Examensarbete

HÖGSKOLAN
I SKÖVDE

WEBBSERVERPROGRAM

Öppen källkods-alternativ till Apache

Examensarbete inom huvudområdet Datalogi
Grundnivå 15 högskolepoäng
Vårtermin 2012

Carlhåkan Svantesson

Handledare: Mikael Berndtsson
Examinator: Anders Dahlbom

Sammanfattning

Det har blivit allt vanligare för företag att marknadsföra sig via Internet vilket oftast innebär att företaget behöver en webbplats. Denna webbplats använder ett webbserverprogram för att hantera kundernars förfrågningar och det webbserverprogram som är störst på marknaden med god marginal är Apache. Apache har existerat i över 15 år och är öppen-källkod.

Det här examensarbetet undersöker om det finns några öppen källkods-alternativ till det marknadsledande webbserverprogrammet Apache genom att titta på funktionalitet och prestanda. Prestandatesterna har genomförts med både statiska och dynamiska webbsidor. De alternativ som undersöks är Nginx och Lighttpd.

Resultaten visar på att både Nginx och Lighttpd i det stora hela presterar bättre än Apache. Det här syns främst i de statiska prestandatesterna där Nginx och Lighttpd presterar mer än dubbelt så bra som Apache. I de dynamiska prestandatesterna så har Nginx och Apache jämförbar prestanda medan Lighttpd inte riktigt kommer upp i samma prestanda. Nginx saknar viss funktionalitet i jämförelse med de andra två, det är dock inga kritiska funktioner som saknas.

Nyckelord: Webbserver, Apache, Nginx, Lighttpd, öppen källkod

Innehållsförteckning

1 Introduktion

För många är Internet och Webben synonymer. Internet består av många olika servrar som tillhandahåller en stor mängd tjänster. Webben är bara en av de många tjänsterna som finns. Det är dock lätt att förstå varför många inte ser någon skillnad mellan de två. Enligt Halsall (2012) så kan en person använda en webbserver för att skicka e-post (webbpost), strömma film eller musik, hämta filer och mycket mer. En webbläsare har ofta flera program integrerade. Som exempel nyhetsgrupp-läsare, e-post- och filhämtningsprogram (Halsall, 2012).

Dock är Webben inte någon liten del av Internet. Statistik genererat av Internet Observatory (2012) visar på att ~31 % av Europas totala bandbredd är webbtrafik. Det är bara strömning av ljud- och videofiler på ~35 % som är större. En studie gjord av Sandvine (2011) visar på att det skiljer sig väldigt mycket mellan olika kontinenter. I t.ex. Nordamerika så står webbtrafiken enbart för 16.59 % av all bandbredd medan i Afrika så står det för 43.84 %.

En webbplats spelar en central roll i många företag. Det är deras ansikte utåt på Internet. Webbplatsen kan göra reklam för företaget eller ge information till kunderna. Det är ett smidigt sätt att sprida information om produkter. Allt från lösningar på vanliga problem till uppdateringar. Många företag använder även webbplatsen för att sälja produkter som komplement till deras butiker.

I en studie som utfört av MarketDirection (2010) har olika marknadsföringskanaler undersökts. Studien visar på att företag främst satsar på deras hemsida. Företagets hemsida är nästan dubbelt så viktig som marknadsföring via Internet (ej hemsida) som är på andra plats. Studien visar också på att företagen tror att deras hemsida fortfarande kommer vara den marknadsföringskanalen som de främst använder om 5 år.

Ett företag som använder sin webbplats till att göra reklam för sig själv är inte så beroende av att webbplatsen är uppe alla timmar på dygnet. Ett företag som inte har någon fysisk butik utan säljer alla sina varor via Internet och skickar dem direkt från lagret är däremot mer beroende av sin webbplats.

Det finns två vägar för ett företag som vill skaffa en webbplats: outsourcing eller införskaffa en webbserver själv. Outsourcing innebär att administration och underhåll av webbservern sköts av någon utomstående. Alternativet är att införskaffa nödvändig utrustning själv. Det är inte bara hårdvara och en internetförbindelse som behövs utan också mjukvara.

Väljer ett företag att administrera en egen webbserver så måste ett webbserverprogram införskaffas. Det finns dock väldigt många olika webbserverprogram att välja på (Greatstatistics.com, 2012a). De två största aktörerna på marknaden är Apache (The Apache Software Foundation, 2011) och Microsoft Internet Information Services (IIS) (Microsoft 2012). Netcraft LTD (2012) gör varje månad en utvärdering över vilka som är de populäraste webbserverprogrammen. Undersökningen visar på att av de en miljon populäraste webbsidorna använder 64.36 % Apache och av alla webbservrar så använder 64.91 %

Apache. IIS har en marknadsdel på 14.99 % av de en miljon populäraste webbsidorna och 14.46% av alla webbservrar. Detta betyder att nästan 80 % av de en miljon populäraste webbsidorna och nästan 80 % av alla webbservrar använder IIS eller Apache. Även andra undersökningar visar på att Apache och IIS har mellan 80-85 % av marknaden (Greatstatistics.com, 2012b; W3Techs, 2012; E-Soft Inc, 2012).

Vid införskaffande av ett webbserverprogram så är det alltid möjligt att välja det som är störst på marknaden. Dock bara för att ett webbserverprogram är störst på marknaden så innebär det inte att det har bäst prestanda. Den här studien undersöker om det finns några öppen källkods-alternativ till det marknadsledande webbserverprogrammet Apache som har bättre prestanda och bibehållen funktionalitet.

1.1 Problem

Apache har mellan 64-69 % av webbservermarknaden beroende på vilken undersökning som används (Netcraft LTD, 2012; Greatstatistics.com, 2012b; W3Techs, 2012; E-Soft Inc, 2012). Dock bara för att ett webbserverprogram är störst på marknaden innebär inte nödvändigtvis att webbserverprogrammet är bäst. Det kan finnas flera andra orsaker som gör att ett webbserverprogram är marknadsledande.

Det saknas vetenskapliga artiklar om just jämförelser mellan olika webbserverprogram. De vetenskapliga artiklar som finns tar upp hur uppbyggnad och funktionalitet påverkar prestandan hos ett webbserverprogram. Coarfa, Druschel och Wallach (2006) undersöker hur kryptering med Transport Layer Security (TLS) påverkar prestandan hos webbserverprogrammet Apache. Pariag et al. (2007) undersöker om det är någon skillnad mellan arkitekturerna: händelsebaserat, trådning och hybrid. Resultatet visar på att händelsebaserat och hybrid presterar 18 % bättre än trådning. Veal och Foong (2007) tittar på hur väl Apache skalar med flera processorkärnor. Studien visar på att vid fler än 4 processorkärnor så uppstår skalbarhetsproblem vilket främst beror på att adressbussen blir full.

Det har dock gjorts ett antal utvärderingar mellan olika webbserverprogram på bloggar (Williams, 2008; Dmitrij, 2011; Atkinson, 2008; Arnold, 2010). En del har bara tittat på funktionalitet medan andra har utfört prestandatester. På grund av detta så kan det vara svårt att få en komplett bild. Ett annat problem är att webbserverprogram hela tiden utvecklas vilket medför att ett test som utfördes för 3-4 år sedan är inaktuellet idag. Det kan även vara svårt att verifiera att testerna är gjorda på ett vetenskapligt sätt eftersom de inte alltid beskriver hur de har kommit fram till resultatet. Ytterligare ett problem är pålitligheten hos en blogg. Vissa bloggar används för att marknadsföra produkter (Svenska dagbladet, 2008) och det är inte alltid lätt att se kopplingen till ett visst företag.

De vetenskapliga studierna (Coarfa, Druschel och Wallach, 2006; Pariag et al, 2007; Veal och Foong, 2007) som har gjorts om webbserverprogram har fokuserat på hur prestandan påverkas i olika situationer. Det här tyder på att prestanda är en viktig egenskap för ett webbserverprogram. En ökning i prestanda får dock inte ske på bekostnad av förlorad funktionalitet. Detta främst för att en minskning i funktionalitet gör att användningsområdena

för webbserverprogrammet minskar. Det är därför viktigt att undersöka om det finns några alternativ till Apache som har bättre prestanda med bibehållen funktionalitet.

På grund av att det finns väldigt många webbserverprogram så finns det inte tid att utvärdera alla. Därför behöver ett urval göras. Appendix A visar en sammanställning av fyra undersökningar gjorda under februari 2012. På grund av att det finns väldigt många webbserverprogram (Greatstatistics.com, 2012a) så visar sammanställningen bara de mest använda. Utifrån denna sammanställning har ett antal kandidater valts bort.

Microsoft IIS valdes bort på grund av dess integration med Microsoft Exchange Server (Microsoft, 2005) samt att den är proprietär. Ett företag som använder Microsoft Exchange Server för e-posthantering kan genom integrationen med IIS lätt skapa ett webbaserat e-postsystem och är därför mindre benägen att använda något annat webbserverprogram. LiteSpeed är en proprietär programvara som tillhandahåller två olika versioner: en gratisversion och en betalversion (LiteSpeed Technologies, 2012). Gratisversionen har mindre funktionalitet och har inte heller lika optimerad prestanda som betalversion (LiteSpeed Technologies, 2012a). Det är därför svårt att genomföra en korrekt jämförelse om enbart gratisversionen används.

Google Web Server (GWS) används enbart internt av Google (Metz, 2010). Oversee Turing verkar det inte finnas någon information om alls. Det finns dock en del saker som pekar på att det är ett webbserverprogram som används för reklam (Oversee.net, 2012). Yahoo Traffic Server (YTS) donerades för över två år sedan till Apache Software Foundation (The Washington Post, 2009) och har bytt namn till Apache Traffic Server (The Apache Software Foundation, 2010). Riverbed Technology har slutat att utveckla Zeus Web Server (ZWS) (Riverbed Technology, 2011). IBM HTTP Server (IBM, 2012) är gratis dock inte öppenkällkod. Kvar finns Nginx (Nginx Inc, 2012), Lighttpd (Kneschke, 2012), Jetty (The Eclipse Foundation, 2012) och Cherokee (Ortega, 2012). Av dessa valdes de två mest använda Nginx och Lighttpd.

För att lättare kunna utvärdera webbserverprogramvarorna så delas problemet in i två frågeställningar.

- Hur presterar Nginx och Lighttpd i jämförelse med Apache i ett prestandatest?
- Har Nginx och Lighttpd samma funktionalitet som Apache?

1.2 Problemprecisering

Undersöka om Apache har några öppen källkods-alternativ på marknaden med bättre prestanda och bibehållen funktionalitet

2 Bakgrund

I denna del presenteras bakgrundsinformation som är viktig för läsaren. Den definierar även en del begrepp för att undvika förvirring.

2.1 Internet

Advanced Research Projects Agency Network (ARPANET) (Abbate, 1999) som var föregångare till Internet skapades av Advanced Research Projects Agency (ARPA) som är en myndighet under USA:s försvarsdepartement (Department of Defense). ARPANET skapades för att ansluta universitet och forskningsanläggningar som ARPA hade gett forskningsanslag till. Den första förbindelsen var mellan University of California, Los Angeles (UCLA) och Stanford Research Institute år 1969.

I mitten av 70-talet så hade ARPA tre olika nätverk: ARPANET, PRNET och SATNET. Alla tre både opererade och skickade information på olika sätt. Dock ville ARPA förbinda dessa tre nätverk. Lösningen var att skapa ett nytt protokoll så att de kunde utbyta information. Detta sätt att binda ihop nätverk kallades *internetwork* eller kort *internet*. Dock var det inte förrän ARPANET och NFSNET förbands i slutet på 80-talet som ordet Internet fick sin nuvarande betydelse (Abbate, 1999).

Internet är ett nätverk som sträcker sig över hela världen (Halsall, 2005). Det stödjer en mängd olika tjänster. Internet förlitar sig på en uppsättning protokoll kallad Transmission Control Protocol/Internet Protocol (TCP/IP) för kommunikation. De olika tjänsterna använder applikationsprotokoll som understödjs av TCP/IP. Exempel är e-post som använder Simple Mail Transfer Protocol (SMTP) för att skicka e-post. När två program kommunicerar med varandra över Internet så används portar. För att underlätta för användarna så har de flesta applikationsprotokoll därför en specifik port knuten till sig.

