LIGHTS IN LINE

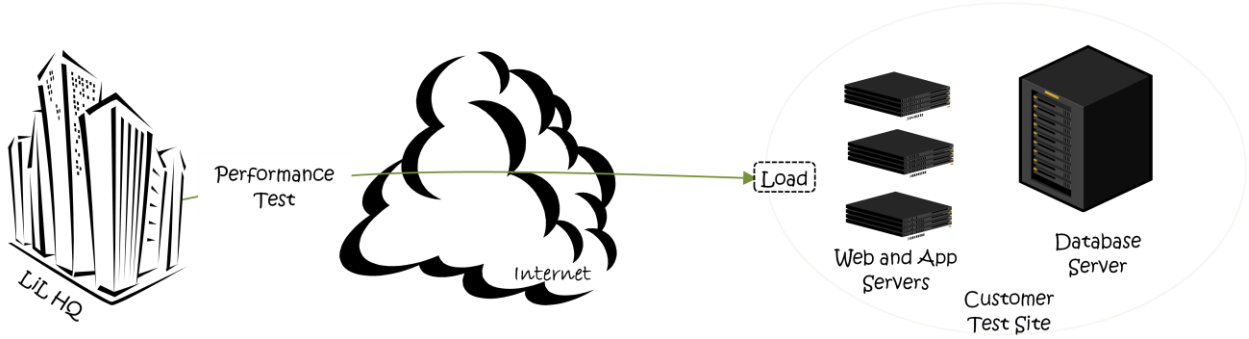
Prestandatest
Förberedelser & Faktainsamling

LIGHTS IN LINE AB
Tegnérgatan 37 |111 61 STOCKHOLM
info@lightsinline.se

Innehåll

1 Introduktion

Detta dokument ger en översikt av vilka förberedelser som behöver göras innan ett prestandatest. Det är Lights In Line's (LIL) erfarenhet att första gången en applikation/system skall prestandatestas går den största delen av tiden åt just till att samla ihop information och att iordningställa testmiljön. Framför allt kan väntetiderna bli långa.



2 Sammanfattning

För att målen med testerna skall nås snabbt och effektivt bör kunden tillhandahålla följande:

- Projektansvarig/ Kontaktperson (en och samma person kan ju ha alla roller som beskrivs här).
- Person med kompetens avseende applikationen/arkitektur som skall testas, detta för att LIL snabbt kunna få hjälp vid analyser i applikationen/arkitekturen.
- Person med kunskap avseende applikationen och systemets faktiska användning (Användningsfall) vid riktig drift, samt förståelse för prestanda målen.
- Person med kunskap avseende testmiljö/produktionsmiljö, som dessutom har rätt att konfigurera målmiljön, konton etc.
- En testmiljö/produktionsmiljö med korrekt konfiguration samt testdata som prestandatester kan utföras mot
- En miljö för placering av LIL s mätutrustning och prestandautrustning (Virtuell eller fysisk).
- Access för LIL till alla maskinerna som omfattas av testet, helst över Internet/VPN.
- I förekommande fall testkonton samt testdata i erforderlig omfattning

3 Testmetoder

1) Svarstidsanalys, för att se hur svarstiden utifrån användarens perspektiv fördelar sig mellan komponenter såsom webbläsare, kommunikation, webbserver/applikationsserver samt databashanterare. Utförs genom att överenskommet Användningsfall utförs med klienten (typiskt en webläsare) och mäts upp med ett ”lyssnarprogram”, t.ex. Wireshark.
2) Samtidighet – Klarar System under test (SUT) att två eller flera användare genomför samma handling (t.ex. loggar in) utan att det föreligger några former av samtidighetsproblem (låsningar på data basnivå, enbart seriell tillgång till vissa resurser etc etc).?
3) Skalbarhetstester - genom att studera applikationernas beteende vid viss last får man en indikation om hur väl den kommer att skala, dvs. man kan förbättra prestandan genom mera hårdvara. Genom detta förfarande behöver inte testmiljöns hårdvara motsvara tänkt produktionshårdvara. (för övrigt skall ju prestandatester fastslå vilken dimension det behövs på hårdvaran). Skalbarhetstester kan genomföras enligt följande metod:
   - SUT belastas med trafiken från 1 Virtuell User (VU) i 5 minuter. Genomsnittlig genomströmning och svarstider noteras.
   - Systemet belastas därefter med fler och fler VU i 5 minuters intervaller och resultaten noteras i en tabell. I ett lågt belastat system skall genomströmningen öka linjärt med lasten (och svarstiderna skall vara konstanta), men efterhand kommer genomströmningen plana ut (eller tom. minska) pga, något i systemet har blivit en flaskhals. I stället kommer svarstiderna öka pga. köbildning. När detta sker avslutas testet.

