Webbapplikationsuppdatering med noll stilleståndstid

William Lindholm

Kandidatavhandling

Datavetenskap vid Åbo Akademi

Handledare: Dragos Truscan

2023

Referat

Denna kandidatuppsats fokuserar på att utforska metoder för att uppdatera lagren i en 3-lagers webbapplikationsuppsättning utan att orsaka stilleståndd. Avhandlingen undersöker utmaningarna med att uppdatera varje lager av arkitekturen och ger praktiska strategier för att uppdatera lagren samtidigt som störningar för användarna minimeras. Målet med arbetet är att ge vägledning för webbutvecklare och systemadministratörer om hur man säkert och effektivt uppdaterar sina webbapplikationer.

Innehållsförteckning

Introduktion

Webbapplikationer är en typ av programvara som nås och oftast används via en webbläsare över internet. Webbapplikationer är i grund och botten bara ett fildelningssystem över internetet dock är dagens webbapplikationer oftast mycket mer avancerade. En webbapplikation i dagens läge består vanligtvis av tre huvudkomponenter: front-end, back-end och databasen. Tillsammans arbetar front-end, back-end och databas för att skapa en funktionell och interaktiv webbapplikation.

Fast det är omöjligt att sätta ett exakt pris på hur mycket stilleståndstid kostar i medeltal finns det vissa studier som ger ett medeltal. A Lerner skriver" Based on industry surveys, the number we typically cite is $5,600 p/minute", han citerar också en Avaya study som sätter priset mellan "$140K - $540K p/hour".

Webbapplikationer kan användas för en mängd olika ändamål, inklusive e-handel, sociala nätverk, projektledning och mer. Men för vilket än ändamålapplikationen används för är det grundläggande att hålla webbapplikationer uppdaterade för att säkerställa deras säkerhet, funktionalitet och prestanda. När nya hot och sårbarheter upptäcks konstant, uppdateringar säker gör att data hålls säkert. Uppdateringar förbättrar även funktionaliteten hos webbapplikationer, åtgärdar buggar och lägger till nya funktioner som förbättrar användarupplevelsen. Användare förväntar sig att webbapplikationer ska vara igång hela tiden, och eventuella stilleståndstid- eller prestandaproblem kan resultera i förlorade intäkter, minskad produktivitet och skada på företagets rykte. Därför är det avgörande att underhålla och uppdatera webbapplikationer regelbundet för att möta användarnas förväntningar och säkerställa deras fortsatta framgång.

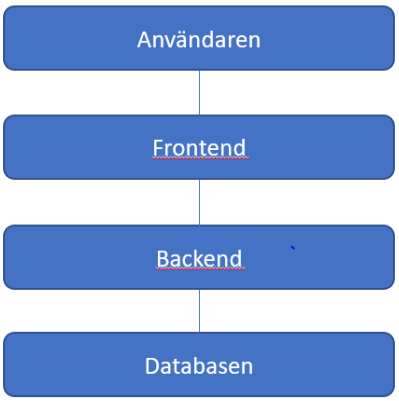
Stilleståndsfria uppdateringar är avgörande för webbapplikationer för att säkerställa kontinuerlig tillgänglighet och oavbruten användarupplevelse. När man uppdaterar en webbapplikation kan det leda till avbrott i tjänsten vilket i sig leder till förlust av intäkter,

användarfrustration och potentiell skada på applikationens rykte. Inom e-handelssektorn, där onlinetransaktioner genererar intäkter, kan driftstopp betydande ekonomiska konsekvenser. Till och med några minuters driftstopp under en period med hög trafik, till exempel en rea eller semesterperiod, kan resultera i förlorade intäkter och missnöjda kunder. Exempel kan vara om en e-handelswebbplats går ner i flera timmar under Black Friday, kan kunder välja att handla någon annanstans, vilket resulterar i en betydande försäljningsförlust.

Webbapplikationsstrukturen med 3 lager är en populär arkitektur för att bygga skalbara och underhållbara webbapplikationer. Strukturen består av ett presentationslager (frontend), ett affärslogiklager (backend) och ett datalagringslager (databasen). Presentationsskiktet hanterar användarinteraktion och visar information för användaren, medan affärslogikskiktet innehåller applikationens logik och bearbetar förfrågningar från presentationslagret. Datalagringslagret ansvarar för att lagra och hämta data från en databas eller annan datakälla. Detta skiktade tillvägagångssätt möjliggör separation av problem och enklare underhåll och utveckling av webbapplikationer.

