($tidBestemt==1){
                #Vent antall sek
                sleep $tid;
                #Stop belastning etter at tid har gaatt ut
                $avsluttTraader = 1;
        }

        #Vet til alle traader er stoppet
        for(my $i=1; $i<=$antTraader; $i++){
                $cpuTraader[$i]->join;
        }

        #Vet til iotraad har stopped dersom det er kjort i testen
        if($ioPaa == 1){
                $ioTraad->join;
        }
        $avsluttTraader = 0; #Traader maa få lov til å kjore neste gang

}


while(1){

#Lytt paa angitt port
my $server = IO::Socket::INET->new( Proto     => 'tcp',
                                    LocalPort => PORT,
                                    Listen    => 1,
                                    Reuse     => 1);
die "Error: Kan ikke sette opp server paa port\n" unless $server;

#Vent paa ekstern oppkobling
my $kontroll_port = $server->accept();

# EKSEMPEL
# Finn 50% med IO disk skriving
# Kommando: "4 120 1000 50 0 1" = 4 tråder, 120sek, 1000 tegn per generering, 50 %
belasting, 0 (Ukjent usleep, finn den), 1 = IO ta med disk skriving
#
# Kjor med 50 % CPU last og disk skriving - 3297 (usleep) funnet med først nevte
kommando
# 4 120 1000 50 3297 1

#Vent paa mottat tekst linje
while(<$kontroll_port>) {
        ($antTraader, $tid, $rtxtLengde, $prosentLast, $soveTid, $ioPaa) = split
/\W+/,$_;

        print "$antTraader, $tid, $rtxtLengde, $prosentLast, $soveTid, $ioPaa\n";

        #Kjor stoy til mottatt stop kommando
        if($tid eq "evig"){
                my $tidBestemt = 0; # Vi skal kjore evig
                #Ta tid for kjoring
                my $start = [ Time::HiRes::gettimeofday( ) ];
                #Start cpu belastning
                my $belastningTraad = new threads \&startBelastning, $tidBestemt;
                # Alle mottatte meldinger stopper belastingen
                if(<$kontroll_port>){}
                lock($avsluttTraader);
                #Stopp stoy
                $avsluttTraader = 1;
                #Vent paa at stoy er avsluttet
                $belastningTraad->join;
                #Total kjoretid
                my $totaltid = Time::HiRes::tv_interval( $start );
                print $kontroll_port "FERDIG $totaltid\n";
        }elsif($tid == 0){
                #Sett tidsregning her - Start
```

```
                my $start = [ Time::HiRes::gettimeofday( ) ];
                print $kontroll_port "Starter Kalkulering\n";
                #Mottat anbefalt belastning for stoy prosent
                my ( $anb_txt, $anb_sov ) = finnBelastning;
                #Sett Stop tidsreging her - Stopp
                my $totaltid = Time::HiRes::tv_interval( $start );
                $totaltid;
                #Returner funnet verdier stoy
                print $kontroll_port "$antTraader $totaltid $anb_txt $prosentLast
$anb_sov $ioPaa\n";
        }else{
                #Kjoring av stoy er bestemt
                my $tidBestemt = 1;
                #Ta tiden fra start
                my $start = [ Time::HiRes::gettimeofday( ) ];
                #Start stoy med kjorelengdetid
                startBelastning $tidBestemt;
                #Tid for stoy kjoring
                my $totaltid = Time::HiRes::tv_interval( $start );
                print $kontroll_port "FERDIG $totaltid\n";
        }


}
close($kontroll_port);


}
```

stoyKlient.pm

```
package stoyKlient;
use strict;
use IO::Socket;
use threads;
use threads::shared;

use strict;
use Exporter;
use vars qw($VERSION @ISA @EXPORT @EXPORT_OK %EXPORT_TAGS);

$VERSION     = "1.00";
@ISA         = qw(Exporter);
#Funksjoner som skal vaere tilgjenglig fra start
@EXPORT      = qw(kjorStoy lesStoyConf stopStoy);
@EXPORT_OK   = qw(kjorStoy lesStoyConf stopStoy);
%EXPORT_TAGS = ( DEFAULT => [qw(&kjorStoy &lesStoyConf &stopStoy)]);

#Stopstatus variabel for stoy
my $stopStoy : shared = 0;