Domain Name System (DNS) är ett applikationsprotokoll som används av de flesta internetapplikationer för att slå upp värdnamns IP-adresser. Detta eftersom det är mycket lättare att memorera ett värdnamn än en IP-adress för människor. Föregångaren till DNS var att använda en *hosts*-fil på varje dator. Varje *hosts*-fil innehöll mappning mellan värdnamn och IP-adresser som fanns på nätverket. Detta blev dock snabbt ohanterligt när Internet började växa. En av de tjänster som gjorde att Internet började växa var Webben (Halsall, 2005).

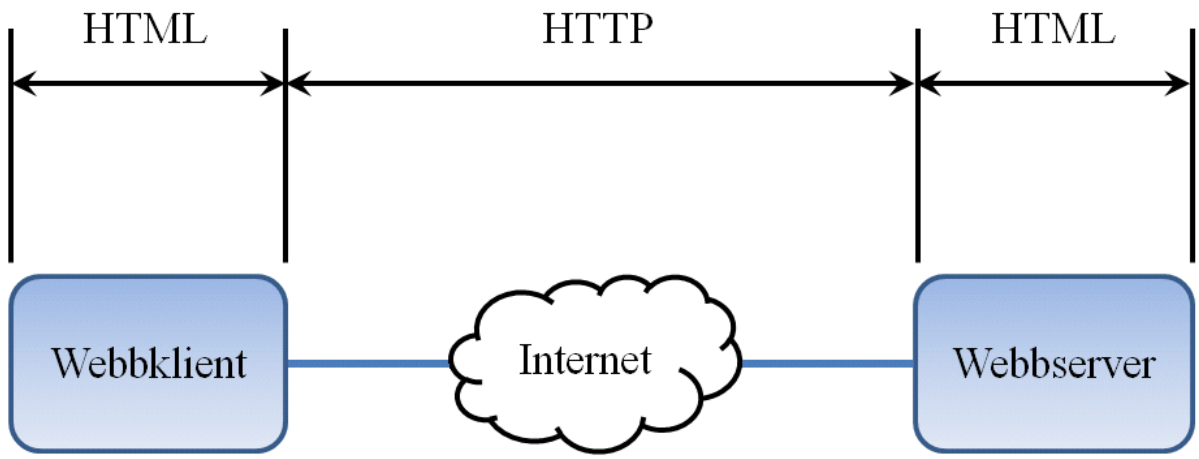
2.2 Webben

European Organization for Nuclear Research (CERN) (Gillies & Cailliau, 2000) är ett vetenskapligt forskningslaboratorium i Genève. Att hålla reda på dokument var ett stort problem vid CERN och ett som bara ökade allt eftersom experimenten blev mer komplicerade. Så 1984 började de utveckla ett program som skulle adressera dokumentproblemet. Resultatet var programmet CERNDOC som kördes på en stor, central IBM dator och kunde lagra tiotusentals dokument i en hierarkisk struktur. Det fanns dock flera problem med CERNDOC. Eftersom hela systemet kördes på en central dator och

använde terminaler för att komma åt informationen så var det inte så flexibelt. Det fanns inga fönster, ingen möjlighet att visa grafik eller ändra font. Vidare så saknades funktioner för att söka i enbart en viss del av hierarkin (Gillies & Cailliau, 2000).

Allt eftersom persondatorer började bli vanligare så ökade också behovet att kunna lagra bilder och grafer i dokument. Tim Berner-Lee såg alla dessa problem och började jobba på ett distribuerat system. I mars 1989 skickade han in ett förslag till CERN om ett nytt system för att lagra dokument (Berners-Lee, 1989). Detta var det första ritningarna på det vi nu kallar Webben. Han skapade även den första webbservern och webbklienten samt skrev den första versionen av Hyper Text Markup Language (HTML).

Webben består av webbservrar och webbklienter (se Figur 1). På webbservrarna finns information lagrad om allt från forskning till underhållning. Webbservrarna väntar på förfrågningar från webbklienterna. När en webbserver får en förfrågan så skickas informationen till webbklienten och återgår sedan till att vänta på andra förfrågningar (Halsall, 2005).



Figur 1 Kommunikation mellan en webbserver och en webbklient (anpassad från Halsall, 2005).

2.3 Webbklient

Webbklient även kallad webbläsare är den del som körs hos användaren (Gillies & Cailliau, 2000). Den första webbläsaren WorldWideWeb (senare Nexus) utvecklades 1990 av Tim Berner-Lee, Webbens fader. Webbläsaren hade grafiskt användargränssnitt men varje bild öppnades i ett nytt fönster. Flera webbläsare skapades och försvann i början på 90-talet. Det var inte förrän Marc Andreessen och Eric Bina på National Center for Supercomputing Applications (NCSA) skapade Mosaic som Webben verkligen började växa. Mosaic var mycket lättare att installera och använda än tidigare webbläsare. Vilket gjorde att även en nybörjare kunde använda Webben. Mosaic var även den första webbläsaren som hade bilder integrerade i samma fönster istället för att ha ett fönster för varje bild (Gillies & Cailliau,

2000). Idag är det främst tre webbläsare som används: Firefox, Internet Explorer och Chrome (StatCounter, 2012).

Webbklienten är den del som efterfrågar filer från webbservern och visar dem för användaren (Halsall, 2005). Är filen som efterfrågas en webbsida så måste webbläsaren även tolka informationen i filen. Webbsidor är skrivna i HTML vilket är det språk som används för att beskriva en webbsidas uppbyggnad. HTML används även för att integrera andra typer av objekt som t.ex. bilder eller Java applets. Dessa måste då hämtas hem separat från webbservern. Det finns flera andra språk som kan användas tillsammans med HTML för att göra webbsidan mer dynamisk. JavaScript och Extensible HTML (XHTML) är exempel på dessa. De laddas hem av webbklienten och exekverar lokalt på maskinen (Halsall, 2005).

2.4 Webbserver

Ordet webbserver har två betydelser. Först kan det vara den hårdvara som är värd för webbplatsen. Detta inkluderar operativsystemet och annan mjukvara. Den andra betydelsen är den mjukvara som hyser webbplatsen och kommunicerar med klienter. I den här studien används den första betydelsen. Mjukvarudelen benämns istället som webbserverprogram.

Även det första webbserverprogrammet utvecklades av Tim Berner-Lee (Gillies & Cailliau, 2000). Version 0.1 av CERN httpd släpptes i juni 1991. År 1993 så började NCSA utveckla ett webbserverprogram för att distribuera med den webbläsare som de höll på att utveckla. Marc Andreessen på NCSA tyckte att CERN httpd var för stor och komplex, och ville istället ha något mindre och lättare att förstå. Detta var början på NCSA httpd som senare blev Apache (Gillies & Cailliau, 2000).

En webbserver är ofta en kraftfull maskin eftersom den ska kunna kommunicera med många klienter samtidigt (Halsall, 2005). På webbservern så körs ett webbserverprogram som väntar på att webbklienter ska ansluta. När en webbklient ansluter så frågar den efter en specifik fil. Detta kan t.ex. vara en webbsida, en bild eller en video. Om webbklienten har behörighet och filen existerar så skickas den till webbklienten.

Det finns även möjlighet att skapa sidor dynamisk på webbservern. Detta sker med hjälp av Common Gateway Interface (CGI), PHP: Hypertext Preprocessor (PHP) eller några av de andra server-skriptspråken. Till skillnad från t.ex. Javascript som exekverar koden på klienten så exekveras koden istället på servern. Detta medför att det krävs mer av webbservern både vad gäller funktionalitet och prestanda (Halsall, 2005).

2.4.1 Apache

Apache (The Apache Software Foundation, 2011a) är en vidareutveckling av Rob McCool's webbserver, NCSA httpd. McCool lämnade NCSA i mitten av 1994 vilket gjorde att utvecklingen av NCSA httpd stannade upp. Detta medförde att många webbadministratörer utvecklade och fixade buggar helt på egen hand. Några av dessa webbadministratörer gick ihop för att bättre kunna koordinera utvecklingen. De skapade en e-postlista för att dela information samt började samla in alla buggfixar och testa dem. Detta var början till Apache. Den första publika versionen av Apache var 0.6.2 och släpptes i april 1995. Det var i princip

NCSA httpd med alla buggfixar och förbättringar applicerade. Mindre än ett år senare så var Apache den populäraste webbservern (The Apache Software Foundation, 2011a). I juli 1995 så skrev Robert Thau en ny kodbas som ersatte NCSA httpd. Även övriga funktioner portades till den nya kodbasen. Vilket ledde till att Apache 1.0 släpptes i januari 1996 (Mockus, Fielding & Herbsleb, 2000). Nu över 15 år senare så är Apache fortfarande den populäraste webbservern. Apache kan konfigureras på tre olika sätt genom Multi-Processing Modules (MPM) (The Apache Software Foundation, 2011).

- MPM Prefork - Hanterar förfrågningar genom att skapa en ny process för varje anslutning. Detta är en väldigt minneskrävande implementation men gör också att det inte behöver finnas någon support för trådning.
- MPM Worker - Skapar ett antal arbetsprocesser som använder trådning för att besvara förfrågningar.
- MPM Event – Är baserad på MPM Worker och skiljer sig bara genom att använda en huvud tråd för att distribuera ut förfrågningar. Denna modul är dock bara i experiment stadiet.

För att tolka PHP-filer med Apache så används en modul kallad *mod_php*. Den här modulen saknar dock stöd för att hantera trådning vilket gör att Worker istället måste använda ett externt program för tolkning av PHP-filer. Vid grundläggande installation så används PHP-CGI men det är även möjligt att använda något annat program. För att kommunicera med PHP-CGI så använder Worker *mod_fcgid* (FastCGI) vilket är ett gränssnitt för att anropa externa program. Apache har stöd för Windows, NetWare och Unix (The Apache Software Foundation, 2011).

2.4.2 Lighttpd

Lighttpd (Kneschke, 2007) uttalas *lighty* och skapades 2003 av Jan Kneschke. Det var först tänkt som en demonstration för hur man löser C10k-problemet. C10k står för ”Concurrent 10 000” och är hur en webbserver kan hantera 10 000 samtida anslutningar. Lighttpd är händelsebaserad och körs som en process utan trådning. Det finns dock möjlighet att använda flera processer för att sprida ut belastningen på flera processorkärnor. Precis som Apache Worker så har Lighttpd inte stöd för tolka PHP-filer internt så även här används PHP-CGI. Den har stöd för Linux och FreeBSD (Kneschke, 2007).