- Resultaten redovisas i tabell och grafer.
- Performance loggar från målsystemet analyseras, och relevanta counters plottas och jämförs med mätresultaten. Målet är att identifiera vilken del av systemet som var flaskhalsen, t.ex. CPU i en webserver.
- I värsta fall kan inte någon flaskhals identifieras i performance datan, utan måste hittas någon annan stans: det kan röra sig om anrop till externa system som inte har övervakats, eller köbildning pga. konfigurerade resurser, t.ex. thread eller connection pools.

4) Stabilitetstest - Utföra tester under längre tid, över en natt och eller 24/48 timmar, just för att upptäcka några former av minnesläckage (resurser återlämnas ej, och servern blir till slut överlastad)
5) Stresstest- Målet är att överbelasta SUT för att i kontrollerad miljö påvisa vilka egenskaper som SUT har, kraschar SUT, blir svarstiden enbart längre, eller ger SUT några andra indikationer.

4 Prestandamål och Testfall

En av utmaningarna vid prestandatester är att omsätta verksamhetens prestandamål vad applikationen skall klara av till mätbara mål som kan användas vid prestandatesterna. Genom en dialog med verksamheten i kombination med nuvarande produktionsdata och nuvarande konfigurationsuppsättningar så kan man komma ner till en modell för mätbara mål.

Viktigt i processen är att dokumentera resonemanget så att det blir transparant för verksamheten och de som skall utföra prestandatesterna och skriva rapporten.

Verksamheterna använder oftast termer som att applikationen exempelvis skall klara av t.ex. *5000 samtidiga användare*. Det är nu som vi prestandatestare tillsammans med verksamheten i flera moment skall bryta ner detta krav till mätbara mål.

Första momentet är att definiera vad användarna gör och omforma detta till användningsfall.

Här bör man inledningsvis välja ut ett till tre användningsfall som till delar uppfyller kriterierna :

- går relativt djupt i infrastrukturen,
- används ofta
- inte har för många steg

Exempel på ett användningsfalls steg kan vara :

- steg 1,logga in,
- steg 2, sök på status på mitt ärende,
- steg 3, logga ut.

Andra momentet är att omsätta ”Klara av ” till mätbara mål, exempelvis får svarstiden i genomsnitt inte överstiga 1 sekund för varje steg och svarstiden får aldrig vara över 10 sekunder.

Tredje momentet är att få koll på hur många ggr per dag dessa 5000 kommer att utföra ett användningsfall (AF) definierat i moment 1. Vi antar att dessa 5000 ”samtidiga” användare under 8 timmar (kontorstid kl 09 00 – 17 00) utför det definierade användningsfallet 1 gång. Belastningen blir då 5000/(8* 3600) = 0,17 användningsfall per sekund.

Denna strikt matematiska modell utgår ifrån det perfekta med jämn belastning över dagens alla sekunder och så är ju inte fallet i verkligheten. Vi tar oss igenom fortsatt resonemang i nästa moment.

Fjärde momentet är att få koll hur stor belastningen är under de mest intensiva timmarna, minuterna sekunderna. Strikt matematiskt så kan naturligtvis samtliga 5000 under samma sekund försöka utföra användningsfallet, sannolikheten är dock så liten att vi inte räknar med det. Nu kommer resonemanget med verksamheten:

- Under vilken eller vilka timmar används applikationen ?
- Finns det volymmätningar, exempelvis från web-loggarna, när trafiken är som intensivast

Exempelvis kommer vi fram till att de mest intensiva timmarna är mellan 10 – 11 och 14-15 och att 70 procent av användarna utför sitt användningsfall vid dessa 2 timmar. Efter dessa siffror blir belastningen 0,70*5000/(2*3600) = 0,49 användningsfall per sekund.

Det blir troligen inte här heller någon jämn belastning varje minut och sekund. Genom ytterliggare analys, samt att lägga på lite säkerhetsmarginal, kanske man når fram till att peak-belastningen vi skall klara är 4 * 0,49 , ca 2 användningsfall per sekund.
Genom dessa fyra moment har vi alltså brutet ner verksamhetens mål till mätbara mål , 5000 samtidiga användare genererar en last på 2 användningsfall per sekund och svarstiden för de enskilda stegen i användningsfallet skall i snitt vara 1 sekund.

Dags att testa !

5 Målsystem (System Under Test, SUT)

Det är att föredra att testerna kan utföras på ett system separat från driftsmiljön, eller åtminstånde utan att systemet används av någon annan samtidigt. Enligt vårt resonemang ovan (och beroende på målet med testen) så behöver inte testmiljön vara exakt likadan som driftsmiljön med avseende på hårdvaran.

