**Engineer**

# Latens

väntan mellan frågan och den första delen av svaret

## Vad det är

Latens är den tid det tar innan ett system börjar svara: från att en begäran lämnar klienten tills den första byten av svaret är tillbaka. Den kallas också fördröjning, och för en nätverksväg som mäts fram och tillbaka talar man om rundturstid, på engelska round-trip time (RTT). Inom webbprestanda går samma idé under namnet time to first byte (TTFB). Svarstiden är dess större syskon och räknas ända till den sista byten av svaret.

Latens förväxlas lätt med genomströmning och bandbredd, men de tre svarar på olika frågor. Bandbredd är hur många bitar en länk kan bära per sekund. Genomströmning är hur mycket arbete ett system hinner med per sekund. Latens är hur länge ett enskilt arbete väntar. Ett batchjobb som körs varje natt kan ha utmärkt genomströmning och usel latens, och en chattserver det omvända.

Varje millisekund av den kommer från något av några få ställen:

- **utbredning**, när signalen färdas sträckan: ljus i optisk fiber hinner ungefär 200 km per millisekund, så en väg på 1 000 km kostar minst 10 ms fram och tillbaka innan någon programvara har kört;
- **överföring**, att lägga ut byten på ledningen, vilket märks för stora nyttolaster på långsamma länkar;
- **köande**, att vänta bakom annat arbete i en routers buffert, en trådpool, en anslutningspool eller ett lås, den enda delen som växer med lasten;
- **bearbetning**, själva arbetet: databasfrågan, mallen, krypteringen;
- **rundturer i protokollet**, samtalet innan den första användbara byten. En ny HTTPS-anslutning över TCP behöver först en TCP-handskakning och en TLS-handskakning, och TLS 1.3 kortade sin fullständiga handskakning till en enda rundtur.

IETF:s arbetsgrupp IP Performance Metrics spikade nätverksvarianten i RFC 2681: fördröjningen tur och retur löper från den första biten i ett paket som skickas till den sista biten i svaret som tas emot. Två av dess val går att föra över till vilken tjänst som helst. Ett svar som aldrig kommer räknas som odefinierat, informellt oändligt, i stället för att i tysthet lämnas utanför urvalet. Och den statistik som beskrivs är percentiler, medianen och minimum. RFC:n konstaterar också att minimum bara speglar utbredning och överföring, medan allt över minimum tyder på trängsel, en uppdelning som håller lika bra för ett API som för en väg mellan routrar.

För en enskild webbförfrågan delar curl upp väntan i sina steg:

```
curl -s -o /dev/null "$URL" -w '
dns        %{time_namelookup}
connect    %{time_connect}
tls        %{time_appconnect}
first byte %{time_starttransfer}
total      %{time_total}
'
```

Varje värde är sekunder sedan starten, så skillnaderna mellan dem är kostnaderna: att slå upp namnet, att öppna anslutningen, TLS-handskakningen och sedan serverns egen betänketid fram till den första byten.

Använd begreppet i ett beslut

Använd ”Vad är latens?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Övervakning och larm som säger till medan ett fel fortfarande är litet, och siffrorna som visar hur ofta det förblir så.

**Se våra tjänster** Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Byggde företagets första observerbarhetsdashboard — företagets första överblick över de egna systemen

**Byggde företagets första observerbarhetsdashboard** Hör av er →

## Varför det spelar roll

Latensen är det som användaren känner; genomströmningen är det som serverräkningen speglar. När en sida är långsam gäller beslutet var pengarna ska läggas. De fem komponenterna ovan behöver olika åtgärder; fyra av dem visar varför köp av hårdvara inte är en allmän lösning. Avstånd minskas genom att flytta närmare: en cache, ett CDN, en region nära användarna. Fördröjningen från

rundturer minskas genom att göra färre av dem: återanvända anslutningar, batcha databasfrågor och hämta parallellt. Bearbetningstiden kortas genom att göra mindre arbete: ett index, en bättre frågeplan. Mer kapacitet kan minska köandet; snabbare processorer, mer minne eller bättre I/O kan också korta bearbetningstiden när dessa resurser är flaskhalsen. En större server är inte automatiskt snabbare: mät var tiden går innan du väljer hårdvara.

