Institutionen för kommunikation och information
Examensarbete i datalogi med inriktning mot nätverks- och systemadministration C 15hp
Vårterminen 2011

Automatiserad administration av virtuella plattformar

Jämförelse av prestanda i administrativa operationer mellan olika virtualiseringsplattformar

Anton Lindström

Automatiserad administration av virtuella plattformar

Examensrapport inlämnad av Anton Lindström till Högskolan i Skövde, för Kandidatexamen (B.Sc.) vid Institutionen för kommunikation och information. Arbetet har handletts av Jakob Ahlin.

2011-06-03

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

Signerat: _____

Automatiserad administration av virtuella plattformar

Anton Lindström

Sammanfattning

Virtualisering är något som växer fram mer och mer. Genom fördelar såsom användning av den fysiska hårdvarans fulla kapacitet och lägre energiutnyttjande är det en bra investering att flytta fysiska servrar till virtuella sådana. Automatiserade processer som exempelvis administration av virtualiseringsplattformar kan bidra till mindre arbetsbörda och mer utfört arbete. Det är då viktigt att administrationen går snabbt för att spara så mycket tid som möjligt och kunna utföra så många operationer som möjligt.

I arbetet jämförs två stora aktörer på virtualiseringsmarknaden, VMware ESXi samt Citrix XenServer. Dessa två virtualiseringsplattformar jämförs sedan prestanda i administrativa operationer. Jämförelser skedde med skript skrivna i programmeringsspråket Perl mot API:er och dessa utförde sedan de operationer som testades.

Genom tester mot virtualiseringsplattformarnas API:er med hjälp av skript visar resultaten att VMware ESXi har bättre prestanda i asynkrona anrop och Citrix XenServer har bättre prestanda i informationsinhämntningsoperationer.

Nyckelord: administration, virtualisering, automation

Innehållsförteckning

1 Introduktion

Martin Ford's bok, *The Lights in the Tunnel* (2009), grundar sig på hur ökad teknologi påverkar ekonomin på sikt men har stora förankringar i datalogivetenskap. I kapitel två skriver författaren under rubriken *Automation comes to the tunnel* att vid automatisering kan personal effektiviseras och ersättas med automatiserad arbetskraft som exempelvis robotar. På så vis kan ett företag öka lönsamhet genom att färre löner behöver betalas ut (Ford, 2009). Boken ser i detta kapitel på automatisering som en negativ del då personal avskedas och ger en negativ inverkan på ekonomin hos de anställda.

I ett senare kapitel nämner Ford (2009) att automation inte bara är något som påverkar arbetstagare utan också är något bra, där arbetsuppgifter kan skickas till annan part eller automatiseras. Ju lättare det är att automatisera desto fler företag kommer att implementera det. Genom att bryta ned stora arbeten till små operationer kan dessa automatiseras eller outsourceas. Automatisering bidrar därmed till mindre arbetsbörda och mer utfört arbete. En systemadministratör kan därmed frias från repetitiva arbeten som tar mycket tid till att hjälpa användare samt förbättra tjänster som denne utför.

Virtuella miljöer växer fram mer och mer i dagens samhälle och ser ut att bli en teknologi att räkna med i framtiden. 50% av tillfrågade företag i artikeln *IT managers under pressure to automate more* (Bednarz, 2010) har för avsikt att införa eller utveckla virtualisering. För att hålla en maskinpark konsistent samt erhålla en homogen miljö krävs automation. Av tillfrågade företag i ovan nämnda artikel vill 34% utöka eller införa automatisering (Bednarz, 2010). Genom automation finns en mall som följs genom hela skapandet av servrar när exempelvis skalbarheten är ett faktum. Automatisering av komponenter i en virtualiserad miljö kan också förebygga fel där systemadministratören har otillräcklig kunskap om hur en maskin är uppbyggd. En fullständig mall bidrar då till skydd mot problem som den mänskliga faktorn kan vara en orsak till.

Virtualisering används också i stor utsträckning på grund av att den fullständiga kraften hos ett datacenter inte används fullt ut vid vanlig allokering av tjänster vilka också används relativt sparsamt vid vissa tider. Nya datacenter byggs upp genom att se alla servrar som en stor pool av resurser för att på så vis ge olika applikationer tillgång till dessa (Merchant et al., 2007). Genom att minska serverkapacitet och på så vis minska strömförbrukning då lasten är låg ger detta en effektiv kostnadsbesparing . Virtualisering är idag en av den vanligaste åtgärden hos svenska myndigheter för att spara energi. Detta ger också en återbetalningstid, energimässigt på cirka 3 månader (Energimyndigheten, 2010).

Vid utskalning av ett system genom skript eller enkla gränssnitt finns möjligheten att skapa nya virtuella servrar genom mätvärden i exempelvis trafikökningar. Tider då trafik minskar till en tjänst finns möjligheten att minska lasten och lastbalansera fysiska servrar vilket bidrar till lägre strömförbrukning och stabilare servrar. Med hjälp av automatisering kan organisationer undvika risker såsom förlorad data och problem på grund av obalanserade virtuella maskiner (Hernick, 2010).

Detta arbete syftar till att bistå företag med rekommendationer om vilken virtualiseringsplattform som presterar bäst i administrativa operationer när dessa står i valet om att införa en ny plattform för virtualisering.

2 Bakgrund

För att tillgodogöra sig de begrepp samt tekniker som används i arbetet kommer dessa att redogöras i detta kapitel. Detta syftar till att ge läsaren en introduktion till virtualisering och andra, för arbetet relevanta definitioner.

2.1 Definitioner

Nedan ges förklaring av centrala förkortningar samt begrepp.

Asynkrona anrop: Även kallat icke-blockerande. När ett anrop skickats blockeras inte sändningen tills svar har mottagits. Används asynkrona anrop kan flera skickas simultant istället för att anropen köas upp (se *synkrona anrop*).

API: Application Programmable Interface innebär ett givet sätt att kommunicera med en mjukvara från andra program.

Prestanda: I detta arbete definieras prestanda som hur snabbt någonting utförs och sedan returnerar ett svar.

SDK: Software Development Kit, ett ramverk eller hjälpmedel i mjukvara för att underlätta utveckling mot specifika plattformar eller mjukvaror.

Synkrona anrop: Även kallat blockerande. Ett anrop skickas och blockerar en process tills denna är helt avslutad. Genom att använda synkrona anrop köas dessa upp efter varandra och körs en efter en tills de är alla exekverade.

Upptid: Tiden mellan start av dator till nuvarande tid.

2.2 Virtualisering

Virtualisering innebär att en fysisk server konsolideras, vilket betyder att den slås ihop med flera andra servrar under samma hårdvara för att möjliggöra effektivisering av prestanda samt strömförbrukning. Strömförbrukning minskas på grund av att färre fysiska servrar behövs och därmed minskar även behovet av storskalig kylning (Energimyndigheten, 2010).

Virtualisering kan delas upp i tre olika modeller: hypervisor, host-baserad och hybrid. Hypervisor-modellen har ett lager längst ner i systemet som hanterar all input och output (I/O) till enheter samt virtualisering av dessa. Den andra modellen kallas host-baserad och är en typ av virtualisering som använder en värd för styra hantering av tilldelade enheter. Detta medför att den virtuella maskinens egenskaper påverkas av värdens operativsystem. Den tredje modellen är den så kallade hybrid-modellen, denna använder sig av en liten hypervisor som hanterar virtualiseringsegenskaper i CPU och minne men lägger I/O hantering hos en värd (Nakajima et al., 2007).

De ovanstående modellerna kan också delas upp i mjukvaruvirtualiserade och hårdvaruvirtualiserade. Hypervisorer är då en form av hårdvaruvirtualisering och host-baserad är en form av mjukvaruvirtualisering. Det finns också en form av virtualisering som kallas virtuella behållare. Denna form skapar virtuella behållare och delar upp operativsystemet i flera partitioner. Detta har implementerats länge i UNIX-system och benämns ofta som BSD Jails. Virtuella behållare implementeras också i Solaris där det kallas Solaris Zones (Daniels, 2009).

För att underlätta begreppet virtualisering har IBM (2007) i sitt whitepaper *Virtualization in Education* förklarat det som följer:

> *"*Simply put, virtualization is a technique for hiding the physical characteristics of computing resources from the way in which other systems, applications or end users interact with those resources" (IBM, 2007, s. 2).