2.4.3 Nginx

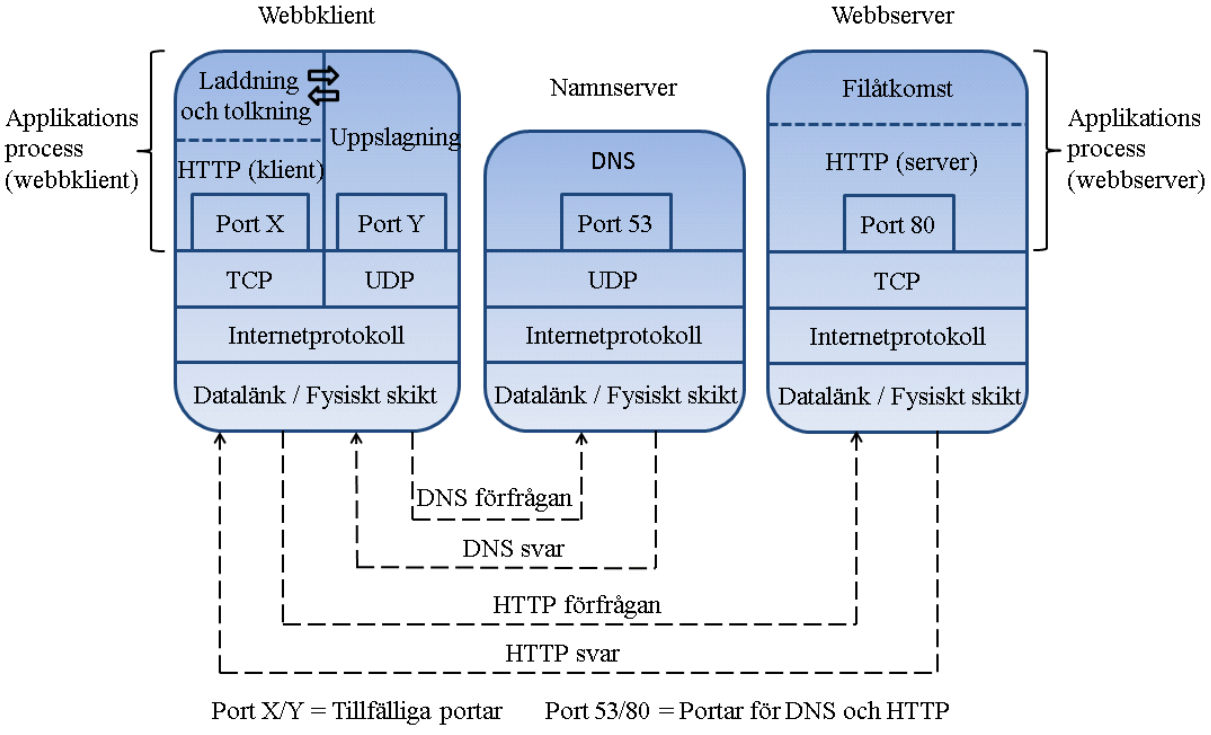
Nginx (Nginx Inc, 2012) uttalas *engine-x* och började utvecklas 2002 av Igor Sysoev. Det skapades för Rambler, Rysslands näst största webbplats, för att lösa C10k-problemet. Den första publika releasen var 2004. Nginx består av en huvudprocess och flera arbetsprocesser. Arbetsprocesserna använder inte trådning och körs under en vanlig användare. Precis som Lighttpd så är Nginx händelsebaserad. Nginx har heller inte något internt stöd för att tolka PHP-filer utan använder också PHP-CGI. Den har stöd för Linux, FreeBSD, Solaris och MacOS X (Nginx Inc, 2012).

2.5 Kommunikation

Kommunikation mellan webbserver och webbläsare sker med hjälp av protokollet Hypertext Transfer Protocol (HTTP) (Halsall, 2005). De tar hjälp av de underliggande Internetprotokollen Transmission Control Protocol (TCP) och Internet Protocol (IP) för att transportera informationen. HTTP är ett enkelt förfrågan-svars protokoll där webbläsaren skickar förfrågningar och webbservern svarar. En förfrågan är när webbläsaren antingen begär information eller skickar information till webbservern.

Figur 2 illustrerar de steg en webbläsare går igenom för att hämta information från en webbserver. Först kontaktas DNS-servern för att få reda på den IP-adress som värdnamnet använder. Webbläsaren kontaktar därefter webbservern med IP-adressen på protokollspecifika porten för HTTP vilket är 80. Sedan är det fritt fram för webbläsaren att efterfråga information. Det finns även möjlighet att kryptera all HTTP-trafik som skickas mellan webbläsare och webbserver. Då används istället HTTP Secure (HTTPS) på port 443.

När en webbläsare hämtar en webbsida så innehåller den oftast andra typer av objekt (ex: bilder och Java applets). Därför använder HTTP version 1.1 och senare ihållande-förbindelser. Istället för att skapa en förbindelse för varje fil som ska hämtas så används en förbindelse för alla filer (Halsall, 2005).



Figur 2 Kommunikation mellan webbläsare, DNS-server och webbserver (anpassad från Halsall, 2005).

3 Metod

Den här delen syftar till att förklara vilka metoder som kommer att användas för att utvärdera webbserverprogrammen. Huvudsakligen kommer två metoder att användas: experiment och litteraturstudie.

3.1 Prestandatest

Att använda prestandatester från tillverkarens webbplats är något som bör undvikas. Detta för att tillverkaren vill att deras produkt ska framstå i en så bra dager som möjligt. Det innebär inte nödvändigtvis att de skulle lägga ut falsk information utan att de enbart testar egenskaper som just deras produkt är bra på. Det är även möjligt att de har utfört ett test där deras produkt jämförs med någon konkurrents produkt. Dock så kanske bara de egenskaper där deras produkt presterar bättre presenterats.

Det är därför önskvärt att använda en mer neutral källa. Dock har väldigt få vetenskapliga jämförelser genomförts vad gäller just prestanda mellan webbserverprogram. De tester som har utförts har varit på bloggar där det är svårt att verifiera om de är neutrala och att de är genomförda på ett vetenskapligt sätt.

Istället för att förlita sig på prestandatester från tillverkare eller bloggar så utförs ett experiment. Laborationsmiljön som sätts upp för experimentet består av 2 datorer: en klient och en webbserver. På klienten körs testprogrammet och på webbservern körs webbserverprogrammet som ska testas. Webbservern kommer att använda tre hårdiskar, en för varje webbserverprogram. Detta för att säkerställa att de olika webbserverprogrammen inte påverkar varandra eller ändrar på inställningar som är globala för hela operativsystemet.

Prestandatesterna genomförs i två steg. I det första steget så installeras och konfigureras webbserverprogrammen utan någon som helst optimering. Detta för att se hur de presterar med förvalda värden. I andra steget så optimeras konfigurationsfilerna för den specifika hårdvaran. Detta eftersom det inte är möjligt att optimera ett webbserverprogram för alla typer av hårdvara vilket medför att jämförelsen blir mer likvärdig. Resultaten från första och andra steget jämförs sedan för att belysa eventuella skillnader.

Den hårdvarukomponent som påverkar ett webbserverprogram mest är Random Access Memory (RAM) (The Apache Software Foundation, 2012b). Andra hårdvarukomponenter som påverkar är CPU, nätverkskort och hårddisk. Eftersom ett effektivt webbserverprogram kan hantera fler förfrågningar samtidigt så är det svårt att tolka bandbredd (nätverkskort) och antal hämtade filer (hårdisk). Experimentet kommer därför titta på CPU- och minnesanvändning. Pariag et al. (2007) använder förfrågningar per sekund för att visa på skillnader mellan olika webbserverarkitekturer. Det kommer även att användas här för att visa på skillnader. CPU- och minnesanvändning kommer att avläsas från servern medan förfrågningar per sekund kommer att avläsas från klienten.

Ett webbserverprogram hanterar två typer av filer: statiska och dynamiska filer. Det är därför viktigt att testa båda dessa typer av filer. Varje steg delas således upp i två delar. Där den

första delen kommer att använda en statisk fil som genereras innan testet utförs och som innehåller slumpmässig data. Samma fil kommer sedan att användas av alla tre webbserverprogrammen för att minska antalet variabler som kan påverka resultatet. Testerna kommer att utföras genom att klienten hämtar samma statiska fil från servern flera gånger. Detta kommer ske under olika belastningar för att se hur det påverkar de olika webbserverprogrammen. Belastningen kommer att åstakommas genom att variera antalet samtidiga anslutningar som hämtar den statiska filen.

Den andra delen kommer att använda dynamiska filer. Dessa filer kommer att genereras av ett skript under prestandatestet. Detta betyder att alla tre webbserverprogram kommer att använda samma skript istället för samma dynamiska fil. Den andra delen av testerna utförs precis som den första med skillnaden att klienten hämtar den dynamiskt genererade filen istället för den statiskt genererade filen. Även olika belastningar kommer att användas här.

Istället för ett prestandatest så skulle en litteraturstudie kunna undersöka genomförda prestandatesterer. En stor nackdel med denna metod är att det saknas vetenskapliga prestandatester vilket medför att mer tvivelaktiga källor måste användas. Det här inkluderar bloggar, forum och tillverkarnas hemsidor. Fördelen är att flera typer av tester kan inkluderas i studien eftersom det tar mycket kortare tid att analysera ett test än att genomföra ett.

3.2 Funktionalitet

Funktionaliteten undersöks för att se om de två webbserverprogrammen saknar några funktioner i jämförelse med Apache. Det kan även vara en avgörande faktor om webbserverprogrammen har jämförlig prestanda. Funktionaliteten hos de olika webbservrarna kommer att undersökas genom att besöka tillverkarnas webbsidor och därifrån utläsa information om funktionalitet. Eftersom denna information kommer direkt från tillverkaren så finns det en risk att de försöker att få sin produkt att framstå i en så bra dager som möjligt. Dock eftersom det handlar om vad tillverkarens webbserver kan och inte kan göra så bör det inte vara något problem eftersom helt avsaknad av marknadsförd funktionalitet kan få väldigt mycket negativ publicitet.

Istället för en litteraturstudie så skulle interjuver med ett antal företag kunna genomföras för att få reda på vilken funktionalitet som är viktigt. Detta skulle göra att mer relevant funktionalitet skulle fås fram. Nackdelen är att om man pratar med en allt för begränsad mängd företag eller väldigt snarlika företag så kan viktig funktionalitet missas. Fördelen är att funktionalitet som inte används inte heller behöver presenteras. Detta medför att företag som är osäkra på vilken funktionalitet som de behöver inte behöver undersöka funktioner som inte används.

3.3 Förväntat resultat

Apache kan konfigureras för att använda processer eller trådning. Enligt The Apache Foundation (2011) så klarar den trådbaserade lösningen av fler förfrågningar än den processbaserade. Studien som genomfördes av Pariag et al. (2007) visade på att en händelsebaserad arkitektur presterade bättre än en arkitektur som använder trådning. Både

Nginx och Lighttpd använder en händelsebaserad arkitektur och bör därför prestera bättre än Apaches båda konfigurationer.

4 Genomförande

Gemonförande innehåller information om hur laborationsmiljön är uppbyggd, hur testerna genomförs och vilken funktionalitet som finns.

4.1 Laborationsmiljö

Laborationsmiljön består av två datorer: en klient och en server. De är sammankopplade med en korskopplad TP-kabel för att minska att andra faktorer påverkar prestandatesterna. Både klient och server använder en Intel Core 2 Quad Q9300 processor med klockfrekvens på 2.5GHz, 8GB RAM och Gigabit nätverkskort.

Operativsystemet som installeras på server och klient måste stödjas av alla tre webbserverprogrammen. Det är möjligt att använda flera olika operativsystem dock så ökar antalet faktorer som kan påverka prestandatestet. Det är därför att föredra att endast använda ett operativsystem till alla tre webbserverprogrammen.

De operativsystem som alla tre webbserverprogrammen har stöd för är FreeBSD och Linux. För att välja operativsystem så används en undersökning gjord av W3Techs (2012a). Den visar på att BSD (FreeBSD, NetBSD, OpenBSD, etc) endast har en marknadsandel på 1.2 % medan Linux har hela 32,8 %. Den största Linuxdistributionen är Debian med 9,9 % följt av CentOS på 9,4 %. Samtliga hårddiskar installeras därför med Debian 6 64bit. Vid installation av Debian används förvalda alternativ och endast grundläggande system paket installeras. Efter installationen så uppdatera systemet och webbserverprogrammen installeras.

4.1.1 Webbserverprogram

Apache kan konfigureras på tre olika sätt: Prefork, Worker och Event. Event är fortfarande i experimentstadiet men de andra två används i produktionsmiljöer. Båda dessa konfigurationer installeras och testas därför på separata hårddiskar. I tabellen nedan så visas vilka versioner av webbserverprogrammen samt vilken version av PHP som används.

Program	Version
Apache	2.2.16
Lighttpd	1.4.28
Nginx	0.7.67
PHP	5.3.3-7

4.2 Prestandatest

Prestandatestet genomförs med hjälp av Weighttp 0.3 som är utvecklat av en av programmerarna bakom Lighttpd (Icy, 2012). Weighttp är ett kommandotolk-verktyg som ansluter till ett webbserverprogram och hämtar en fil. Det finns möjlighet att specificera hur många gånger filen ska hämtas och hur många samtida anslutningar som ska användas samt om förbindelsen ska vara ihållande. Till skillnad från ApacheBench (The Apache Software Foundation, 2011b), som är ett vanligt förekommande program vid prestandatester av

webbservrar, så kan Weighttp använda flera processorkärnor. Detta medför att Weighttp kan belasta webbserverprogrammen mer vilket ger tillförlitligare prestandatester.