#Antall trader for stoykjoring
my $stoyTraader;
#Skal stoy kjore med traader
my $stoyMedIo;

#Addresse til stoyserver
my $stoyServer;
#Port for stoystyring
my $stoyPort;

#Filnavn for stoy konfig fil
use constant STOYKONFIG => "stoyKlient.conf";


sub lesStoyConf{
        #Bestemte og testet stoy verdier
```

```
my @stoyKjoreTid;
my $pauseTid;

#Lik Array lengde
my @stoyString;
my @stoySleep;
my @stoyProsent;


#Aapne stoy konfig fil
open FIL, STOYKONFIG or die $!;


while(<FIL>){


	my @linje = split /\s+/,$_;

	#Fyll opp stykonfigurasjon avhengig av linjetekst
	while(my $ord = shift(@linje)){
		if($ord eq "kjoretid:"){
			while(my $ord = shift(@linje)){
				#Stoykjoretid
				push(@stoyKjoreTid, $ord)
				}
			}
		if($ord eq "stoyString:"){
			while(my $ord = shift(@linje)){
				#Tekst lengde for hver kjoring

				push(@stoyString, $ord)
				}
			}
		if($ord eq "stoySleep:"){
			while(my $ord = shift(@linje)){
				#Sleep lengde for hver kjoring

				push(@stoySleep, $ord)
				}
			}
		if($ord eq "stoyProsent:"){
			while(my $ord = shift(@linje)){
				#Stoy prosent for hver stoykjoring

				push(@stoyProsent, $ord)
				}
			}
		if($ord eq "stoyTraader:"){
			while(my $ord = shift(@linje)){
				#Antall traader brukt i stoy

				$stoyTraader = $ord;
				}
			}
		if($ord eq "stoyMedIo:"){
			while(my $ord = shift(@linje)){
				#Skal stoy kjore med IO belastning

				$stoyMedIo = $ord;
				}
			}
		if($ord eq "stoyServer:"){
			while(my $ord = shift(@linje)){
				#IP adresse for stoy server

				$stoyServer = $ord;
				}
			}
```

```
                        if($ord eq "stoyPort:"){
                                while(my $ord = shift(@linje)){
                                        #Port for stoyserver

                                        $stoyPort = $ord;
                                        }
                                }
                        if($ord eq "pauseTid:"){
                                while(my $ord = shift(@linje)){
                                        #Pause tid mellom hver kjoring

                                        $pauseTid = $ord;
                                        }
                                }

                }

        }

        close FIL;

        #Lever hentet verdier til kjorende script
        return (\@stoyKjoreTid, \@stoyString, \@stoySleep,\@stoyProsent, $pauseTid);

}

#Funksjon for aa stoppe stoy under kjoring
sub stopStoy{
        lock($stopStoy);
        $stopStoy = 1;
}

sub kjorStoy{
        my($stoyProsent,$stoyTid,$genString,$nanoSleep) = @_;

        #Koble til stoy server
        my $socket = IO::Socket::INET->new(PeerAddr => $stoyServer,
                                        PeerPort => $stoyPort,
                                        Proto    => "tcp",
                                        Type     => SOCK_STREAM)
    or "Feil, port er opptatt eller annen nettverks feil\n";


        if($socket){
                #Send kjore parametere til stoyserver
                print $socket "$stoyTraader $stoyTid $genString $stoyProsent
$nanoSleep $stoyMedIo\n";
                #Ved evig kjoring maa det ventes paa stop fra test script
                if($stoyTid eq "evig"){

                        while ($stopStoy !=1){}

                        #Stopp stoyscript
                        print $socket "stop\n";
                }
                #Vent paa bekrefet stop status fra stoy server
                my $answer = <$socket>;
                my ($stoyStatus,$kjoreTid) = split /\s/,$answer;

                #Stoy er bekreftet stoppet lukk port
                close($socket);
        }

        #Sett tilbake stop variabel for flere kjoringer
        lock($stopStoy);
        $stopStoy = 0;
```

```
}
1;
```

stoyKlint.conf

```
stoyServer: 128.39.82.227
stoyPort: 1500

kjoretid: 300 900 1800 3600

stoyString: 1959 6308
stoySleep: 2016 303
stoyProsent: 30 80

stoyTraader: 3
stoyMedIo: 1
pauseTid: 120
```