I detta arbete diskuteras virtuella maskiner (VM). I artikeln *Server Virtualization Architecture and Implementation* (2009) beskriver författaren att en virtuell maskin är en abstraherad miljö mellan den fysiska hårdvaran och användaren (Daniels, 2009).

2.3 Administrativa operationer i en virtuell miljö

I en virtuell miljö krävs flera operationer för att täcka alla behov för konfiguration av den virtuella plattformen. För att en virtuell miljö skall vara fungerande krävs dock endast operationer för initial installation av virtuella maskiner samt uppstart.

Då en initial installation endast sker en gång är inte prestanda av vikt för konfigurering och installation av virtuella maskiner. Däremot är hantering av dessa något som är viktigt. Framförallt kan uppstart och nedstängning av de virtuella maskinerna vara av vikt i en snabbt föränderlig miljö.

Övervakning är en annan viktig del av en systemadministratörs arbete. Problem kan oftast undvikas genom att undersöka hur system mår och vad som beskrivs i loggar. För att underlätta för administratören kan avvikelser i denna information inhämtas genom automatisering (Hein, 2010).

Operationerna att slå på och av virtuell ström samt inhämta information om virtuella servrar används då dessa är just alldagliga och används ofta av systemadministratörer. I arbetet begränsas tester till dessa operationer då de också är väldigt lika i de API som testas.

Relevanta operationer för arbetet är:

- Uppstart samt avstängning av virtuella maskiner.
- Informationsinhämtning från virtuella maskiner, upptid samt om maskinen är avstängd eller påslagen.

2.4 Metoder för kommunikation mot API

För att kommunicera med olika API över nätverket används olika protokoll och metoder för att överföra data. API är oftast dokumenterade och definierade av tillverkare av respektive tjänst för att på så sätt tillåta integration med andra tjänster eller produkter.

I boken *Web Services Essentials* (2001) beskriver författaren fyra olika tillvägagångssätt för kommunikation mot webbaserade API med Remote Procedure Calls (RPC) och XML över nätverk; XML-RPC, SOAP, UDDI och WDSL (Cerami, 2001). I detta arbete är det dock endast väsentligt att ha ett övergripande förstående för XML-RPC och SOAP.

XML-RPC är en enkel metod för att skicka RPC över nätverket. Metoden använder XML med HTTP för att skicka förfrågningar och hämta svar till och från en server. Denna teknik är dock relativt begränsad då den ej kan returnera data som objekt vilket behövs i mer avancerade applikationer (Cerami, 2001).

En annan metod för att möjliggöra RPC över nätverket är SOAP. SOAP är en vidareutveckling på XML-RPC och har stöd för både XML Schema och namnrymder. Dessutom innehar returdata från applikationer både anropad metod samt parametrar. SOAP är, till skillnad från XML-RPC, inte begränsat till HTTP utan kan användas med andra protokoll som till exempel SMTP. I specifikationen från W3C beskrivs dock endast transport via HTTP (Cerami, 2001).

2.5 Tidigare forskning

En genomsökning av en artikelsökmotor vid biblioteket i Högskolan i Skövde gav flera utslag vid sökning på automatisering av virtuella miljöer. I sökresultaten handlar det främst om behov att hantera processer genom automation på ett organiserat sätt. Detta är ett krav för att hålla en låg budget samt personalstyrka. Detta inkluderar också konfigurationshantering samt automatisering av uppdatering och installation (Bednarz, 2010).

Många undersökningar relateras ofta till de publika molnen (*Amazon EC2, Rackspace* etc.) där automatisering är en del av att automatiskt allokera de resurser som behövs vid en given tidpunkt. Applikationerna blir då horisontellt skalbara vilket medför att ju fler servrar som används ju större arbetslast klarar applikationen (Babu et al., 2009).

I studien *Automated Control in Cloud Computing: Challenges and Opportunities* (Babu, et al., 2009) beskrivs metoder samt effekter av horisontell skalning där kapaciteten ökar i förhållande till CPU-användning. Detta kan utföras med hjälp av flera olika policies vilka ger olika resultat i allokering av virtuella maskiner. Studien introducerar en ny sådan policy som till skillnad från föregående sådana kan bistå med en jämnare och bättre prestanda vid tilldelning av virtuella servrar (Babu et al., 2009).

Undersökning inom området finns också med inriktning på möjligheter att hantera resurser genom kontroll av virtuella servrar inom en enda plattform. Artikeln *Adaptive Control of Virtualized Resources in Utility Computing Environments* (Merchant et al., 2007) diskuterar Quality of Service (QoS) samt olika kontrollsystem för att använda så mycket kapacitet av datacentret som möjligt (Merchant et al., 2007). Hur olika plattformar hanterar kontrollen och hur dessa har möjlighet att hantera automatiserade funktioner ingår dock inte i undersökningen. Författarna medger också att mer arbete kring dynamiska kontroller bör utföras då de kontroller som testats endast var statiska. De nämner också migrering, delade resurser såsom minne, hårddisk, I/O samt nätverk och schemaläggare i Xen som områden som de vill undersöka djupare.

3 Problem

Nedan motiveras varför det generella problemet som uppstår vid administration av virtuella servrar är relevant att undersöka. Genom motivering av problem kan sedan en preciserad frågeställning nedtecknas och denna ämnar arbetet att ge svar på.

3.1 Problembild

Flera olika plattformar existerar idag för att virtualisera på. Detta arbete kommer fokusera på Citrix XenServer och VMware ESXi. Citrix XenServer bygger på en öppen-källkodsplattform, Xen, men har inkorporerats i en proprietär produkt (Citrix, 2011a). Den öppna plattformen används av exempelvis Amazon för att tillhandahålla virtualiserade servrar i molnet (*Amazon EC2*) (Amazon, 2011a). Xen är också mycket snabbt vilket stödjs av undersökningar i *Xen and the Art of Virtualization* (Barham et al., 2003) där författarna presenterar att plattformen endast har ett fåtal procent overhead i förhållande till icke-virtualiserade servrar (Barham et al., 2003). Xen har för närvarande inget gränssnitt som kan användas med exempelvis XML-RPC och därför används Citrix XenServer.

VMware ESXi är en annan virtualiseringsplattform där virtuella servrar hanteras utifrån en klient-mjukvara. En enkel version är gratis men flera funktioner får köpas till via licensiering. Detta är också den plattform som är mest utbredd.

Citrix XenServer och VMware ESXi täcks in i undersökningen då dessa representerar en stor del av de plattformar som används ute i företagen idag. Xen används som tidigare nämnts av Amazon (Amazon, 2011a) som har en av de största plattformarna för virtualisering med kunder som bland annat NASA och nyhetsföretaget The Guardian (Amazon, 2011b). Detta gör jämförelsen med denna plattform mycket intressant då den bör vara optimerad för prestanda i automation samt administration i en storskalig miljö. Eftersom Xen inte har ett fungerande XML-RPC API i skrivande stund, kommer Citrix XenServer användas då denna också är en plattform med kommersiellt stöd och support. Denna virtualiseringsplattform är en helhetslösning som implementerar Xen som virtualiseringsmotor och är mer lik VMware ESXi än bara Xen.

Andra virtualiseringsplattformar som exempelvis Microsoft Hyper-V har inte valts då detta skulle medföra en större undersökning vilket skulle ge mindre tid till undersökning av de valda plattformarna. Enligt Hermick (2010) kommer dock organisationer troligtvis att använda sig av flera olika plattformar i framtiden. Valet föll på VMware ESXi samt Citrix XenServer då dessa är som tidigare nämnts de största hypervisorerna i branschen. Andra virtualiseringsplattformar som följer andra modeller så som host-baserad virtualisering valdes inte då dessa inte möjliggör automatisering och administration i stor skala. Detta eftersom de oftast används med ett färre antal virtuella maskiner.

Flera operationer kan utföras mot det administrativa gränssnittet i virtuella plattformar. Operationer inkluderar att skapa nya virtuella servrar, starta dessa, ge dem mer eller mindre resurser (i form av RAM-minne, processorkraft etc), ändra nätverksinställningar, tillhandahålla information om virtuella servrar samt hämta dess prestandadata (VMware, 2011c).