Prestandatesterna genomförs i två steg. I första steget så används medföljande konfigurationsfiler och i andra steget så optimeras konfigurationsfilerna för de olika webbserverprogrammen. Varje steg är sedan vidare uppdelat i två delar. Den första delen använder statiska filer medan den andra använder dynamiska filer. Detta påverkar dock inte hur Weighttp hämtar filerna eftersom de genereras av servern (webbserverprogrammet). Varje webbserverprogram testas med en fil under 4 olika belastningar. Belastningen består i att 10, 100, 500 och 1000 samtida förbindelser upprättas och hämtar filen flera gånger. 1000 samtida förbindelser valdes som övre gräns för att tester vid uppsättning visade på att det var då webbserverprogrammen började missa förfrågningar. På grund av att antal förfrågningar specificeras i Weighttp så är det svårt att köra prestandatestet en viss tid, dock så körs alla prestandatester minst tre minuter. Varje prestandatest körs tre gånger med en omstart av samtliga datorer mellan varje gång. Om något av de tre värdena avviker allt för mycket från resten så körs hela prestandatestet om. Resultatet är sedan medelvärdet av de tre värdena. Det skulle dock vara önskvärt att köra prestandatestet flera gånger för att få en högre statistisk säkerhet men inom ramen för det här examensarbetet så finns det inte tid till det.

I tabellen nedan visas de olika testerna som utförs.

Steg	Del	Samtida förbindelser			
Ej optimerad	Statisk	10	100	500	1000
	Dynamisk	10	100	500	1000
Optimerad	Statisk	10	100	500	1000
	Dynamisk	10	100	500	1000

Den statiska filen genereras innan prestandatestet med hjälp av *dd* som är ett av Debians grundläggande systemverktyg (GNU, 2012). För att generera slumpmässig data så används *dd* tillsammans med ”/dev/urandom”. De dynamiska filerna genereras med hjälp av ett skript på servern under prestandatestet. PHP är det mest använda serverskriptspråket (W3Techs, 2012b) och används av 77.9% av marknaden. Det här är följt av ASP.NET som används av 21.4%. På grund av den breda användningen av PHP så används det i den här studien. Skriptet som kan ses nedan genererar ”length” bytes.

```
<?php
$length=100;
$characters = '0123456789abcdefghijklmnopqrstuvwxyz';
for ($i=0;$i < $length;$i++) {
        echo $characters[$i%36];
}
?>
```

4.2.1 Filstorlek

Både de statiska och dynamiska filerna är 100B stora. Valet av filstorlek grundar sig i vad som är möjligt att testa. Om filstorleken är för stor så kommer nätverkskorten att nå max kapacitet innan webbserverprogrammet. Storleken på ett paket utan någon data (payload) varierar mellan webbserverprogram men ligger mellan 300B och 400B. Nedan visas beräkningar med filer som är 100B och 1000B. Kapaciteten på nätverkskorten är 1Gbit/s (1 000 000 000b).

```
1 000 000 000/((400+100)*8) = 250 000 Förfrågningar/sekund

1 000 000 000/((400+1000)*8) = 89 285 Förfrågningar/sekund
```

Det här betyder att om 1000B filer används så kan inte antal förfrågningar per sekund överstiga 89 285 för då har nätverkskortet nått max kapacitet. Så för att vara på den säkra sidan så används därför 100B filer vilket tillåter 250 000 förfrågningar per sekund. Bandwidth Monitor (bmon) som är ett verktyg för att övervaka bandbredd används för att säkerställa att inte nätverkskorten når max kapacitet.

4.2.2 Minneanvändning

Som tidigare nämnts så avläses minnesanvändningen av varje webbserverprogram på servern. Det finns två olika värden som associeras med minnesanvändning: Virtual SiZe (VSZ) och Resident Set Size (RSS). VSZ inkluderar allt minne som processen använder oavsett om det ligger i Random Access Memory (RAM) eller på hårdisken. Även minne som processen har efterfrågat men som ännu inte är allokerat av kerneln är inkluderat. RSS inkluderar enbart det minne som finns i RAM. I det här fallet så är RSS att föredra eftersom det viktiga är att undersöka hur mycket minne som utnyttjas just nu, inte hur mycket den har efterfrågat.

Dock finns det ett problem med RSS och VSZ, och det är att de inkluderar delade bibliotek (shared libraries). Detta gör att ett webbserverprogram som skapar en process för varje processorkärna eller en process för varje anslutning kommer att räkna minnesanvändningen för samma bibliotek flera gånger (en för varje process). För att komma runt detta problem så avläses istället den totala minnesanvändningen av hela systemet. Detta utgör inget problem eftersom alla webbservrarna använder samma hårdvara och operativsystem. För att avläsa den totala minnesanvändningen så används *sar*. *Sar* är en del av paketet ”sysstat” och kan samla in information om olika systemaktiviteter. Beräkning av minnesanvändning sker enligt formeln nedan.

```
kbmemused - (kbbuffers + kbcached) = minnesanvändning
```

Buffrar och cache tas bort för att det är minne som går att använda men som inte är markerat som ledigt. Anledning till detta är att om ett program körs flera gånger så behöver inte programmet laddas in i RAM varje gång utan finns kvar i cache. Skulle ledigt minnesutrymme ta slut så kommer däremot cache att skrivas över.

4.2.3 CPU-användning

CPU-användningen avläses precis som minnesanvändningen på servern. Till skillnad från minnesanvändning så är det här möjligt att avläsa CPU-användingen för varje

webbserverprogram. Dock eftersom en del webbserverprogram använder externa program för att exekvera PHP-kod så är det nödvändigt att avläsa den totala CPU-användningen. Även här används *sar* från paketet ”sysstat”.

4.2.4 Optimering av operativsystemet

Ett operativsystem som inte är optimerat kan vara en flaskhals för ett webbserverprogram. För att förhindra att detta inträffar så appliceras Veal och Foong (2007) modifieringar på Linux kerneln. Vilket bland annat ökar det maximala antalet öppnar filer, samtida SYN-förfrågningar, väntande anslutningar och väntande paket. Modifieringarna ökar även det minne som olika buffrar kan använda till exempel buffrar för att skicka och ta emot data. Även de förändringar som Weighttp’s skapare (Icy, 2012) föreslår appliceras på kerneln. Det berör främst hur återanvändning av sockets ska ske.

4.2.5 Optimering av webbserverprogram

Optimeringen av konfigurationsfilen görs vid 1000 samtida anslutningar eftersom tre av webbserverprogrammen inte klarade att svara på alla förfrågningar vid så många samtida anslutningar. Målet är förutom att öka prestandan även försöka få ner antalet missade förfrågningar till noll.

Vid optimering så ändras vissa parameterar medan andra tas bort eller läggs till. Det här sker genom att metodiskt testa vilka parameterar och parametervärden som presterar bäst. Parametrar som existerar på alla fyra webbserverprogrammen sätts till samma värde. Exempel på detta är hur många förfrågningar en klient kan göra över en ihållande förbindelse. Moduler som sällan används tas bort, exempel är visning av innehållet i kataloger. Även extra information som kan användas för felsökning tas bort vilket främst är loggning av förfrågningar och information som skickas med i HTTP-headern.

En viktig del är att se till att webbserverprogrammen använder alla processorkärnor. Alla har stöd för detta men dokumentationen för Lighttpd varnar för att en del moduler inte kommer att fungera korrekt (Lang, 2009). Inga av dessa moduler används under prestandatesterna men det är värt att notera. I Apache är det möjligt att använda ”.htaccess” filer för att göra ändringar i konfigurationen för en enskild katalog. Enligt Apaches dokumentation (The Apache Software Foundation, 2012) så försämrar detta prestandan och skall endast användas när konfigurationsfilerna inte finns att tillgå. Det hindrar dock inte systemadministratörer från att använda ”.htaccess” filer vilket förmodligen beror på att många guider använder dem. Dock i de här prestandatesterna så används inga ”.htaccess” filer.

I det dynamiska prestandatestet så ersätts PHP-CGI med det nyare PHP-FastCGI Process Manager (PHP-FPM). PHP-FPM är en server-daemon som tar hand om skapandet/reglerandet av FastCGI-processerna. Detta skiljer sig från PHP-CGI där webbserverprogrammet själv tar hand om skapandet/reglerandet. PHP-FPM optimeras ytterligare genom att använda Unix domain socket istället för TCP/IP sockets för kommunikation. Apache Prefork kommer att fortsätta att använda *mod_php*.

Cachelagring gör att ofta hämtade filer lagras i RAM vilket medför att det tar mindre tid att hämta dem. Alternativt lagras en fildeskriptor till filen i minnet. Varken de statiska eller

dynamiska prestandatesterna använder någon sorts cachelagring. Detta främst för att Weighttpd hämtar samma fil flera gånger vilket gör att filen alltid kommer att ligga lagrad i minnet. Med PHP så är det även möjligt att lagra den genererade filen precis som en statisk fil. Dock i en produktionsmiljö så finns det ingen möjlighet att lagra alla filer i minnet eller att ha alla PHP-filer förgenererade. Vidare så finns det en uppsjö av moduler och instickspprogram som används för cachelagring vilket medför att varje webbserverprogram måste genomgå ytterligare prestandatester.

4.3 Funktionalitet

Här listas de funktioner som är framtagna från Apaches webbsida.

4.3.1 Grundläggande autentisering

Grundläggande autentisering gör det möjligt för ett webbserverprogram att kräva att en webbklient (användare) ska identifiera sig själv (Franks, et al., 1999). Identifieringen sker med hjälp av användarnamn och lösenord som antingen finns sparat i webbklienten eller fylls i av användaren. Denna metod tillämpar ingen kryptering vilket gör att en angripare kan avlyssna kommunikationen mellan webbservern och webbklienten för att få reda på användarnamn och lösenord (Franks, et al., 1999).

4.3.2 Referat-autentisering

På grund av att användarnamn och lösenord skickas i klartext med grundläggande autentisering så utvecklades referat(digest)-autentisering (Franks, et al., 1999). Referat-autentisering krypterar inte användarnamn och lösenord utan använder istället MD5 för att beräkna en kontrollsumma av användarnamn, lösenord, metod, adress och ett värde (nonce) från webbserverprogrammet. Denna kontrollsumma används sedan för att identifiera användaren (Franks, et al., 1999).

4.3.3 HTTPS-stöd

HTTPS används för att skapa en krypterad förbindelse mellan en webbserver och en webbklient. HTTPS implementeras antingen med hjälp av Secure Socket Layer (SSL) eller efterträdaren Transport Layer Security (TLS) (Rescorla, 2000). Om bara ett av protokollen stöds så specificeras detta.

4.3.4 Komprimering

Komprimering av HTTP-trafik innebär att en algoritm används för att försöka minska den information som ska skickas utan att något går förlorad. De algoritmer som används är GNU zip (gzip), Lempel-Ziv-Welch (LZW) och z library (zlib) (Fielding, et al., 1999). En studie från Verve Studios (2010) visar på att alla webbläsare som har stöd för *zlib* också har stöd för *gzip* men att alla webbklienter som har stöd för *gzip* inte har stöd för *zlib*.

4.3.5 FastCGI och SCGI

CGI är ett enkelt gränssnitt för att exekvera externa skript och program på en webbserver genom webbserverprogrammet, och sedan skicka resultatet till webbklienten. Ett problem med CGI är att varje gång ett skript eller program exekveras så startas en ny process (Robinson & Coar, 2004). FastCGI och Simple CGI (SCGI) är två sätt att lösa detta problem.

FastCGI (Brown, 1996) löser detta problem genom att använda ihållande processer som inte avslutas efter att en förfrågan är utförd. SCGI (Schemenauer, 2008) liknar FastCGI men är utformat för att vara lätt att implementera.

4.3.6 Server Side Include

Server Side Include (SSI) är direktiv som kan placeras i webbsidor för att exekvera program eller hämta information. Det är lättare att använda än CGI men inte lika flexibelt (Gundavaram, 1996).

4.3.7 Virtuell hosting

Virtuell hosting är när ett webbserverprogram hanterar webbplatser för flera olika domännamn. Det kan antingen vara IP-baserat vilket betyder att varje domännamnen har olika IP-adresser eller namn-baserat vilket betyder att flera domännamn har samma IP-adress (The Apache Software Foundation, 2012a).

4.3.8 Stöd för IP version 6

IP version 6 (IPv6) är efterföljaren till den nuvarande IP version 4 (IPv4). IPv6 utvecklades främst för att antalet IPv4-adresser inte längre räcker till (Halsall, 2005).