Syftet med detta arbete är att undersöka metoder för att uppdatera lagren (i en 3-lager webbapplikation) utan att orsaka stilleståndd. 3-lager arkitekturen är ett populärt tillvägagångssätt för att bygga skalbara och underhållbara webbapplikationer, bestående av ett presentationslager, ett affärslogiklager och ett datalagringslager. Men att uppdatera dessa lager kan vara utmanande, eftersom alla stilleståndds tid kan påverka användarupplevelsen och affärsverksamheten. I arbete kommer jag undersöka olika strategier för att uppdatera varje lager i arkitekturen utan att orsaka stilleståndd, såsom rullande- och blågröna uppdateringar.

Bakgrund



Frontenden för en webbapplikation är den del som användaren interagera med direkt. Det är vad användaren ser igenom och interagerar med i sin webbläsare. Till frontenden räknas därför det grafiska användargränssnittet (GUI), layouten, formulären, knapparna, menyerna och alla andra visuella eller interaktiva element. Front-end är vanligtvis byggt med hjälp av webbteknologier som HTML, CSS och JavaScript. Det finns också andra teknologier som har byggts på dessa grunder, ett bra exempel är typescript. Utöver dessa teknologier har det också byggts massor med frameworks som react, angular, mithril, svelte och andra.

Backenden av en webbapplikation är den del som hanterar applikationens logik, samt proccesserar data före det förvaras i databasen och före data skickas till användaren. Backenden är ansvarigt för att ta emot och bearbeta förfrågningar från front-end, hantera data (möjligtvis spara eller förfråga efter mera data från databasen) och skicka svar tillbaka till front-end. Back-end är vanligtvis byggd med hjälp av teknologier på serversidan som PHP, Python, Ruby eller Java.

Databasen för en webbapplikation är den del som lagrar data. Det är där all data som webbapplikationen behöver för att fungera lagras, beroende på vad webbapplikationen gör kan databasen lagra information såsom användarprofiler, produktinformation, beställningar. Databasen kan lagras på samma server som webbapplikationen, eller så kan den finnas på en separat server.

Webbservrar som Apache och Nginx är viktiga komponenter i det moderna internet. Dessa program fungerar som mellanhänder mellan webbläsare och applikationens filer och skripter. På mest grundläggande nivå fungerar webbservrar genom att ta emot förfrågningar från webbläsare och svara med innehållet. Om begäran är en statisk fil, såsom en HTML-, CSS- eller bildfil, kan servern enkelt läsa filen från disken och skicka den tillbaka den till webbläsaren. I kontrast om begäran är för en dynamisk sida måste dock servern först skicka begäran till en modul eller plugin som kan köra det nödvändiga skriptet eller koden som generera ett svar.

Apache och Nginx erbjuder dock en mängd funktioner och alternativ som gör att de kan skräddarsys efter behoven hos enskilda webbplatser och applikationer. Utöver sin roll som webbservrar kan Apache och Nginx även användas som omvända proxyservrar och lastbalanserare. Omvända proxyservrar kan hjälpa till att skydda webbplatser från attacker genom att fungera som en mellanhand mellan webbläsare och den webbapplikationen, medan lastbalanserare kan fördela inkommande trafik över flera servrar för att säkerställa att belastningen är jämnt fördelad och att ingen server blir överbelastad.

I dagens läge finns det många olika database typer, bland annat relations- och objektorienterade databaser. Relationsdatabaser lagrar data i tabeller med en fast struktur, medan objektorienterade databaser lagrar data som objekt som har både data och funktioner. Objektorienterade databaser stöder inkapsling, polymorfism och andra objektorienterade principer, vilket gör dem bättre lämpade för komplexa applikationer. Relationsdatabaser är mer lämpade för enklare applikationer som kräver vanliga frågemekanismer.

Border jag också ta upp typ Graph databaser eller blir det för mycket?