För att bibehålla en konsistens i en virtualiserad miljö bör operationer ske på samma sätt varje gång (Chalup et al., 2007). Detta kan utföras genom att använda automatisering via funktioner som skapar exakt samma konfiguration varje gång det uppstår förändringar i miljön. Att hantera processer genom automation betyder även att fel som beror på mänskliga faktorn minskar eftersom exakt samma process utförs genomgående.

Miljöer som är virtualiserade kan växa och bli mycket komplexa. Därför är det viktigt att det finns information och översikt för att ge administratörer de detaljer som behövs för att veta var virtuella maskiner finns och vad de har för status. Hantering av virtuella servrar samt att tillhandahålla information om dessa är av stor vikt för att administrera virtuella miljöer (Hermick, 2010).

Skapande av virtuella servrar, exempelvis när ett system skall skalas ut bör kunna ske snabbt och effektivt. Nya servrar kan konfigureras på en halvtimme i motsats till 3 veckor som det kan ta för en fysisk server att installeras (Energimyndigheten, 2010). Förkonfigurerade, virtuella servrar kan ta kortare tid än en halvtimme att få i drift och kan ge organisationer en snabb expansionsmöjlighet.

Krav på att kunna skala upp och ned applikationer snabbt behöver mötas i en virtuell miljö (Armburst et al., 2009). Det krävs då att de administrativa gränssnitten är så snabba att de kan hantera arbete med stora mängder virtuella maskiner. När en applikation inte kräver så mycket resurser kan nedskalning av denna spara mycket energi och kan därmed bidra till minskad kostnad.

Denna undersökning kommer fokusera på prestanda hos den administrativa delen av virtuella plattformar. Detta skiljer sig betydligt från tidigare arbeten som snarare fokuserar på prestanda hos virtuella servrar i plattformarna. Ur en systemadministrationssynpunkt är det av vikt att alla administrativa processer fungerar och kan utföras så snabbt som möjligt så att tjänster som erbjuds i virtuella servrar inte avbryts.

Vid mätning av prestanda hos virtualiserade plattformar krävs att likvärdiga operationer utförs genom ett liknande gränssnitt för att testerna kan jämföras på ett rättvist sätt.

3.2 Frågeställning

Undersökningen ämnar att jämföra hur prestanda av automatiserad administration skiljer sig mellan virtuella plattformar för relevanta operationer.

3.3 Delmål

För att uppnå frågeställningen kommer ett antal delmål att besvaras. Dessa delmål är:

- Undersöka hur relevanta operationer kan tidsmätas mot virtualiseringsplattformar.
- Utvärdera skillnaden i tidsåtgång för operationer mot virtualiseringsplattformars API.
- Identifiera rekommendationer för organisationer i beslut om införande av virtualiseringsplattformar där administrativa operationer är av vikt.

Delmålet att undersöka hur relevanta operationer kan mätas mot virtualiseringplattformars API inkluderar att finna hur anrop mot API kan se ut och hur skript samt topologi skall byggas upp. Skript kan användas då en laborationsmiljö blir tillgänglig och allokerad. Detta inkluderar att fastställa topologi, allokera hårdvara samt bestämma mjukvara på klient-dator.

En väsentlig del är också att skriva skript för att utföra de operationer som beskrivs i bakgrundskapitlet. Dessa skrivs för att kunna göra de operationer som specificerats mot virtualiseringsplattformarna.

Genom syftet att bistå organisationer med råd och information om hur väl jämförda virtualiseringsplattformar presterar i administrativa operationer bör API-anropens tidsåtgång mätas.

3.3 Avgränsningar

För detta arbete kommer avgränsningar främst ske i form av vilka operationer som kommer användas för att få det förväntade resultatet (avsnitt 3.4). En ytterligare avgränsning sker också i operationer där dessa måste vara övergripande sådana som används i en miljö dagligen. Med övergripande menas att operationerna inte är specialiserade till en enda plattform eller en specifik implementation och används för att kunna få en virtuell maskin i ett startat läge.

3.4 Förväntat resultat

Resultatet skall utmynna i ett diagram där tydliga mätvärden presenteras. Detta kommer att bli en prestandatabell där data från tester mot virtuella plattformar redovisas. Resultaten kommer att presenteras i två kategorier där uppstart och nedstängning är en kategori och inhämtning av information den andra. Dessa värden kommer vara tid mätta i sekunder. För att uppnå syftet om att bistå med rekommendationer till företag som står inför ett val mellan virtuella plattformar bör en analys utföras av insamlad data.

4 Metod

Genom givna API skall ett antal operationer mot plattformarna Citrix XenServer samt VMware ESXi göras med administration i åtanke. Samma operationer skall användas mot de olika plattformarna för att ge mätvärden där en jämförelse kan ske. De administrativa operationerna skall genom en mängd förfrågningar kontrollera hur snabbt olika plattformar svarar.

För att genomföra arbetet kommer en studie utföras genom experimentell forskningsansats. Genom att jobba med skript mot virtualiseringplattformars API kommer detta liknas vid att implementera en modell för att köra simuleringar mot plattformens administrativa funktioner. Den experimentella forskningsansatsen finns beskriven i *Thesis Projects* (Berndtsson et al., 2008).

Experiment innefattar skript som används för att automatisera anrop mot de API som tillhandahålls av virtualiseringsplattformar. Anropen skall returnera information om virtuella maskiner på plattformen. Hur lång tid ett antal anrop samt returneringar tar skall bokföras och presenteras för att på så vis få en uppfattning om hur prestanda i automatisering förhåller sig mellan de två testade plattformarna.

4.1 Testmiljö

En testmiljö bör i detta fall innefatta en server där virtualiseringsplattformen är installerad samt en administratörsdator. Administratörsdatorn kommer att ha egenskrivna test-skript som mäter hur lång tid ett visst antal operationer tar. Mätdata hämtas sedan från den utdata som returneras från skriptet. Från denna data kommer sedan reflektioner nedtecknas.

De olika operationerna skall utföras på 10 stycken virtuella maskiner. Detta kommer att vara en mängd som är hanterlig för en stor mängd fysiska servrar men bildar ej för stor last på dessa. Informationen som hämtas skall hämtas 50 gånger för att sedan mäta vilken tid varje anrop tar i medelvärde. Operationerna kommer att mätas per gång, det vill säga, hur lång tid det tar för virtualiseringsplattformen att hämta information eller göra en operation på 10 virtuella servrar. Information om servrar hämtas 50 gånger då detta kommer att vara ett antal som gör det möjligt att beräkna medelvärde utan att ett enskilt värde gör för mycket skillnad.

4.2 Skriptutveckling

Dessa skrivs genom att till viss del använda vattenfallsmetoden där utveckling av skript sker i olika steg. Stegen i vattenfallsmetoden inkluderar systemkrav, mjukvarukrav, analys, programdesign, kodning, testning och verksamhet. (Royce, 1987). Vattenfallsmodellen används då den anses passa in i arbetet genom att utvecklingen kommer att ske i steg och begränsningar i arbetet gör att en iterativ process inte är möjlig.

I utvecklingen av skript kommer inte vattenfallsmetoden följas strikt utan kommer förenklas och kortas ned då detta arbete inte har programutveckling som fokus. Genom att skriptet inte är ett fullskaligt program kan också flera delar minskas. Då systemkrav och mjukvarukrav inte behöver vara skilda åt i denna implementation slås

dessa ihop för att bilda en generell kravspecifikation. Kravspecifikation, analys samt programdesign mynnar sedan ut i kodning, testning och verksamhet vilket presenteras under resultat.

4.2.1 Kravspecifikation

Kraven på de program som skrivs är att de relevanta operationerna, informationsinhämtning samt uppstart och nedstängning av virtuella maskiner fungerar mot de API som tillhandahålls av virtualiseringsplattformarna.

Skripten skall också fungera med både synkrona och asynkrona anrop i uppstart- och nedstängningsoperationer. Dessa skript skall också vara skrivna i programmeringsspråket Perl vilket VMware ESXi har stöd, samt guider för (VMware, 2011c). Citrix Xen har ej dokumenterat stöd för programmeringsspråket men använder XML-RPC vilket Perl innehar bibliotek för (Blackperl, 2011). Perl är ett språk som ofta används inom systemadministration och är tillgängligt för nästan alla operativsystem. Programmeringsspråket har även väldigt stor databas av moduler för att förenkla många uppgifter (Blank-Edelman, 2009).