4.3.9 Forward och Reverse proxy

En forward proxy är en server som finns mellan klienten och webbservern (The Apache Software Foundation, 2011c). Klienten skickar en förfrågan till en forward proxy som i sin tur skickar en egen förfrågan till den webbserver som klienten vill kommunicera med. Forward proxyn skickar sedan tillbaka svaret från webbservern till klienten. För webbservern så ser det ut som att det är forward proxyn som har skickat en förfrågan. Detta kräver dock att klienten är konfigurerad för att kommunicera med en forward proxy.

En reverse proxy är också en server som finns mellan klienten och webbservern. Dock så tar den inte emot förfrågningar till andra webbservrar utan fungerar utåt sett precis som en vanlig webbserver. Så för klienten ser det ut som att den enbart kommunicerar med reverse proxyn även fast reverse proxyn skickar förfrågningar vidare till andra webbservrar. Detta gör att klienten inte behöver vara konfigurerad på ett speciellt sätt (The Apache Software Foundation, 2011c).

5 Resultat

Här presenteras de resultat som har fåtts från prestandatesterna och vilka funktioner som stöds samt en analys av resultateten.

5.1 Statiskt prestandatest

Diagram 1 visar på att det inte skiljer sig så mycket i hur många förfrågningar de klarar av. Apache Worker är den som sticker ut och klarar ungefär 10 000 fler förfrågningar per sekund. Lighttpd är det enda webbserverprogrammet som lyckas svara på alla förfrågningar (se Diagram 2). Dock är det en försvinnande liten del av alla förfrågningar som inte besvaras av de andra. Exempel Nginx som har mest antal missade förfrågningar svarar inte på 0,0114 % av alla förfrågningar. Att Apache Prefork missar förfrågningar beror förmodligen på att maximalt antal processer som kan startas är 150 vilket medför att bara 150 samtida förfrågningar kan besvaras. Apache Worker har ett liknande problem och kan bara starta 16 processer med maximalt 25 trådar per process vilken betyder att 400 samtida förfrågningar kan besvaras. Att de sedan klarar av att svara på 500 samtida anslutningar beror förmodligen på att Apache kan ha 511 anslutningar som står i kö (backlog). Nginx har ett maxvärde på 1024 samtida anslutningar med en kö på 511 anslutningar men missar ändå flest förfrågningar. Lighttpd har också ett maxvärde på 1024 samtida anslutningar men har en kö på 1024 anslutningar. Den slutsats som kan dras från det här är att processorn inte hinner med att svara på alla förfrågningar så en längre kö är nödvändig för att inte förfrågningar ska missas. Det förklarar dock inte varför Nginx missar fler förfrågningar än båda konfigurationerna av Apache.

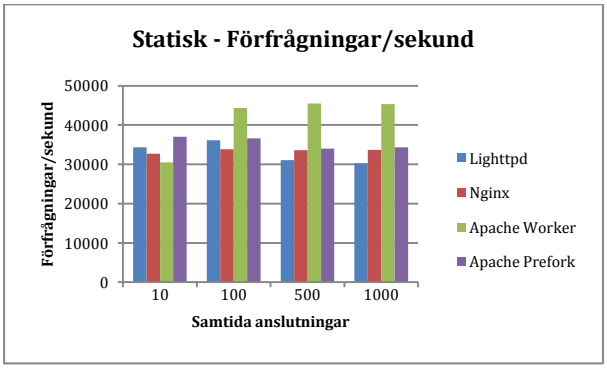


Diagram 1 Antal förfrågningar per sekund.

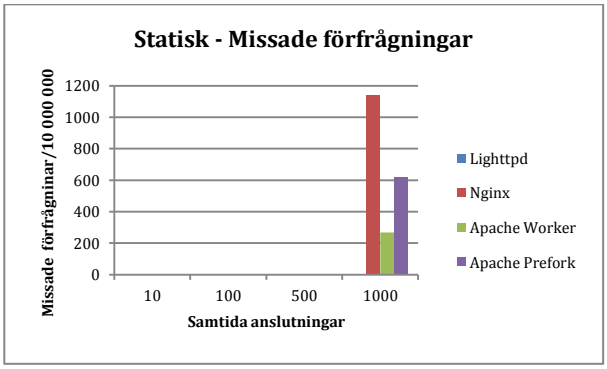


Diagram 2 Missade förfrågningar per 10 miljoner.

Diagram 3 visar på att Nginx och Lighttpd endast använder en fjärdedel av Apache Worker/Prefork. Detta beror på att den förvalda konfigurationen för Nginx och Lighttpd endast använder en processorkärna medan Apache Worker/Prefork använder fyra processorkärnor. Det förklarar även varför Nginx missar fler förfrågningar än Apache Worker/Prefork. Apache Prefork är som sagt tvungen att starta en ny process för varje ny anslutning så det är därför inte konstigt att den använder mest minne (se Diagram 4). Lighttpd marknadsför sig som ett lättviktigt webbserverprogram vilket tydlig kan ses.

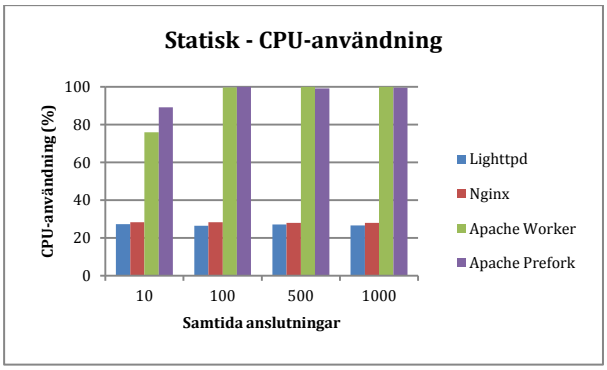


Diagram 3 CPU-användning i procent.

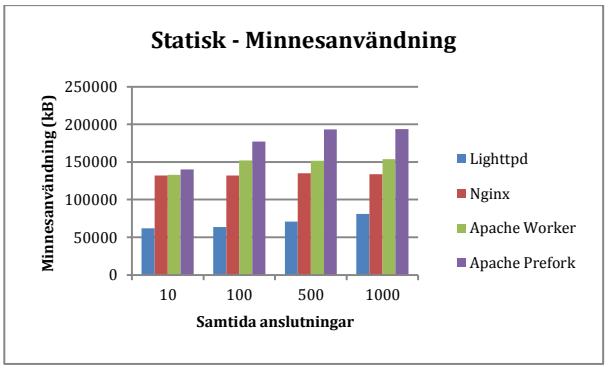


Diagram 4 Minnesanvändning i kilobyte.

5.2 Dynamiskt prestandatest

I det första dynamiska prestandatestet använder webbserverprogrammen antingen *mod_php* eller PHP-CGI. Diagram 5 visar tydligt hur överlägsen Apache Prefork med *mod_php* är. Det är dock underligt att de andra tre webbserverprogrammen visar på så olika resultat när de använder samma externa program, PHP-CGI. Vid 1000 samtida anslutningar så kollapsar Nginx helt och har nästan 25 gånger fler missade förfrågningar jämfört med tvåan, Apache Worker (se Diagram 6). Även i det här testet så svarar Lighttpd på alla förfrågningar även fast den använder ett externt program.

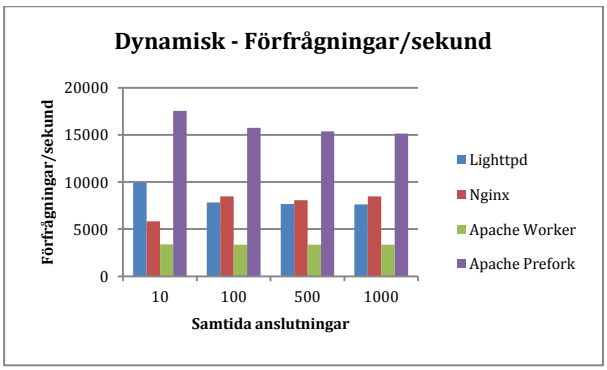


Diagram 5 Antal förfrågningar per sekund.

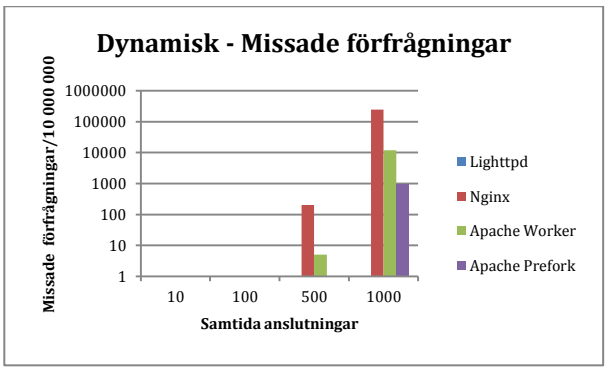


Diagram 6 Missade förfrågningar per 10 miljoner.

Precis som i det statiska prestandatestet så använder Nginx och Lighttpd en processorkärna. Detta hindrar dock inte PHP-CGI från att använda fler processorkärnor (se Diagram 7). Minnesanvändingen följer tidigare mönster förutom att Nginx använder betydligt mycket mer minne vid 500 samtida anslutningar (se Diagram 8). Att Nginx gått från att använda mer minne än Apache Prefork vid 500 samtida anslutningar till att använda mindre vid 1000 samtida kan verkar underligt men förklaringen finns i Diagram 6. Nginx hinner helt enkelt inte med att svara på alla förfrågningar vilket medför att den inte behöver allokera lika mycket minne.

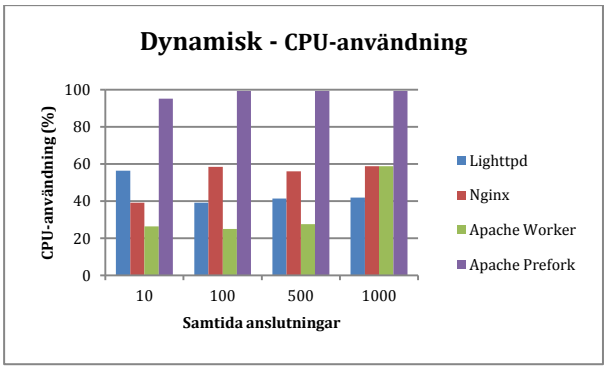


Diagram 7 CPU-användning i procent.

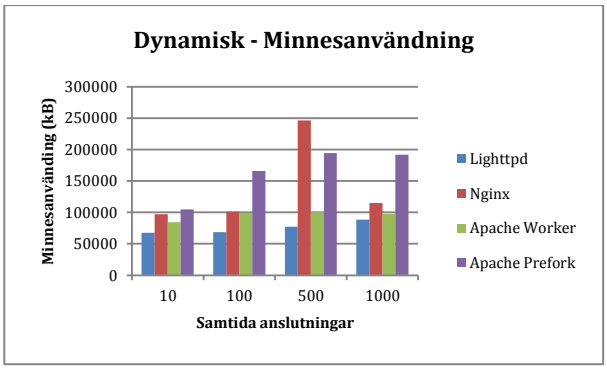


Diagram 8 Minnesanvändning i kilobyte.

5.3 Statiskt prestandatest med optimering

Vid optimering så presterar Nginx och Lighttpd mellan två till tre gånger bättre än Apache (se Diagram 9). En jämförelse mellan Diagram 1 och 9 visar tydligt på att Webbserverprogrammens konfigurationsfiler är olika mycket optimerade från början. Nginx och Lighttpd är de webbserverprogram som har ökat mest i prestanda. Detta beror till stor del på att de har gått från att använda en processorkärna till att använda fyra. Dessutom så klarar alla webbserverprogrammen att svara på samtliga förfrågningar.