När det gäller mjukvara, applikationen som testas samt innehåll i databaser etc. bör dessa vara identiska.

För referensändamål skall all HW, SW, appar, versioner, patchar etc. dokumenteras.

Om applikationen som testat anropar någon extern tjänst skall detta tydligt framgå.

Under själva testen skall SUT alla servrar övervakas. I MS miljö används Performance Monitor. Helst bör LIL kunna komma åt servrarna för att göra detta, och bör därför få användarkonton.

När det gäller databaser gäller specifikt att datat bör vara så produktionslikt som möjligt, framför allt när det gäller mängden poster. Innehåller testet någon form av skrivning mot databasen, bör man ha en metod för att kunna ”nollställa” den innan varje test.

6 Användardata

Som indata till Användningsfallen behöver man ofta en uppsättning användarnamn, lösenord mm. Detta bör vara konfigurerat i SUT innan test, och dessutom tillhandahållas i form av en .csv fil eller excel. All data bör var verifierad (rättigheter etc.) i förväg.

7 Lastgenerator

Testverktyget som används är oftast Microsoft Visual Studio Test Edition. Denna används både för att skapa testkoden (”Webbtest”) och för att generera lasten mot SUT. För detta krävs en eller flera (beroende på lasten) Windows maskiner, fysiska (kan tillhandahållas av LiL) eller virtuella. De kan vara placerade lokalt i anslutning till SUT, men även hos LiLs olika driftcentraler, beroende på behov av bandbredd. I de fall de placeras nära SUT, skall LiL helst beredas access till maskinerna från sitt kontor, t.ex. via VPN.

Krav på maskiner:

Typ	**Utrustad med**
Visual Studio	32bit, 2-4 CPU, 4GB RAM, 50 GB disk, Win7, Visual Studio Ultimate
*VS Load controller and test results repository	32bit, 2 CPU, 4GB RAM, 50 GB disk, Win Server 2008 (SQLexpress)
*VS Load Agent	32bit, 2 CPU, 2GB RAM, 10 GB disk, Win Server 2008

Bilaga - Begrepp

Begrepp	**Förklaring**
Användningsfall	En kedja av ett eller flera användningssteg inom en applikation
Användningssteg	En inom applikationen utförd handling av en användare
Svarstid för användningsfall	Summan av svarstiden för alla användningssteg (utan think time)
Svarstid för användningssteg	Tidsåtgång för att genomföra ett användningssteg (utan think time)
Think time	I testverktyg förekommande variabel för den betänketid en användare nyttjar mellan två användningssteg.
Elapsed time	Svarstid + think time för ett användningsfall eller steg
Bakgrundsbelastning	I produktionsmiljön förekommande nyttjande av de i systemet ingående resurserna
Peak-load	Högsta belastning på ett system under en viss tidsenhet, t.ex. timme
Svarstidsanalys	Manuell mätning av användningsfall med avseende på datamängd, svarstid och bandbredd för att snabbt kunna peka på eventuella problem som skulle kunna uppkomma vid större belastning (i.e. innan mer komplexa tester utförs)
Belastningstest -	Samlingsnamn för tester av varierande intensitet och längd
- referens/baseline	En lågintensiv genomkörning av t.ex. ett användningsfall för att skapa en referensmätning, för framtida bruk
- verifiering	För att verifiera att ett system uppfyller definierade prestandamål vid givna villkor
- stress	Vad händer när man överbelastar ett system ?
- stabilitet	Vad händer när man belastar ett system på ”normal” nivå under lång tid ?
- skalbarhet	Öka belastningen på ett system tills det börjar bli ”mättat”. Vilken del av systemet är då den svaga länken ?
Skript	Instruktioner till ett automatiskt testverktyg
VU	Virtuell användare vilken emulerar användningssteg hos verkliga användare
Genomströmning	Antal användningsfall som systemet förmår behandla per tidsenhet
PI	Prestandaindex, definieras som genomströmning/svarstid
Maximal genomströmning	Ett användningsfall körs först med en VU, därefter med stegvis ökande antal. Genomströmningen kommer initialt att öka i takt med antal VU, och i takt med att genomströmningen ökar, ökar normalt sätt även svarstiden. När den punkt där genomströmningen är som högst hittats definieras denna som systemets maximala genomströmning.
Optimal genomströmning	Om man ser svarstiden vid varje mätpunkt av genomströmningen som den kostnad genomströmningen har i svarstid, kan man identifiera en punkt där genomströmningen är som billigast, dvs *den punkt där vi får mest genomströmning för svarstiden*. Denna punkt definieras som systemets optimala genomströmning. Den fås fram genom att beräkna Prestanda Index (PI), och den punkt med högst PI ger systemets optimala genomströmning
B/p	Bytes per paket – en indikering på om nätverket används effektivt