Tester skall också ske med både kryptering genom SSL (HTTPS) och utan (HTTP). Detta för att fastställa hur stor påverkan kryptering har på operationer i administrativa gränssnitt. Om virtualiseringsplattformen inte stödjer HTTP eller HTTPS kommer endast referensvärde av den som stödjer protokollet användas. Detta referensvärde kommer att jämföras med det andra protokollet.

4.2.2 Analys

I API-dokumentationen för de båda virtualiseringsplattfomarna (VMware, 2011a & Citrix, 2011b) beskrivs API:er som är relativt lika. Ett anrop som är väldigt likt är exempelvis när virtuella maskiner startas. Citrix XenServer och VMware ESXi använder dock två olika former av RPC. Xen använder XML-RPC (Citrix, 2011b) medan VMware ESXi istället använder SOAP (VMware, 2011d).

VMware ESXi tillhandahåller ett toolkit för Perl (VI Perl Toolkit), vilket betyder att programmeringen sker enklare genom att fördefinierade rutiner redan finns inbyggda. Detta toolkit använder VI SDK som är plattformsoberoende då det baseras på diverse webb-tekniker. VI SDK har också möjlighet att kopplas till andra produkter från VMware vilket ger en stor portabilitet inom den egna produktportföljen (Jin, 2010).

Citrix XenServer har dock inget SDK men eftersom anropens utförande är relativt enkla krävs inga fler extra bibliotek än just XML-RPC. Båda plattformarna kan använda HTTPS vilket är till fördel för att säkert transportera information mellan server och klient (VMware, 2011d & Citrix, 2011b).

Genom att båda plattformars API använder två olika metoder för att skicka data i XML-form betyder det att dess funktionalitet skiljs åt. Dessa funktionella skillnader kommer att ingå i reflektion över prestanda i respektive virtualiseringsplattform.

Virtualiseringsplattformarna har möjlighet att både utföra synkrona och asynkrona anrop. I detta arbete kommer anrop att testas både synkront och asynkront. Detta används för att jämföra hur virtualiseringsplattformarna presterar i dem båda och hur

de skiljs åt. Eftersom arbetet utförs med prestanda av virtualiseringsplattformen i åtanke är det därför viktigt att skicka många anrop och sedan låta plattformen hantera dem och returnera svar. Detta kan utföras i asynkrona anrop. När alla svar returnerats är mätningen klar och det är denna mätdata som skall utvärderas. Asynkrona anrop bör ge en eventuell exponentiell ökning i tid eftersom mer processering av anrop sker i förhållande till en linjär ökning vid synkrona anrop. Information hämtas dock endast synkront då detta anrop endast stödjer synkron överföring av VMware ESXi.

4.2.3 Programdesign

Skriptet skall först anropa virtualiseringsplattformens API för att autentisera sig. När autentiseringen är utförd skall en session användas så autentisering ej behöver ske inför varje anrop.

Då det kommer vara flera versioner av skripten kommer de att fungera olika. Skripten kommer framför allt delas in i två olika typer, informationsinhämtning och uppstart samt nedstängning. Informationsinhämtning kommer att hämta detaljer om huruvida en virtuell maskin är avstängd eller påslagen samt hur länge denna har varit påslagen. Uppstart och nedstängning kommer istället att hämta information om den virtuella maskinen är påslagen eller avslagen. Vid påslaget läge kommer skriptet att skicka ett anrop som gör att den virtuella maskinen stängs av och vid avslaget läge kommer ett anrop sändas om att slå på maskinen.

När skript körs i asynkront läge kommer skriptet vänta in anropen så alla exekverats och returnerat svar. Detta medför alltså att när operationen är färdig avslutas skriptet. Genom att köra skript med Unix-programmet *time* kan tid som skriptet tagit för att exekveras visas. Detta är alltså vad som används för att se hur lång tid operationerna tagit. Unix-programmet mäter också den tid det tagit för hela processen att köra klart vilket gör att det är den tid det tar för systemadministratören att påbörja och slutföra en operation som är den huvudsakliga mätpunkten.

5 Resultat

I nedanstående avsnitt presenteras resultaten av den jämförelse av prestanda i administrativa operationer mellan de två virtualiseringsplattformarna som utförts. Prestandatestning presenteras enligt kapitel 3.4, förväntat resultat.

5.1 Utformande av testmiljö

I testmiljön sattes en skyddad topologi upp för att så lite nätverkstrafik som möjligt skulle störa den data som transporterades mellan server och klient. Genom att använda en skyddad miljö påverkades inte resultatet nämnvärt av nätverkets kapacitet utan påverkades istället av hur virtualiseringsplattformarna hanterar operationer.

Serverhårdvara som användes var HP Compaq dc7900 Convertible Minitower vilket inkluderade ett Intel Q45 chipset, en Intel Core 2 Duo E8400 processor med klockfrekvens på 3,0 GHz samt 800MHz 8GB DDR2 RAM. Klientens hårdvara baserades på HP Compaq dc7600 Convertible Minitower som bestod av ett Intel 945G Express Chipset med en Intel Pentium D 820 processor med klockfrekvens på 2,8 GHz och RAM-minne på 2GB DDR2 med busshastighet på 533 MHz.

I arbetet användes operativsystemet GNU/Linux Ubuntu 10.04 på klienten. Den Ubuntu-versionen som rekommenderades av VMware (2011b) hade tyvärr inte längre något stöd från det företag som utvecklar operativsystemet och användes därför inte. Eftersom Ubuntu 10.04 är en efterföljare till den version som VMware rekommenderade är detta vad som tillämpades i testmiljön. Denna version var också av typen LTS vilket står för Long Term Support. Detta betyder att operativsystemet har stöd en längre tid än exempelvis de nyare versionerna Ubuntu 10.10 samt 11.04. Genom att operativsystemet är en efterföljande version som VMware rekommenderade var denna relativt lik den föregående versionen.

5.2 Skriptutveckling

HTTPS-kommunikation fungerade inte i Perlskript med Citrix XenServer och kunde därför inte användas i testerna. Detta på grund av begränsningar i skript vilket skapade segmenteringsfel. Virtualiseringsplattformen har dock stöd för HTTPS men då det skapade problem i skriptet begränsades detta i implemetationen.

Exempelskript användes och modifierades för att kunna ge den funktionalitet som önskats. Skript-exempel fanns väl dokumenterat i VMwares dokumentation (VMware 2011d) och en del av detta användes i skriptet i bilaga A och B. Skript i bilaga C och D för Citrix XenServer var betydligt sämre dokumenterat för programmeringsspråket Perl. Exempelskript som hittades kunde modifieras och på så viss passa den tillämpning som önskats (Unixfoo, 2007).

Testning av skript utfördes för att kontrollera funktionalitet samt tidsdifferenser. När testerna slutförts och skripten var fungerande och höll konsistenta värden sattes de i verksamhet.

5.3 Prestandajämförelse

Nedan beskrivs de mätvärden som inhämtats från de skript som skrivits. Dessa värden skall svara på hur prestanda av automatiserad administration i virtuella plattformar varierar för relevanta operationer. Dessa operationer beskrivs i kapitel 2.3. Genom att utföra dessa operationer skall mätvärden utmynna i prestandatabell samt grafisk representation av dessa.

Efter sammanställning av information kan tydliga variationer urskiljas i resultatet mellan de båda plattformarna. Resultatet från dessa tester presenteras i Tabell 1 samt Figur 1. Genom att undersöka både synkrona och asynkrona anrop kunde en tydlig skillnad uppenbara sig. VMware ESXi har en tydlig distinktion mellan de båda anropstyperna och genom att använda asynkrona anrop kan administratören nå en ökad prestanda. I operationen för avstängning var det nästan 5 gånger snabbare att använda asynkrona än synkrona anrop. I uppstartsoperationen hade ESXi nästan dubbelt så bra prestanda i de asynkrona anropen.