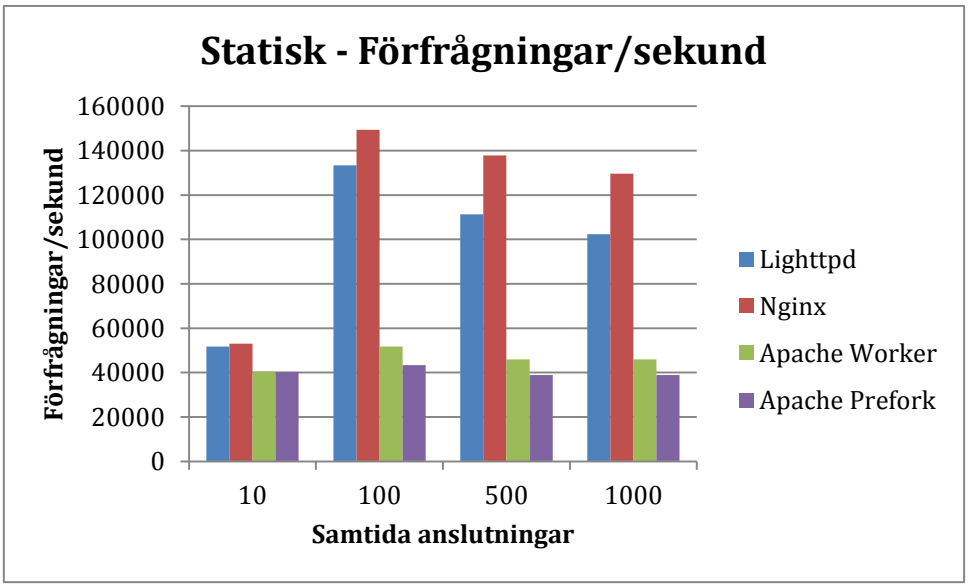


Diagram 9 Antal förfrågningar per sekund.

Vid optimering tas moduler som sällan används och extra information som används vid felsökning bort (se 4.2.5 Optimering av webbserverprogram) vilket medför att alla webbserverprogram tar upp mindre minne (se Diagram 10). Även Apache Prefork använder mindre minne per process dock eftersom Apache Prefork behöver starta fler processer för att hantera alla förfrågningar så tar den upp mer minne totalt. På grund av att alla webbserverprogram nu använder fyra processorkärnor så ligger CPU-användningen för alla webbserverprogram på 100 %.

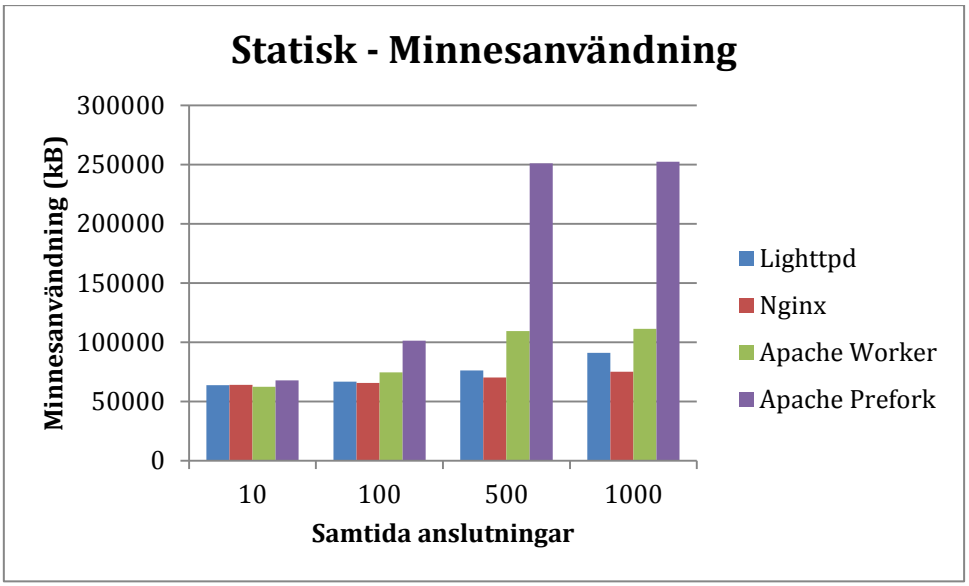


Diagram 10 Minnesanvändning i kilobyte.

5.4 Dynamiskt prestandatest med optimering

Vid optimering av webbserverprogrammen inför det dynamiska prestandatestet så har PHP-CGI ersatts med PHP-FPM (se 4.2.5 Optimering av webbserverprogram). Apache Prefork använder fortfarande *mod_php*. Med PHP-FPM så klarar Nginx av nästan lika många förfrågningar som Apache Prefork och Lighttpd är inte långt efter (se Diagram 11). Precis som i det statiska prestandatestet så lyckas alla webbserverprogrammen svara på samtliga förfrågningar.

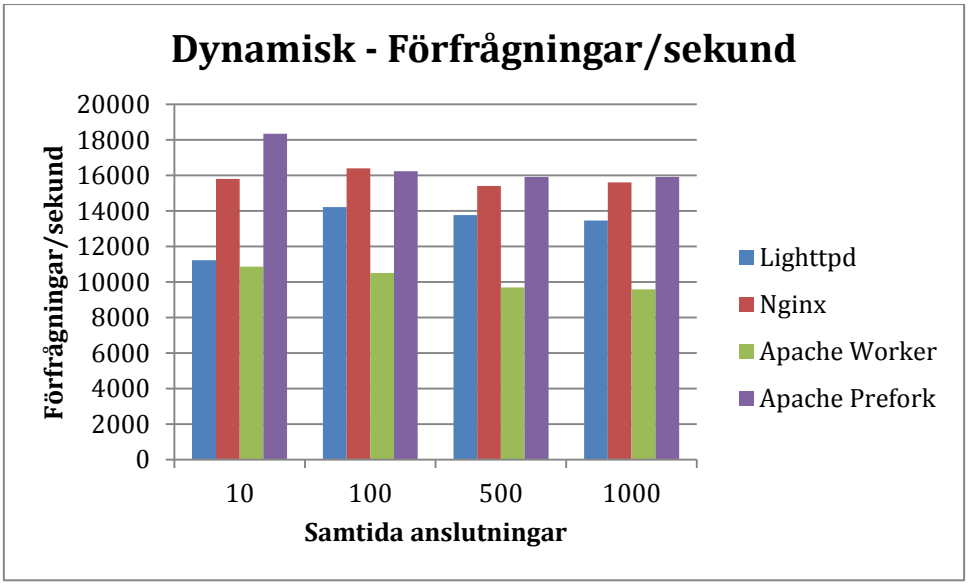
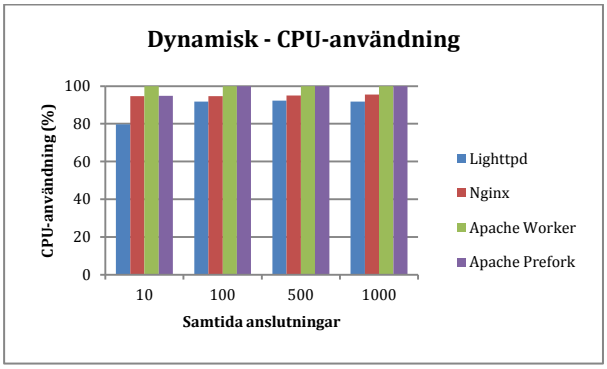


Diagram 11 Antal förfrågningar per sekund.

Diagram 12 visar på att Nginx och Lighttpd inte använder 100 % av processorn vilket skulle kunna bero på att PHP-FPM inte har fullt stöd för att köras på flera processorkärnor. Den lilla skillnaden i CPU-användningen mellan Nginx och Lighttpd kan dock inte förklara varför

Lighttpd presterar sämre. Båda använder samma implementation av FastCGI vilket ger indikationer på att något är annorlunda i Lighttpds kommunikation med PHP-FPM.



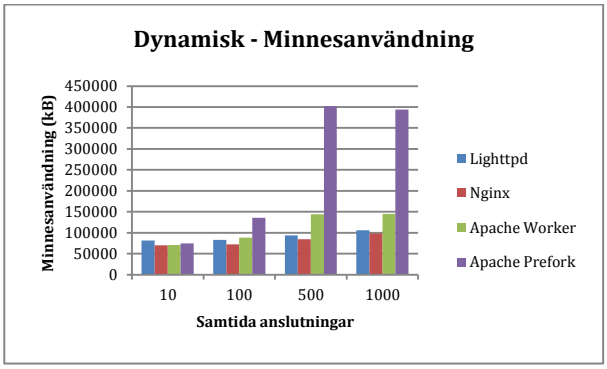


Diagram 12 CPU-användning i procent.

Diagram 13 Minnesanvändning i kilobyte.

Diagram 13 visar på att Apache Prefork har mellan tre till fyra gånger så hög minnesanvändning som de andra vilket beror på att det är mer krävande att hantera dynamiska filer än statiska. För de andra webbserverprogrammen har minnesanvändningen också ökat men det grundar sig i att PHP-FPM startar ett antal processer för att hantera PHP.

5.5 Funktionalitet

Tabellen nedan innehåller de funktioner som de olika webbserverprogrammen har stöd för.

Funktioner	Apache	Nginx	Lighttpd
Grundläggande Autentisering	Ja	Ja	Ja
Referat-autentisering	Ja	Nej	Ja
HTTPS-stöd	Ja	Ja	Ja
Komprimering	gzip	gzip	gzip och zlib
FastCGI och SCGI	Båda	Båda	Båda
Server Side Includes	Ja	Ja	Ja
Virtuell hostning	Ja	Ja	Ja
IP version 6	Ja	Ja	Ja
Forward och Reverse Proxy	Båda	Reverse	Båda

Alla tre webbserverprogrammen har funnits i flera år det är därför inte så konstigt att de har stöd för de flesta funktionerna. Det finns dock några undantag. Nginx saknar stöd för referat-autentisering i den nuvarande versionen. Dock finns det en färdigutvecklad modul för referat-autentisering som bara behöver mer testning (Nginx Inc, 2011). Lighttpd är det enda webbserverprogrammet som implementerar *zlib* komprimering. Dock är detta ingen direkt fördel eftersom de webbklienter som har stöd för *zlib* också har stöd för *gzip* (se 4.3.4 Komprimering). Nginx saknar även stöd att konfigureras som en Forward Proxy.

5.6 Analys

Det skiljer väldigt mycket i hur konfigurationsfilerna är optimerade vid en default installation. När förvalda värden används så hanterar Lighttpd och Nginx statiska filer lika bra som

Apache Prefork. Vid optimering av konfigurationsfilerna så hanterar Nginx över tre gånger så många förfrågningar som Apache Prefork. Lighttpd är inte långt efter Nginx och anledningen att den inte presterar lika bra beror förmodligen på att Lighttpd inte har fullt stöd för att använda flera processorkärnor (se 4.2.5 Optimering av webbserverprogram).

Utan optimering så hanterar Apache Prefork dynamiska förfrågningar två gånger så snabbt som alla andra. Dessutom så missar Nginx en massa förfrågningar när antalet samtida anslutningar överstiger 500. När PHP-FPM istället används så presterar Nginx och Apache Prefork lika bra medan Lighttpd inte riktigt uppnår samma prestanda. Den största skillnaden är att PHP-FPM själv tar hand om skapande/reglerande av PHP-processer vilket medför att webbserverprogrammet inte behöver ha kunskap om hur PHP fungerar.

Prestandatesterna visar också på att Apache Prefork tar upp mellan tre till fyra gånger mer minne än Nginx och Lighttpd. Det här beror på att Apache Prefork behöver starta en process för varje samtida anslutning som ska hanteras medan Nginx och Lighttpd endast startar en process per processorkärna. Skillnaden i minnesanvändning kan också vara en indikation på att när större filer används så kommer mer RAM att behövas. Dock behöver fler prestandatester utföras med olika filstorlekar innan någon slutsats kan dras. Det som även syns i prestandatesterna är att webbserverprogrammen tar hela processorn i anspråk för att kunna svara på förfrågningarna så snabbt som möjligt vilket medför att mätning av CPU-användning inte tillför något.

Webbserverprogrammen i det här prestandatestet använder tre olika arkitekturer. Apache är antingen enbart processbaserad eller processbaserad i kombination med trådning. Nginx och Lighttpd är händelsebaserade men kan använda flera processer för att distribuera förfrågningarna mellan flera processorkärnor. Vad som tydligt kan ses i de optimerade statiska prestandatesterna är att händelsebaserad arkitektur presterar mycket bättre än de andra två. Det här beror främst på att i en händelsebaserad arkitektur så behöver inte webbserverprogrammet ha en process eller tråd för varje samtida anslutning som behöver synkroniseras (Pariag et al, 2007). Värt att nämna är att Apache håller på att utveckla en Multi-Processing Modul som är händelsebaserad.