I en 3-lagers webbapplikation har varje lager sin egen specifika funktionalitet och beroende av lagren under det. Så alla ändringar som görs i ett lager kan potentiellt påverka lagren ovanför det. Till exempel tar bort en kolumn från databasen, men backend fortfarande förväntar sig kolumnen, kan det orsaka problem som programkrascher eller fel i frontend. På liknande sätt, om du gör ändringar i frontend-lagret, kan det också påverka backend- och

databaslagren. Det är bland annat dessa problem jag kommer lyfta fram och framföra lösningar till.

Ta upp skalbarhet och hur det påverkar applikationen. Samt cloud?.

Small-mid är det inte lönsamt och köpa enorma server setups med docker, loadbalancing, etc... Kommer därför att satsa på mindre applicationer som ändå har en database, back och front end.

Börja också och skriva huvudämnet (kontributionen), blue green metoden, hitta andra färdiga metoder.

Skriv en backgrund, vad är en webapplicakgtion, vad innebär det att uppdater med 0 ner tid. Finns det något skrivet om 0 downtime update. Har någon redan skrivit om det, om ej hur skall det definieras.

I detta arbete kommer jag att använda mig av ett enkelt abstrakt projekt som bygger sig på 3-lagers webbsystem principen. System använder PostgreSQL för databasen, Django REST-ramverket för backend och React för frontend. Som problem så måste namnet på en tabell i databasen ändras och frontend-lagret använder tabellnamnet för att fråga efter objekten från REST API. Om man bara skulle uppdatera namnet på alla ställen samtidigt skulle det resultera i fel eller crashar i applikationen. Detta beror på att backend-lagret antar att tabell namnet existerar vilket det inte gör ifall inte uppdateringarna sker på exakt samma sekund. Samma gäller också för frontend-lagret och backend-lagret. På vilket sätt kan man då göra en sådan här uppdatering utan stillestånd?

Blågröna uppdaterings metoden

Den blågröna uppdaterings metoden är en populär teknik som används i programvaruutveckling för att uppdatera webapplikationer utan stillestånds och minimal risk för fel. Det innebär att upprätthålla två identiska produktionsmiljöer: en "blå" miljö, som representerar den aktuella liveversionen av applikationen, och en "grön" miljö, som representerar den uppdaterade versionen.

Innan uppdateringsprocessen börjar ställs in den gröna miljön med alla nödvändiga ändringar eller uppdateringar. Den testas sedan noggrant för att säkerställa att den fungerar korrekt och alla tester godkänns. När den gröna miljön är klar växlas trafiken gradvis från blå miljö till grön miljö med hjälp av tillexempel en lastbalanserare.

Denna process kan göras i små steg, vilket minimerar eventuella störningar för användarna. Om några problem eller fel upptäcks under distributionsprocessen kan trafiken snabbt växlas tillbaka till den blå miljön för att hålla tillgängligheten för applikationen. Denna återställningsprocess kan göras snabbt, utan någon betydande inverkan på användarna. När den gröna miljön hanterar all trafik, kan den blå miljön stängas av och eventuella återstående

resurser kan allokeras till den gröna miljön. Denna process slutför implementeringen och den uppdaterade applikationen är nu live och tillgänglig för användarna.

En konsekvens av den blågröna uppdateringsmetoden är när det kommer till kostnaden eftersom den kräver underhåll av två identiska produktionsmiljöer. Detta innebär att det finns en dubblering av resurser, såsom servrar, lagring och nätverksutrustning. Detta kan öka den totala kostnaden.

En annan konsekvens av den blågröna uppdateringsmetoden är när det kommer till uppdatering samt synkroniseringen av databasen, dessa saker kan vara tidskrävande och komplicerat. Dessutom, om databasändringarna inte hanteras korrekt kan det leda till dataförlust och korruption.

Rullande uppdateringar

Frontend uppdatering

Backend uppdatering

Databas uppdatering

Kostnadsanalys

Referenser

http://www.gslb.cn/Load-Balancing.pdf

https://litslink.com/blog/web-application-architecture

https://ibm.github.io/data-science-best-practices/architecture.html

https://ojs.journals.cz/index.php/RJEBI/article/view/230

http://iwgcr.org/wp-content/uploads/2014/03/downtime-statistics-current-1.3.pdf

https://blogs.gartner.com/andrew-lerner/2014/07/16/the-cost-of-downtime/