I Citrix XenServer märkdes bara en mindre skillnad mellan synkrona och asynkrona anrop. Detta kunde då bero på att de asynkrona anropen hanterades som synkrona lokalt på servern vilket på så sätt skulle göra att anropen i praktiken alltid kommer att vara synkrona. Genom att skicka asynkrona anrop kommer endast det exekverande skriptet vara asynkront men i exekvering av operation på en server kommer dessa anrop köas upp och utföras efter varandra. Det finns också möjligheter att felaktigheter i det skript som skrivits gör att dessa operationer utförs med likartad tid som de asynkrona operationerna.

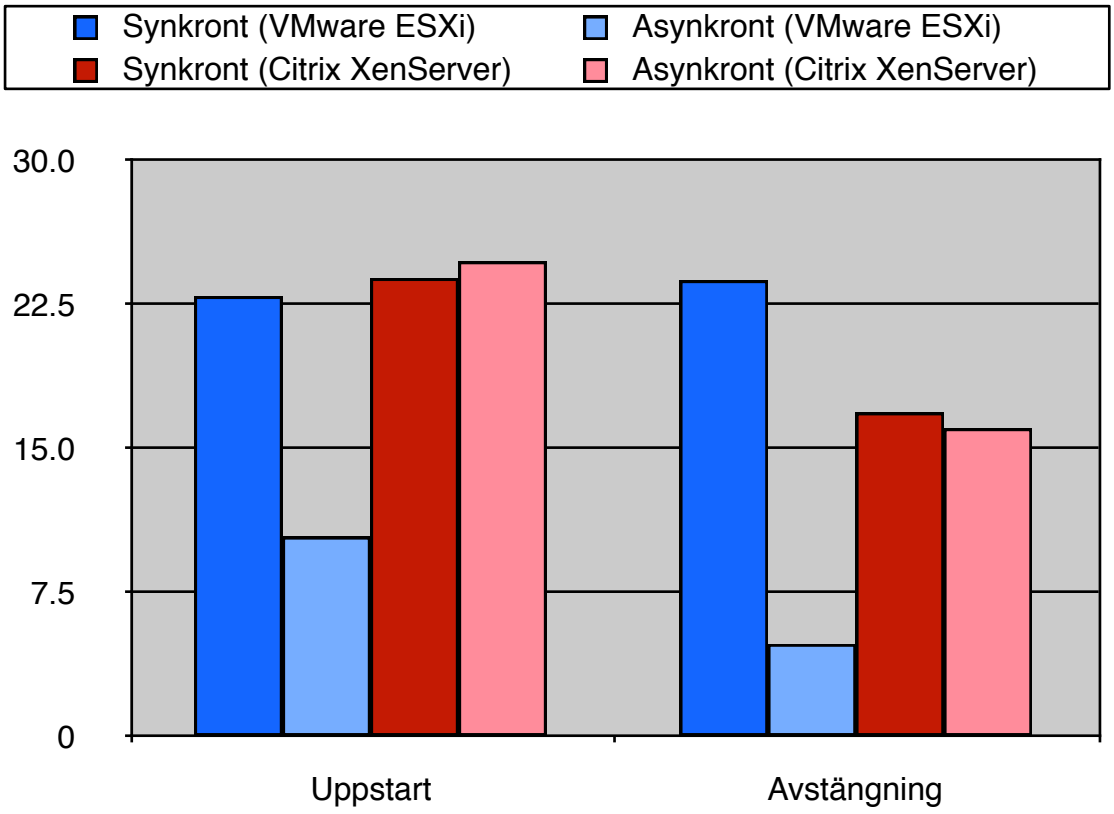
Tabell 1: Jämförelse av operationer genom HTTP.

	VMware ESXi	Citrix XenServer
Asynkron uppstart	10.4194 s	24.7538 s
Asynkron avstängning	4.8386 s	16.0632 s
Synkron uppstart	22.9442 s	23.8826 s
Synkron avstängning	23.7720 s	16.8858 s

Tabell 2: Jämförelse av operationer genom HTTPS.

	VMware ESXi	Citrix XenServer
Asynkron uppstart	10.4194 s	Inte tillgänglig
Asynkron avstängning	4.8386 s	Inte tillgänglig
Synkron uppstart	23.1462 s	Inte tillgänglig
Synkron avstängning	23.0734 s	Inte tillgänglig

Genom tester av uppstart och nedstängning av virtuella maskiner visade sig skillnader mellan avstängning och påslagning av de virtuella maskinerna i asynkrona anrop. Detta kan relateras till att det i praktiken tar tid att skicka in ström till en dator medan det går fort att bara bryta strömmen på fysiska maskiner. Vad som däremot förbryllar är att uppstart och nedstängning var relativt lika i synkrona anrop för VMware ESXi vilket kan ses tydligt i Figur 1. Lägre stapel i diagrammet påvisar högre prestanda och lägre tidsåtgång.



Figur 1: Jämförelse av operationer i HTTP.

En stor skillnad mellan virtualiseringsplattformarna var att Citrix XenServer tar signifikant längre tid i HTTP än vad VMware ESXi, framför allt på de asynkrona anropen. Prestanda i avstängning var däremot bättre i Citrix XenServer för synkrona anrop medan uppstart av virtuella maskiner är långsammare än i VMware ESXi.

Genom att testa både med och utan kryptering i VMware ESXi kan det konstateras att de värden som inhämtats i Tabell 1 samt Tabell 2 var relativt lika. Värden skilde relativt lite i synkrona anrop och inte alls i asynkrona. I resultatet kan då ses att HTTPS kunde användas utan märkvärd prestandaförändring. Att den synkrona avstängningen tog längre tid i HTTP är däremot ett konstigt fenomen. HTTP bör inte påverkas negativt då HTTPS lägger till krypteringsalgorimter till API-anrop mot virtualiseringsplattformen.

Tabell 3: Jämförelse av informationsoperationer genom HTTP.

	VMware ESXi	Citrix XenServer
Synkront	2.1466 s	0.5340 s

Tabell 4: Jämförelse av informationsoperationer genom HTTPS.

	VMware ESXi	Citrix XenServer
Synkront	2.2536 s	Inte tillgänglig

De informationsinhämtande operationer som utförts mot virtualiseringsplattformarna har en lägre tid vilket kan förklaras med att de endast läser information och inte behöver skriva eller ändra någonting. Att inhämta information om hur länge en virtuell maskin har varit igång samt om den är startad eller ej är operationer som kan behöva kontrolleras ofta i en stor miljö. Genom att inkludera flera informationskällor i virtualiseringsplattformarna kan detta stå till grund för bland annat kapacitetsplanering. Att mäta hur ett system mår är väldigt viktigt. Detta kan också ses på bilar som innehar flera olika mätvärden (bensinmätare, hastighet med mera) (Allspaw, 2008). Tillhandahållande av information om virtuella maskiner är också av stor vikt för att administrera virtuella miljöer (Hermick, 2010).

I tester med informationsinhämtning visar sig Citrix XenServer vara nästan 5 gånger så snabb som VMware ESXi. Då Citrix XenServer hade en så betydligt mycket högre prestanda i operationer för informationsinhämtning kan detta tyda på att användning av SOAP istället för XML-RPC kan ge mer avancerade operationer men till en kostnad av långsammare API.

I överföringar som använder SSL kan prestanda minskas med cirka 0.2-0.6 sekunder. Prestanda i SSL-anslutningar beror på bland annat serverkapacitet och filstorlek (Iyer et al., 2000). Eftersom VMware ESXi använder SSL-kryptering i HTTPS-anrop är detta direkt applicerbart på dessa. Resultaten visar dock andra värden. En förhandlad nyckel kan sparas och återanvändas under en session (Iyer et al., 2000). Detta skulle bidra till att efterföljande anrop använder samma förhandlade nyckel och ger endast första paketet ökad tid.

6 Analys

I detta kapitel analyseras det resultat som arbetet utmynnat i, den forskningsmetod som använts samt arbetets validitet.

6.1 Resultat

I testerna framgår det tydligt att VMware ESXi har varit den virtualiseringsplattform som har hanterat anrop snabbast i operationer med uppstart och avstängning i asynkrona anrop. Citrix XenServer har istället varit betydligt snabbare i informationsoperationer samt avstängning i synkrona anrop.

Då HTTPS inte används kan komplikationer uppstå i stora miljöer utan dedikerade administreringskanaler. Genom att inte skydda överföring kan användarnamn och lösenord avlyssnas vilket kan få förödande effekter för en organisation. Används dedikerade administreringskanaler, såsom ett separerat nätverk kan överföringen vara mer skyddad och på så vis istället få en ökad hastighet i operationer där tidsaspekt är av stor vikt.