Apache Prefork med *mod_php* presterar lite bättre än Nginx med PHP-FPM men det finns en stor nackdel med *mod_php* och det är att den är integrerad i Apache. Integrationen gör att om *mod_php* skulle sluta fungera så måste alla Apache processer startas om. En annan svaghet med integrationen är att det inte är möjligt att skicka förfrågningar vidare till en annan webbserver.

PHP-FPM har inte dessa svagheter för att det är ett självständigt program. På grund av att ett FastCGI-gränssnitt finns mellan webbserverprogrammet och PHP-FPM så är det heller inte svårt att dirigera om förfrågningar till andra webbservrar. Det här är dock främst aktuellt för webbplatsers med mycket trafik och där större delen är dynamiska förfrågningar.

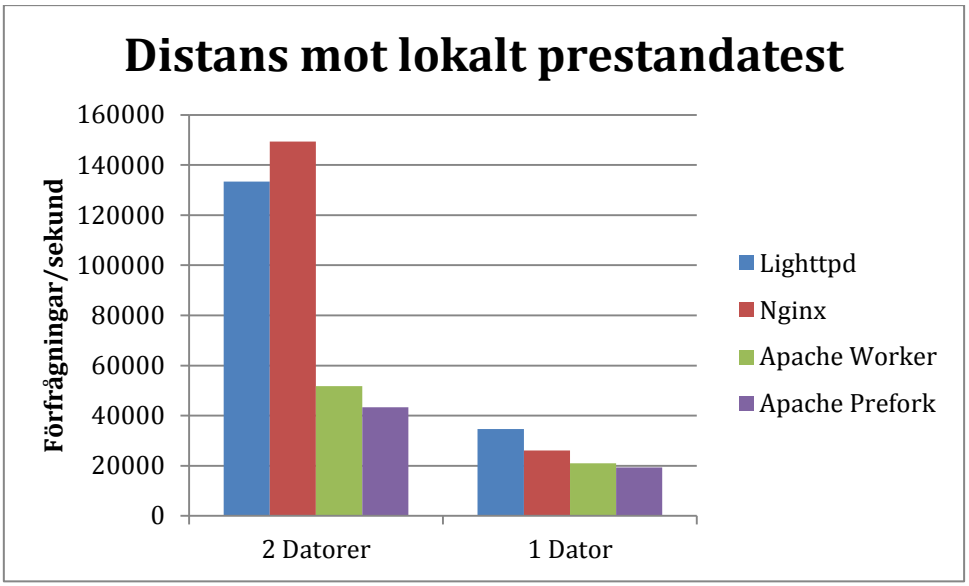


Diagram 14 Skillnaden mellan att använda 2 datorer och 1 dator för att utföra prestandatesterna.

De prestandatester som utförts på diverse bloggar har installerat webbserverprogrammet och testprogrammet på samma dator vilket medför att nätverkskortet inte längre är någon flaskhals. Ett stort problem med detta är att webbserverprogram och testprogrammet måste dela på processorn så ett webbserverprogram som inte har så hög CPU-användning har en klar fördel. Diagram 14 visar skillnaden mellan prestandatester som utförts lokalt och de som utförts mellan en server och en klient. Lokalt (en dator) så presterar Lighttpd bättre än Nginx medan när testprogrammet installeras på klienten (två datorer) så presterar Nginx bättre. Därför är det av vikt att testprogrammet körs på en annan maskin för att inte påverka prestandatestet.

Webbserverprogrammen har stöd för nästan alla funktioner vilket inte är så konstigt eftersom de har existerat i nästan 10 år. Nginx saknar dock stöd för referat-autentisering vilket nödvändigtvis inte är något problem eftersom det är möjligt att kombinera HTTPS med grundläggande autentisering för att få samma säkerhet. En modul håller även på att utvecklas för referat-autentisering. Det är inte heller möjligt att konfigurera Nginx som en Forward proxy vilket kan ses som en nackdel om ett företag enbart vill använda ett webbserverprogram för alla webbservrar.

6 Slutsats

Det här examensarbetet har undersökt om det finns några öppen källkods-alternativ till Apache som har bättre prestanda och bibehållen funktionalitet. De webbserverprogram som undersöktes var Lighttpd och Nginx som båda skapats för att klara av 10 000 samtida förbindelser.

De statiska prestandatesterna visade på att när medföljande konfigurationsfiler användes så uppnådde både Nginx och Lighttpd samma prestanda som Apache Prefork. Vid optimering så presterade Nginx och Lighttpd två till tre gånger bättre än Apache Prefork. Ökningen berodde förmodligen till stor del på att Nginx och Lighttpd endast använde en processorkärna i de första prestandatesterna medan Apache använde fyra.

När medföljande konfigurationsfiler användes i de dynamiska prestandatesterna så klarade Apache Prefork av att hantera dubbelt så många förfrågningar som Nginx och Lighttpd. Vid optimering så hade Nginx och Apache Prefork jämförbar prestanda medan Lighttpd inte riktigt uppnådde samma prestanda. Prestandatesterna visade också på att Apache Prefork använde tre till fyra gånger mer minne än de övriga webbserverprogrammen. Det här kan vara en indikation på att vid användning av större filer så kommer mycket mer RAM att behövas. Dock behöver vidare testning göras för att dra några slutsatser om detta.

Baserat på prestandatesterna så presterade Nginx och Lighttpd i det stora hela bättre än Apache. Det här syns främst i de statiska prestandatesterna där både Nginx och Lighttpd presterar mycket bättre än Apache Prefork. I de dynamiska prestandatesterna så finns det ingen klar vinnare. Dock behöver Nginx och Lighttpd använda PHP-FPM för att uppnå samma prestanda. Detta gäller särskilt Nginx som annars får problem vid över 500 samtida anslutningar.

Prestandatesterna visade också på att webbserverprogrammen konfigurationsfiler var väldigt olika optimerade efter installationen. Nginx och Lighttpd var de som förbättrades mest vid optimering främst för att de var konfigurerade för att köras på endast en processorkärna. Nginx var den enda som saknade viss funktionalitet dock var det inga kritiska funktioner som saknades.

6.1 Diskussion

Att undersöka om det fanns några alternativ till webbserverprogrammet Apache var mer tidskrävande än väntat. Främst för att medföljande konfigurationsfiler för webbserverprogrammen var väldigt olika optimerade. T.ex. så använde Nginx och Lighttpd bara en processorkärna medan Apache använde fyra. Det här medförde att prestandatesterna till stor del utvärderade konfigurationsfilerna istället för att utvärdera webbserverprogrammen. Därför genomfördes ett andra prestandatest under mer jämlika förhållanden där t.ex. alla webbserverprogrammen använde fyra processorkärnor.

Avsaknaden av vetenskapliga artiklar inom ämnet gjorde att experiment var ett passande val för att undersöka prestandaskillnader. På grund av tidsbrist så fanns det dock inte tid att utföra

flera kombinationer av tester, vilket medförde att prestandatesterna inte visar hur mycket skillnad det är mellan PHP-CGI och PHP-FPM, och hur mycket som beror på optimeringen. Testerna visar inte heller på hur mycket prestanda som Nginx och Lighttpd förlorar på grund av att de bara använde en processorkärna i första prestandatestet. Prestandatesterna tittar inte heller på andra filstorlekar vilket kan påverka resultatet. Dock bör skillnaden (förfrågningar/sekund) mellan webbserverprogrammen minska desto större filer som används eftersom det belastar operativsystem, hårddisk och nätverkskort mer. Dock är det oklart hur minnesanvändningen kommer påverkas när större filer används.

Funktionaliteten undersöktes genom att titta på vilka funktioner som fanns listade på Apaches hemsida och sedan undersöka vilka av dessa som Lighttpd och Nginx hade stöd för. Ett problem med den här metoden är att bara för att Apache har listat något som funktionalitet så innebär det inte att det används i produktionsmiljöer. Samma sak gäller för funktionalitet som inte är listad för att det är självklart att det finns i ett webbserverprogram. Det här kan medföra att viss funktionalitet antingen saknas eller är onödig.

Att varje prestandatest enbart har körts tre gånger beror på att inom ramen för examensarbetet så fanns det inte tid för att köra flera tester. Iterationen genom alla prestandatester för en del tog nästan 10½ timme vilket betyder att om varje prestandatests skulle köras tio gånger så skulle det ta över 200 timmar. Variationen är dock relativ liten mellan de olika gångerna ett test körs. Största skillnad från det beräknade medelvärdet är 0,53 % och största skillnaden mellan två värden är 0,84 %.

Testerna som gjordes av Pariag et al. (2007) visade på att en händelsebaserad arkitektur presterade 18 % bättre än en arkitektur som använder trådning. I de prestandatester som genomfördes här så presterade de händelsebaserade webbserverprogrammen mellan 122 – 182 % bättre än de som använde trådning. Det kan bero på att händelsebaserade webbserverprogram har utvecklats mycket mer än trådbaserade. Dock går det inte säga med säkerhet eftersom Pariag et al. (2007) modifierade de olika webbserverprogrammen så att det enda som skiljde var arkitekturen.

Prestandatesterna som utförts på diverse bloggar använder oftast en dator vilket skapar flera problem. Först så blir prestandan (förfrågningar/sekund) lägre, men det ändrar även resultatet. Ett webbserverprogram som klarar sig på mindre processorkraft presterar bättre än ett som måste ha en viss processorkraft för att fungera optimalt (se 5.6 Analys). I de tester som Arnold (2010) har gjort på sin blogg så presterar Lighttpd bättre än Nginx vilket inte stämmer överens med prestandatesterna i den här studien. Förklaringen kan ses i Diagram 14 (5.6 Analys) där Lighttpd presterar bättre än Nginx när testprogrammet och webbserverprogrammet ligger på samma dator. Dock när testprogrammet och webbserverprogrammet ligger på olika datorer så presterar Nginx bättre.

6.2 Samhälleliga, vetenskapliga och etiska aspekter

Studien visade på att både Nginx och Lighttpd är bra alternativ till Apache. Dessutom vid optimering av konfigurationsfilerna så presterade Nginx och Lighttpd i det stora hela bättre än Apache Prefork. Företag som är på väg att införskaffa ett webbserverprogram kan således använda den här studien för att underlätta valet.

Som tidigare nämnts är prestandatesterna uppdelade i två delar, där första delen använder medföljande konfigurationsfiler och andra delen använder optimerade konfigurationsfiler. Dessa två delar kan användas av systemadministratörer för att påvisa vikten av att optimera ett webbserverprogram.

Arbetet visade även på att CPU-användning är en parameter som inte är värd att undersöka i ett prestandatest eftersom webbserverprogrammen försöker använda så mycket av processorn som möjligt. Det är dock viktigt att CPU-användingen övervakas för att upptäcka eventuella flaskhalsar. Vidare så undersöktes ingående hur minnesanvändningen bör avläsas och vilka problem som finns med att titta på enskilda programs minnesanvändning. Slutligen så visade studien på att testprogrammet och webbserverprogrammet bör köras på separata datorer för att de inte ska påverka varandra. Det här gör att arbetet är en bra grund till framtida prestandatester för webbserverprogram. Det finns inga relevanta etiska aspekter att ta upp i den här studien eftersom den inte behandlar människor i någon form.

6.3 Framtida arbete

Som nämnts tidigare så finns väldigt många webbserverprogram på marknaden och det skulle vara intressant att se hur de presterar i jämförelse webbserverprogrammen i den här studien. Det skulle även vara intressant att se om det är någon skillnad mellan webbserverprogram som är proprietära och öppen källkod.

I de dynamiska prestandatesterna så undersöktes bara PHP dock finns det flera andra serverskriptspråk som används i produktionsmiljöer. ASP.NET är det som är störst förutom PHP men det finns även andra som Java, ColdFusion och Perl. Det skulle därför vara intressant att titta på hur dessa serverskriptspråk presterar i jämförelse med PHP.

En reverse proxy är när ett webbserverprogram konfigureras för att skicka vidare en del förfrågningar, till exempel svarar på alla statiska förfrågningar och skickar vidare alla dynamiska förfrågningar. Alla tre webbserverprogrammen har stöd för att konfigureras som en reverse proxy men de prestandatester som utförts i den här studien tittar inte på den aspekten. Det skulle därför vara intressant att se om det är någon prestandaskillnad mellan webbserverprogrammen när de är konfigurerade som en reverse proxy.