Eftersom Citrix XenServer inte kunde testas med HTTPS fanns det ingen möjlighet att undersöka hur denna virtualiseringsplattform varierar i prestanda mellan HTTP och HTTPS. Tyvärr finns inte heller möjligheten att jämföra operationer över SSL mellan Citrix XenServer och VMware ESXi. Då HTTPS i VMware ESXi har så liten påverkan på prestanda jämfört med HTTP kan det mycket väl vara så att skillnaden i Citrix XenServer inte heller är speciellt stor mellan HTTP och HTTPS. Vilket skulle kunna validera att Citrix XenServer har bättre prestanda i informativa operationer även i HTTPS.

Genom att också ta hänsyn till säkerhet och användbarhet i skriptmiljö anser jag att VMware ESXi har en mer komplett produkt med högre säkerhet samt god dokumentation och är därför att föredra vid automatisering av administrativa operationer i virtuella plattformar. VMware ESXi användes också med ett SDK och Toolkit vilket inte undersöktes. Detta kan väl ha bidragit med hjälpmedel som gjort det enklare för utvecklare men till ett pris av prestandaförlust.

Båda virtualiseringsplattformarna finns i flera versioner där en av dessa är gratis. Citrix XenServer inkluderar i sin gratisversion ett API medan VMware ESXi inte inkluderar detta i gratisversionen. Detta kräver då en uppdatering. Flera andra funktioner såsom hög tillgänglighet och feltolerans kräver uppdatering på båda virtualiseringsplattformarna (Citrix, 2011a; VMware, 2011b).

Eftersom VMware ESXi inte inkluderar ett API i sin gratisversion kan detta vara en nackdel när små företag implementerar virtualisering med automatisering och skriptad administration. API:et för Citrix XenServer anser jag vara mer svårkonfigurerat och tar längre tid att sätta sig in i än det för VMware ESXi. Exempelskript i Perl för Citrix Xenserver mot dess API fanns inte heller officiellt dokumenterat.

I artikeln *IT managers under pressure to automate more* (Bednarz, 2010) beskriver artikelförfattaren att 50% av tillfrågade företag har planer på att virtualisera. Genom att visa att administrationen inte nödvändigtvis behöver vara extremt tidskrävande kan virtualisering med fördel introduceras i företag. Virtualiseringsplattformar minskar

inte heller prestanda från icke-virtualiserade servrar med mer än ett fåtal procent (Barham et al., 2003).

Genom att resultatet visar att Citrix XenServer är snabbare i informationsoperationer och VMware ESXi i påslagning och avstängning av virtuella maskiner kan virtualiseringsplattform väljas efter vilka operationer som är mest kritiska i företagets miljö. Möjligheten att använda flera virtualiseringsplattformar inom samma företag är också möjligt och kommer troligen användas i framtiden (Hermick, 2010).

6.3 Metod

I arbetet användes den experimentella forskningsansatsen som finns beskriven i *Thesis Projects* (Berndtsson et al., 2008). Med denna forskningsansats kunde de experiment som utförts med hjälp av skript validera vilken av de två virtualiseringsplattformarna som har bäst prestanda. Även implementationsmetod kunde användas som forskningsansats men då problemet inte är baserat på implementation av automatiserad administrering av virtuella plattformar utan snarare en jämförelse är den experimentella forskningsansatsen närmare det arbete som faktiskt skulle utföras.

Hade däremot en fallstudie utförts kunde flera användare kunnat besvara frågor om hur de upplever att prestanda i virtualiseringsplattformar varierar. I ett sådant fall finns det möjlighet att se vilken virtualiseringsplattform som användare föredrar i automatiserings- och administreringssyfte. Dock ger detta en subjektiv bild av prestanda och ger inga mätvärden som kan jämföras. Däremot kan en bredare bild av virtualiseringsplattformarna analyseras utifrån användares synpunkter.

När skript skrevs användes vattenfallsmetoden som utvecklingsmetod då denna innefattar metodsteg som var relevanta i arbetet. Vattenfallsmetoden är en icke-iterativ utvecklingsprocess som gör att arbetet med skripten blir successiva och varje metodsteg måste genomarbetas noggrant innan nästa steg påbörjas. Eftersom skript är relativt små kräver inte själva utvecklingen lika mycket arbete. Arbetet med kravspecifikation, analys och programdesign är dock desto viktigare i utvecklingsprocessen. Dessa steg krävs för att fastställa skriptets syfte samt hur det skall, i stora drag, fungera.

Metodvalen anses vara lämpliga för detta arbete och har bidragit till mer struktur. I fallet med vattenfallsmetoden har denna även bidragit till att varje steg har arbetats igenom noga och för att sedan inte behöva iterera tillbaka. Att inte iterera tillbaka kan ses som en nackdel i flera fall vid utvecklingsprojekt. I detta arbete anses dock att detta bidragit till ett noga genomarbetat förarbete för att sedan komma till utveckling där skriptens generella funktioner redan finns definierade.

Den experimentella forskningsansatsen fungerade mycket väl i sammanhanget. Variabler som användes vid experiment var främst operationer som skickades till virtualiseringsplattformarna vilket medförde resultat som varierade för de olika operationerna. Eftersom operationerna också kördes flera gånger ger detta mer robusta resultat och säkerställer en högre pålitlighet.

6.4 Validitet

Arbetets validitet bygger till stor del på skriptens uppbyggnad. Genom ovana i programmeringsspråket Perl är detta också något som kan ses som en stor brist. Skripten skrevs så att de skulle vara relativt lika samt fungera på liknande sätt men eftersom de båda virtualiseringsplattformarna fungerar på olika sätt är det mycket svårt att skriva dessa skript så de fungerar exakt likadant.

7 Slutsatser

Detta arbete har utförts för att ge underlag till beslut om virtualiseringsplattformar där prestanda i administrationsoperationer är av vikt. Genom att tidsmäta relevanta operationer samt utvärdera tidsåtgång i dessa kan rekommendationer för organisationer i beslut om införande av virtualiseringsplattformar identifieras. Tidsmätning samt utvärdering av dessa stod sedan till grund för att jämföra hur prestanda av automatiserad administration skiljer sig mellan virtuella plattformar i relevanta operationer.

För att tidsmäta relevanta operationer mot virtualiseringsplattformar användes skrivna skript mot XML-baserade API:er. På ett effektivt och automatiserat sätt kunde då tidsåtgången mätas för dessa operationer.

Tidsåtgång för relevanta operationer mot virtualiseringsplattformarnas API gav inblick i att VMware ESXi hade bättre prestanda genom att använda asynkrona anrop vid uppstart och avstängning av virtuella maskiner. Citrix XenServer hade däremot bättre prestanda i informationsoperationer samt avstängning i synkrona anrop. Asynkrona anrop i VMware ESXi var ungefär 5 gånger så snabba som de synkrona i de båda virtualiseringsplattformarna.

Genom att få en inblick i hur tidsåtgången för de båda virtualiseringsplattformarna såg ut kan också rekommendationer för organisationer i beslut om införande av virtualiseringsplattformar där administrativa operationer är av vikt identifieras.

Ett företag som är i behov av hög prestanda i uppstart samt avstängning av virtuella maskiner bör med fördel välja VMware ESXi då denna virtualiseringsplattform var betydligt mycket snabbare i dessa operationer vid asynkrona anrop. Är företaget i behov av hög prestanda i informativa operationer är Citrix XenServer ett bättre val. Citrix XenServer var nästan 5 gånger så snabb som VMware ESXi i informativa operationer.

Med dessa resultat kan slutsatser dras om att VMware ESXi presterar överlägset bäst när asynkrona anrop utförs. Synkrona anrop vid informativa operationer visar det sig däremot att Citrix XenServer presterar bättre. Slutsatserna medför att Citrix XenServer bör användas av företag där behoven att hämta information från virtualiseringsplattformar är högre än uppstart och avstängning av virtuella maskiner. VMware ESXi bör istället användas där uppstart och avstängning av virtuella maskiner är av större vikt än inhämtning av information.

Detta arbete kan också bidra med en insikt i hur mindre operationer kan integreras och automatiseras mot virtualiseringsplattformar genom skript. Arbetet belyser också skillnader mellan synkrona och asynkrona anrop i ett administrativt kontext. Genom att i administrativa operationer använda asynkrona anrop kan tid sparas och processer effektiviseras. Resultaten kan då ses som ett stöd för att välja asynkrona anrop i vissa virtualiseringsplattformar istället för synkrona. De resultat som framkommit kan med fördel användas av beslutsfattare, systemadministratörer samt utvecklare.

7.1 Framtida arbete

I framtida arbete bör fler operationer analyseras för att få en bredare bild om hur virtualiseringsplattformen presterar i varierande operationer och situationer. Även fler virtualiseringsplattformar än VMware ESXi och Citrix XenServer bör undersökas för att ge en bredare bild av hur prestanda i administrativa operationer hos virtualiseringsplattformar varierar.

Valet att använda programmeringsspråket Perl var på grund av stöd från VMware ESXi samt dess utbredda användning inom systemadministration. Genom att använda ett enda programmeringsspråk kan detta ha annorlunda implementation i olika moduler vilket kan ge ett avvikande resultat. I vidare forskning kan det vara av intresse att undersöka anrop med flera olika programmeringsspråk.

Citrix XenServer kunde inte användas med SSL i studien på grund av begränsningar i de Perl-skript som skrevs. En jämförelse mellan prestanda i operationer över SSL bör eventuellt också utföras då det är av stor vikt att anrop sker över säkra kanaler vilket tidigare nämnts.

I detta arbete har virtualiseringsplattformarna inte haft något system som kört och belastat systemet medan operationer kört. Eftersom ingen belastning ligger på virtualiseringsplattformen kan detta inte ge en autentisk spegling av en verklig miljö. Problemet skulle då vara att belasta de båda virtualiseringsplattformarna likartat utan att de skiljs åt.

Referenser

Allspaw, J. (2008). *The Art Of Capacity Planning.* Sebastopol: O'Reilly Media.

Amazon (2011). *Elastic Computing Cloud (EC2).* Tillgänglig på Internet: http://aws.amazon.com/ec2/. [Hämtad: 2011-06-02].

Amazon (2011). *Web Services Case Studies.* Tillgänglig på Internet: http://aws.amazon.com/solutions/case-studies/ [Hämtad: 2011-06-02].

Andersson C. (2010). *Projekt 3: Serverhallar.* Eskilstuna: Energimyndigheten

Armburst M., Fox A., Griffith R., Joseph A.D., Katz R.H., Konwinski A., Lee G., Patterson D.A., Rabkin A., Stoica I., Zaharia M. (2009). Above the Clouds: A Berkeley View of Cloud Computing. Technical Report No UCB/EECS-2009-28.

Babu S., Chase S. J., Lim C. H., Parekh S. S. (2009). *Automated Control in Cloud Computing: Challanges and Opportinities.* ACDC '09: Proceedings of the 1st workshop on Automated control for datacenters and clouds, 25-30.

Barham P., Dragovic B., Fraser K., Hand S., Harris T., Ho A., Neugebauer R., Pratt I., Warfield A. (2003). *Xen and the art of virtualization.* Proceedings of the 19th ACM Symposium on Operating Systems Principles (SOSP), 164–177.

Bednarz A. (2010). *IT managers under pressure to automate more*. Network World, 27(16), 12.

Berndtsson M., Hansson J., Lundell B., Olsson B. (2008). *Thesis Projects*. London: Springer-Verlag.

Blackperl, (2011). *RPC::XML*. Tillgänglig på Internet: http://www.blackperl.com/RPC::XML/ [Hämtad: 2011-06-02]

Blank-Edelman D. N., (2009). *Automating System Administration with Perl*. Sebastopol: O'Reilly Media.

Cerami E., (2001). *Web Services Essentials*. Sebastopol: O'Reilly Media.

Chalup R. S., Limoncelli A. T., Hogan J. C. (2007). *The Practice Of System and Network Administration.* Boston: Addison Wesley.

Citrix (2011). *XenServer features by edition.* Tillgänglig på Internet: http://www.citrix.com/English/ps2/products/subfeature.asp?contentID=2300456. [Hämtad: 2011-06-02].

Citrix (2011). *XenServer Management API.* Tillgänglig på Internet: http://support.citrix.com/article/CTX125244 [Hämtad: 2011-06-02].

Daniels J., (2009). *Server Virtualization Architecture and Implementation*. Crossroads, 16(1), 8-12. Tillgänglig på Internet: http://portal.acm.org. [Hämtad: 2011-04-06].

Ford M. (2009). *The Lights in The Tunnel.* Acculant Publishing

Hernick J. (2010). Managing Virtual Servers. *Information Week*, 1254, 34-36

Hein R. T., Nemeth E., Snyder G., Whaley B. (2010) *UNIX and Linux System Administration Handbook*. Boston: Prentice Hall.

IBM (2007). Virtualization in Education. Tillgänglig på Internet: http://www-07.ibm.com/solutions/in/education/download/Virtualization%20in%20Education.pdf. [Hämtad: 2011-06-02].

Iyer R., Kant K., Mohapatra P. (2000). *Architectural Impact of Secure Socket Layer on Internet Servers*. IEEE International Conference on Computer Design (ICCD'00), pp. 7-14.

Jin S. (2010). *VMware VI and VSphere SDK: Managing the VMware Infrastructure and vSphere.* Boston: Prentice Hall.

Merchant A., Salem K., Shin G. K., Singhal S., Uysal M., Padala P., Wang Z., Zhu X. (2007). *Adaptive Control of Virtualized Resources in Utility Computing Environments.* Proceedings of the EuroSys 2007

Nakajima J., Tian K., Wang W., Yu K., (2007). *How virtualization makes power management different.* Linux Symposium 2007. Tillgänglig på Internet: http://www.kernel.org/doc/ols/2007/ols2007v1-pages-205-214.pdf. [Hämtad: 2011-06-02].

Rackspace (2011). *Cloud Servers.* Tillgänglig på Internet: http://www.rackspace.com/cloud/cloud_hosting_products/servers/. [Hämtad: 2011-06-02].

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. Tillgänglig på Internet: http://portal.acm.org/citation.cfm?id=41801 [Hämtad: 2011-06-02].

Unixfoo (2007). *Perl Xen API.* Tillgänglig på Internet: http://unixfoo.blogspot.com/2007/12/perl-xen-api.html. [Hämtad: 2011-06-02].

VMware (2011). *API Reference.* Tillgänglig på Internet: http://pubs.vmware.com/vi-sdk/visdk250/ReferenceGuide/. [Hämtad: 2011-06-02].

VMware (2011). *Compare vSphere Kits*. Tillgänglig på Internet: http://www.vmware.com/products/vsphere/buy/small_business_editions_comparison.html. [Hämtad: 2011-06-02].

VMware (2011). *Installation Guide VIPerl*. Tillgänglig på Internet http://www.vmware.com/support/developer/viperltoolkit/viperl16/doc/viperl_install.pdf. [Hämtad: 2011-06-02].

VMware (2011). *Managing VMware ESXi*. Tillgänglig på Internet: http://www.vmware.com/files/pdf/vmware_esxi_management_wp.pdf. [Hämtad: 2011-06-02].

VMware (2011). *vSphere SDK for Perl Programming Guide*. Tillgänglig på Internet: http://www.vmware.com/support/developer/viperltoolkit/viperl40/doc/vsperl_40_proggd.pdf. [Hämtad: 2011-06-02].

Xen (2011). *Xen Home*. Tillgänglig på Internet: http://xen.org. [Hämtad: 2011-06-02].