Referenser

Abbate, J. 1999. *Inventing the Internet.* Cambridge: MIT Press.

Arnold, F. 2010. *Server Benchmark: Apache, Nginx, Cherokee, Lighttpd.* [ONLINE] http://arnisoft.com/239/server-benchmark-apache-nginx-cherokee/ [Hämtad 20120601]

Atkinson, R. 2008. *Apache, Nginx and Lighttpd.* [ONLINE] Tillgänglig: http://jetfar.com/apache-nginx-lighttpd-future-compared/ [Hämtad 20120321]

Berners-Lee, T. 1989. *The original proposal of the WWW, HTMLized.* [ONLINE] Tillgänglig: http://www.w3.org/History/1989/proposal.html [Hämtad: 20120211]

Brown, M. 1996. *FastCGI: A High-Performance Gateway Interface.* Fifth International World Wide Web Conference.

Coarfa, C., Druschel, P., & Wallach, D. 2006. *Performance analysis of TLS Web servers*. ACM Transactions on Computer Systems (TOCS), v.24 n.1, p.39-69, February 2006

Rescorla, E. 2000. *HTTP over TLS.* [ONLINE] Tillgänglig: http://www.rfc-editor.org/rfc/rfc2818.txt [Hämtad: 20120404]

Dmitrij. 2011. *Nginx vs Cherokee vs Apache vs Lighttpd.* [ONLINE] Tillgänglig: http://www.whisperdale.net/11-nginx-vs-cherokee-vs-apache-vs-lighttpd.html [Hämtad 20120321]

E-Soft Inc. 2012. *Web Server Survey – SecuritySpace.* [ONLINE] Tillgänglig: http://www.securityspace.com/s_survey/data/201201/index.html [Hämtad 20120211]

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P. & Berners-Lee, T 1999. *Hypertext Transfer Protocol – HTTP/1.1.* [ONLINE] Tillgänglig: http://tools.ietf.org/html/rfc2616 [Hämtad: 20120404]

Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. & Stewart, L. 1999. *HTTP Authentication: Basic and Digest Access Authentication.* [ONLINE] Tillgänglig: http://www.rfc-editor.org/rfc/rfc2617.txt [Hämtad: 20120404]

GNU. 2012. *Dd invocation – GNU Coreutils.* [ONLINE] Tillgänglig: http://www.gnu.org/software/coreutils/manual/html_node/dd-invocation.html [Hämtad 20120403]

Gillies, J. & Cailliau R. 2000. *How the web was born.* New york: Oxford University Press.

Greatstatistic.com 2012a. *Web Server Online Usage Statistical Survey Feb 2012.* [ONLINE] Tillgänglig: http://www.greatstatistics.com/Feb-2012-stats.php [Hämtad 20120211]

Greatstatistic.com 2012b. *World Web Server Statistics.* [ONLINE] Tillgänglig: http://www.greatstatistics.com/serverstats.php [Hämtad 20120211]

Gundavaram, S. 1996. *CGI Programming on the World Wide Web.* Cambridge: O'Reilly Media

Halsall, F. 2005. *Computer Networking and the Internet.* 5th Edition. Harlow: Addison-Wesley.

IBM. 2012. *IBM - IBM HTTP Server.* [ONLINE] Tillgänglig: http://www-01.ibm.com/software/webservers/httpservers/ [Hämtad 20120314]

Icy. 2012. *Start - Weighttp - lighty labs.* [ONLINE] Tillgänglig: http://redmine.lighttpd.net/projects/weighttp/wiki [Hämtad 20120420]

Internet Observatory. 2012. *Internet Observatory::Europe.* [ONLINE] Tillgänglig: http://www.internetobservatory.net/europe/thelastmonth/pie/ [Hämtad 20120211]

Kneschke, J. 2012. *lighttpd fly light.* [ONLINE] Tillgänglig: http://www.lighttpd.net/ [Hämtad 20120213]

Kneschke, J. 2007. *lighttpd fly light.* [ONLINE] Tillgänglig: http://www.lighttpd.net/story [Hämtad 20120213]

Lang, J-P. 2009. *Docs:MultiProcessor – Lighttpd – lighty labs.* [ONLINE] Tillgänglig: http://redmine.lighttpd.net/projects/lighttpd/wiki/Docs:MultiProcessor [Hämtad 20120424].

LiteSpeed. 2012. *Home.* [ONLINE] Tillgänglig: http://litespeedtech.com/ [Hämtad 20120314]

LiteSpeed. 2012a. *LiteSpeed Web Server Edition.* [ONLINE] Tillgänglig: http://litespeedtech.com/litespeed-web-server-editions.html [Hämtad 20120314]

Oversee.net. 2012. *Oversee.net.* [ONLINE] Tillgänglig: http://oversee.net/ [Hämtad 20120314]

MarketDirection. 2010. *Företagets hemsida – hetaste marknadsföringskanalen.* [ONLINE] Tillgänglig: http://riktning.wordpress.com/2010/01/20/hetaste-marknadsforingskanalen/ [Hämtad 20120321]

Metz, C. 2010. *Google mystery server runs 13% of active websites.* [ONLINE] Tillgänglig: http://www.theregister.co.uk/2010/01/29/google_web_server/ [Hämtad 20120212]

Microsoft. 2012. *The Official Microsoft IIS Site.* [ONLINE] Tillgänglig: http://www.iis.net/ [Hämtad 20120126]

Microsoft. 2005. *IIS Integration.* [ONLINE] Tillgänglig: http://technet.microsoft.com/en-us/library/bb123818%28v=exchg.65%29.aspx [Hämtad 20120313]

Mockus, A., Fielding, R. & Herbsleb, J. 2000. *A case study of open source software development: the Apache server.* Proceedings of the 22nd international conference on Software engineering. ACM, p.263-272.

Netcraft LTD. 2012. *January 2012 Web Server Survey*. [ONLINE] Tillgänglig: http://news.netcraft.com/archives/2012/01/03/january-2012-web-server-survey.html [Hämtad 20120127]

Nginx Inc. 2011. HttpAuthDigestModule. [ONLINE] Tillgänglig: http://wiki.nginx.org/HttpAuthDigestModule [Hämtad 20120423]

Nginx Inc. 2012. *NGINX, Inc.* [ONLINE] Tillgänglig: http://www.nginx.com/ [Hämtad 20120213]

Oretga, A. 2012. *Cherokee Web Server.* [ONLINE] Tillgänglig: http://www.cherokee-project.com/ [Hämtad 20120314]

Pariag, D., Brecht, T., Harji, A., Buhr, P., Shukla, A. & Cheriton, D. 2007. *Comparing the Performance of Web Server Architectures.* ACM SIGOPS/EuroSys European Conference on Computer Systems.

Robinson, D. & Coar, K. 2004. *The Common Gateway Interface (CGI) Version 1.1.* [ONLINE] Tillgänglig: http://tools.ietf.org/rfc/rfc3875.txt [Hämtad 20120405]

Riverbed Technology. 2011. *End-of-Availability and End-of-Support Notice.* [ONLINE] Tillgänglig: http://www.riverbed.com/us/assets/media/documents/company_information/EOAAnnouncement_ZWSandZGLB.pdf [Hämtad 20120314]

Sandvine. 2011. *Global Internet Phenomena Report: Fall 2011.* [ONLINE] Tillgänglig: http://www.sandvine.com/news/global_broadband_trends.asp [Hämtad 20120320]

Schemenauer, N. 2008. *SCGI: A Simple Common Gateway Interface alternative.* [ONLINE] Tillgänglig: http://python.ca/scgi/protocol.txt [Hämtad 20120405]

StatCounter. 2012. *Top 5 Browsers on Feb 2012 | StatCounter Global Stats.* [ONLINE] Tillgänglig: http://gs.statcounter.com/#browser-ww-monthly-201202-201202-bar [Hämtad 20120312]

Svenska dagbladet. 2008. *Allt vanligare med marknadsföring via bloggar*. [ONLINE] Tillgänglig: http://www.svd.se/nyheter/inrikes/allt-vanligare-med-marknadsforing-via-bloggar_1175929.svd [Hämtad 20120321]

The Apache Software Foundation. 2010. *Apache Traffic Server.* [ONLINE] Tillgänglig: http://trafficserver.apache.org/ [Hämtad 20120314]

The Apache Software Foundation. 2011. *The Apache HTTP Server Projekt.* [ONLINE] Tillgänglig: http://httpd.apache.org/ [Hämtad 20120126]

The Apache Software Foundation. 2011a. *About the Apache HTTP Server Projekt.* [ONLINE] Tillgänglig: http://httpd.apache.org/ABOUT_APACHE.html [Hämtad 20120216]

The Apache Software Foundation. 2011b. *ab – Apache HTTP server benchmarking tool.* [ONLINE] Tillgänglig: http://httpd.apache.org/docs/2.0/programs/ab.html [Hämtad: 20120401]

The Apache Software Foundation. 2011c. *mod_proxy – Apache HTTP Server.* [ONLINE] Tillgänglig: http://httpd.apache.org/docs/2.2/mod/mod_proxy.html [Hämtad: 20120424]

The Apache Software Foundation. 2012. *Apache Tutorial: .htaccess files.* [ONLINE] Tillgänglig: http://httpd.apache.org/docs/current/howto/htaccess.html [Hämtad 20120424]

The Apache Software Foundation. 2012a. *Apache Virtual Host documentation.* [ONLINE] Tillgänglig: http://httpd.apache.org/docs/2.2/vhosts/ [Hämtad 20120405]

The Apache Software Foundation. 2012b. *Apache Performance tuning.* [ONLINE] Tillgänglig: http://httpd.apache.org/docs/2.2/misc/perf-tuning.html [Hämtad 20120531]

The Eclipse Foundation. 2012. *Jetty.* [ONLINE] Tillgänglig: http://eclipse.org/jetty/ [Hämtad 20120314]

The Washington Post. 2009. *Yahoo Open Source Traffic Server.* [ONLINE] Tillgänglig: http://www.washingtonpost.com/wp-dyn/content/article/2009/11/02/AR2009110203202.html [Hämtad 20120216]

Veal, B. & Foong A. 2007. *Performance scalability of a multi-core webserver.* Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems, December 03-04, 2007, Orlando, Florida, USA.

Verve Studios. 2010. *Compression Tests: Results.* [ONLINE] Tillgänglig: http://www.vervestudios.co/projects/compression-tests/results [Hämtad 20120422]

W3Techs. 2012. *Usage Statistics and Market Share of Web Servers for Websites, February 2012.* [ONLINE] Tillgänglig: http://w3techs.com/technologies/overview/web_server/all [Hämtad 20120211]

W3Techs. 2012a. *Usage Statistics and Market Share of Unix for Websites, March 2012.* [ONLINE] Tillgänglig: http://w3techs.com/technologies/details/os-unix/all/all [Hämtad 20120320]

W3Techs. 2012b. *Usage Statistics and Market Share of Server-side Programming Languages for Websites, April 2012.* [ONLINE] Tillgänglig: http://w3techs.com/technologies/overview/programming_language/all [Hämtad: 20120404]

Williams, J. 2008. *Apache vs Nginx: Web Server Performance Deathmatch.* [ONLINE] Tillgänglig: http://joeandmotorboat.com/2008/02/28/apache-vs-nginx-web-server-performance-deathmatch/ [Hämtad 20120319]

Appendix A - Sammanställning av webbserverprogram

	Greatstatistics (1)	Securityspaces (2)	W3techs (3)	Netcraft (4)
Apache	68.53%	68.81%	66.3%	64.91%
Microsoft IIS	15.63%	15.14%	18.4%	14.46%
Nginx	6.69%	-	10.4%	9.63%
LiteSpeed	0.87%	-	1.4%	-
Oversee Turing	1.16%	-	-	-
GWS	0.01%	-	1.0%	3.25%
Lighttpd	-	-	0.6%	-
IBM Server	0.24%	-	0.4%	-
YTS	0.25%	-	0.3%	-
Jetty	0.11%	-	0.1%	-
Zeus	0.12%	0.17%	0.1%	-
Cherokee	0.02%	-	-	-

1. Greatstatistics.com (2012b)
2. E-Soft Inc (2012)
3. W3Techs (2012)
4. Netcraft LTD (2012)