Bilaga A

Operationer för att slå av och på den virtuella strömmen för VMware ESXi.

```
#!/usr/bin/perl
# Built on simpleclient.pl script from VMware Documentation

#use strict;
#use warnings;
use VMware::VIRuntime;
# Defining attributes for a required option named 'entity' that
# accepts a string.
 #
my %opts = ();
Opts::add_options(%opts);
# Parse all connection options (both built-in and custom), and then
# connect to the server
Opts::parse();
Opts::validate();
Util::connect();

my @status = ();
my $i = 0;

# Get all VirtualMachine objects
my $vm_views = Vim::find_entity_views(view_type => 'VirtualMachine');
# Send poweroff/on to virtual machines.
foreach my $vm (@$vm_views) {
  $status[$i] = powercycle($vm, "async");
  $i++;

}

$i = 0;

# Wait for every machine to finish.
foreach my $v (@status) {
  my $task_view = Vim::get_view(mo_ref => $status[$i]);

  while ( $task_view->info->state->val eq "running" ) {
    $task_view = Vim::get_view(mo_ref => $status[$i]);
  }
```

```
  $i++;
}

# Disconnect from the server
Util::disconnect();


# Turns power on if VM is off else on
#
# input, $vm and call type,
# call type either async or sync
#
# Returns status of call.
sub powercycle {
  my $vm = $_[0];
  my $call_type = $_[1];
  my $powertask;
  # Refresh the state of each view
  $vm->update_view_data();
  if ($vm->runtime->powerState->val eq 'poweredOff') {
    if ($call_type eq "sync") {
      $vm->PowerOnVM(); # Synchronous
      #print " Started virtual machine: ". $vm->name . "\n";
    } else {
      $powertask = $vm->PowerOnVM_Task(); # Asynchronous
    }
    print "on\n";
  } elsif ($vm->runtime->powerState->val eq 'poweredOn' ){
    if ($call_type eq "sync") {
      $vm->PowerOffVM(); # Synchronous
      #print " Stopped virtual machine: ". $vm->name . "\n";
    } else {
      $powertask = $vm->PowerOffVM_Task(); # Asynchronous
    }
    print "off\n"
  }
  return $powertask;
}
```

Bilaga B

Operationer för att informationsinhämtning för VMware ESXi.

```
#!/usr/bin/perl

# Built on simpleclient.pl script from VMware Documentation
use strict;
use warnings;
use VMware::VIRuntime;
use Date::Parse;
# Defining attributes for a required option named 'entity' that
# accepts a string.
 #
my %opts = ();
Opts::add_options(%opts);
# Parse all connection options (both built-in and custom), and then
# connect to the server
Opts::parse();
Opts::validate();
Util::connect();

# Get all VirtualMachine objects
my $vm_views = Vim::find_entity_views(view_type => 'VirtualMachine');
# Power off virtual machines.
foreach my $vm (@$vm_views) {
  # Refresh the state of each view
  $vm->update_view_data();
  print " VM: " . $vm->name .
  "\n\tPowerstatus: " . $vm->runtime->powerState->val . "\n";
  if ($vm->runtime->powerState->val eq 'poweredOn') {
    my $uptime = $vm->runtime->bootTime;
    my $seconds = time()-str2time($uptime);

    my @parts = gmtime($seconds);
    printf ("\t Uptime: %d days %d h %d min %d s\n",@parts[7,2,1,0]);

  }
}
# Disconnect from the server
Util::disconnect();
```

Bilaga C

Operationer för att slå av och på den virtuella strömmen för Citrix XenServer.

```
#!/usr/bin/perl


use Net::Ping;
use RPC::XML;
use RPC::XML::Client;

my $xenhost = $ARGV[0];
my $user = $ARGV[1];
my $passwd = $ARGV[2];

my $true = RPC::XML::boolean->new(1);
my $false = RPC::XML::boolean->new(0);

my $xenport = 80;

my $state = {
            "Running" => "ON",
            "Halted" => "OFF",
            "Suspended" => "SUSPENDED",
            "Paused" => "SUSPENDED",
            "Unknown" => "UNKNOWN"
            };


my %hash=();

# Establish the XEN API connection
my $xen = RPC::XML::Client->new("http://$xenhost:$xenport");
my $session = extractvalue($xen->simple_request
("session.login_with_password",
              $user,$passwd));

if (! $session) {
    print "connection failure : $xenhost\n";
    exit;
}

my ($host_ref,$domU_ref,$hostname) = ();

$host_ref = extractvalue($xen->simple_request
("session.get_this_host",
            $session,$session));

# Get the reference for all the VMs on the xen dom0 host.
$domU_ref = extractvalue($xen->simple_request("VM.get_all",
$session));

my ($vmtype,@domUarr) = ();
my @vm_tasks;

print "dom0 host - $xenhost\n";
# Loop through all VMs (including templates and Dom0).
foreach (@$domU_ref) {
  my $vm_ref = $_;

  # Filter out templates and DomU
  my $is_template = extractvalue($xen->simple_request
("VM.get_is_a_template", $session, $vm_ref));
```

```
  if ( $is_template eq 1 ) { next; };
  my $vmrecord = extractvalue($xen->simple_request("VM.get_record",
$session,$vm_ref));
  if ( $vmrecord->{domid} eq 0 ) { next; };

  # General VM info
  my $vmname = $vmrecord->{name_label};
  my $vmstate = $vmrecord->{power_state};
  $vmstate = $state->{$vmstate};

  # Shutdown if running, else power up.
  if ( $vmstate eq "ON" ) {
    #$xen->simple_request("VM.hard_shutdown", $session, $vm_ref); #
Synchronous
    my $task_ref = extractvalue($xen->simple_request
("Async.VM.hard_shutdown", $session, $vm_ref)); # Asynchronous
    push(@vm_tasks, $task_ref);
    print "Shutting down $vmname..\n";
  } else {
    #$xen->simple_request("VM.start", $session, $vm_ref, $false,
$true); # Synchronous
    my $task_ref = extractvalue($xen->simple_request
("Async.VM.start", $session, $vm_ref, $false, $true)); # Asynchronous
    push(@vm_tasks, $task_ref);
    print "Starting $vmname..\n";
  }
}

# Wait for _ALL_ tasks to complete
for ($i=0; $i<10; $i++) {
  my $status = extractvalue($xen->simple_request("task.get_status",
$session, $vm_tasks[$i]));
  while ( $status eq "pending" ) {
    $status = extractvalue($xen->simple_request("task.get_status",
$session, $vm_tasks[$i]));
  }
}

# Logout
$xen->simple_request("session.logout", $session);

sub extractvalue {
  my ($val) = @_;

  if ($val->{'Status'} eq "Success") {
    return $val->{'Value'};
  } else {
    return undef;
  }
}
```

Bilaga D

Operationer för att informationsinhämtning för Citrix XenServer.

```
#!/usr/bin/perl


# Xen API Programming.
# unixfoo - http://unixfoo.blogspot.com/

use RPC::XML;
use RPC::XML::Client;
use Date::Parse;

my $xenhost = $ARGV[0];
my $user = $ARGV[1];
my $passwd = $ARGV[2];

my $xenport = 80;

my $state = {
            "Running" => "ON",
            "Halted" => "OFF",
            "Suspended" => "SUSPENDED",
            "Paused" => "SUSPENDED",
            "Unknown" => "UNKNOWN"
            };


my %hash=();

# Establish the XEN API connection
my $xen = RPC::XML::Client->new("http://$xenhost:$xenport");
my $session = extractvalue($xen->simple_request
("session.login_with_password",
              $user,$passwd));

if (! $session) {
    print "connection failure : $xenhost\n";
    exit;
}
my ($host_ref,$domU_ref,$hostname) = ();

$host_ref = extractvalue($xen->simple_request
("session.get_this_host",
            $session,$session));

# Get the reference for all the VMs on the xen dom0 host.
$domU_ref = extractvalue($xen->simple_request
("host.get_resident_VMs",
            $session, $host_ref));

my ($vmtype,@domUarr) = ();

print "dom0 host - $xenhost\n";
foreach (@$domU_ref) {
    my $vm_ref = $_;

    # General VM info
    my $vmrecord = extractvalue($xen->simple_request("VM.get_record",
$session,$vm_ref));

    if ( $vmrecord->{domid} eq 0 ) { next; };
```

```
    my $vmname = $vmrecord->{name_label};
    my $vmstate = $vmrecord->{power_state};
    $vmstate = $state->{$vmstate};

    print " VM: " . $vmname . "\n\tPowerstatus: " . $vmstate . "\n";

    my $uptime = $vmrecord->{start_time};
    my $seconds = time()-str2time($uptime);

    my @parts = gmtime($seconds);
    printf ("\t Uptime: %d days %d h %d min %d s\n",@parts[7,2,1,0]);

}
sub extractvalue {
    my ($val) = @_;

    if ($val->{'Status'} eq "Success") {
        return $val->{'Value'};
    } else {
        return undef;
    }
}
```