Medelvärden får problemet att se mindre ut än det är, eftersom användarna möter svansen. En sida som anropar tio backend-tjänster parallellt väntar på den långsammaste. Om varje anrop oberoende av de andra har 1 % sannolikhet att vara långsamt, så träffar $1 - 0{,}99^{10}$, ca 9,6 % av sidladdningarna, minst ett långsamt anrop. En sällsynt fördröjning i delarna blir en vanlig fördröjning i helheten.

Därför skrivs seriösa mål i percentiler, t.ex. att 99 % av begärandena besvaras inom 300 ms under 30 dagar, och därför är den 99:e percentilen oftast den siffra som är värd att bevaka. Medianen säger hur tjänsten känns en bra dag; de höga percentilerna säger hur den känns för de kunder som är på väg att lämna.

## Så ser det ut i praktiken

I en geografisk kartapplikation drabbade latensen vid rendering av kartlager och vid frågor mot platsdata både användarna och backendens tillförlitlighet. Arbetet inriktades på bearbetning av spatiala databasfrågor i en backend som använde en äldre Microsoft SQL Server-databas. Flytten till PostgreSQL med PostGIS sänkte den genomsnittliga svarstiden från 2,5 sekunder till under 250 millisekunder, en förbättring av prestandan för spatiala databasfrågor med en faktor 10. Bakom det låg GiST- och SP-GiST-index på de spatiala kolumnerna, över 50 miljoner migrerade poster och en konfiguration som finjusterats för samtidig åtkomst; CPU-belastningen i backend minskade med 65 %.

Latens sätter också ett tak för genomströmningen där många transaktioner slåss om samma rad. Anteckningen om låset på en het lagerrad anger taket för en vara till ungefär det antal försäljningar som ett lås bär, delat med hur länge låset hålls. Allt som är långsamt inuti låset sänker taket: ett betalningsanrop som görs medan låset hålls drar ned det till ungefär två försäljningar per sekund. Till och med en skyddsåtgärd har ett pris här. En synkron standby i PostgreSQL lägger till latens vid varje commit, och det förlänger precis den låshålltid som taket delas med.

Och för att hitta vart tiden tar vägen registrerar den egenbyggda övervakningen på en plattform API-latens i sin egen databas och spårar begäranden med OpenTelemetry, så att en långsam begäran kan följas genom frontenden, API:t och databasen i stället för att gissas fram.

## Var det går fel

**Medelvärdet döljer svansen.** Ett medelvärde på 100 ms går ihop med att en begäran av femtio tar tre sekunder. Registrera latensen som ett histogram och läs av dess percentiler. Tänk sedan på hur de kombineras: den 99:e percentilen från fem servrar kan inte medelvärdesbildas till hela flottans 99:e percentil. Prometheus gör samma poäng i sin egen dokumentation, där kvantiler som beräknats av en summary inte kan aggregeras, medan histogrammets hinkar kan summeras först och kvantilen tas fram efteråt.

**Tidtagningen startar på fel ställe.** En server som mäter från det ögonblick då dess hanterare börjar missar tiden som en begäran tillbringade i lastbalanseraren, i acceptkön eller i väntan på en anslutning från poolen. Lastgeneratorer som väntar på varje svar innan de skickar nästa begäran skickar färre begäranden just när systemet står still, så stoppet syns knappt i resultaten. Mät från klienten också, och låt servern rapportera sin andel i en standardiserad header:

```
Server-Timing: db;dur=53, app;dur=47.2
```

**Samtalet är för pratigt.** En sida som kör en databasfråga per rad i en lista betalar en rundtur per rad. Med databasen i en annan region, 30 ms bort räknat tur och retur, kostar hundra rader tre sekunder innan någon fråga har gjort riktigt arbete. En enda fråga med en join, eller ett enda batchat anrop, betalar avståndet en gång.

**Omförsök mångfaldigar lasten.** En klient som ger upp efter en sekund och försöker igen tre gånger gör en långsam server till fyra gånger så mycket trafik, vilket gör den ännu långsammare. Ge varje begäran en total tidsgräns, vänta exponentiellt längre med slumpmässig spridning (jitter) mellan försöken och sätt ett tak för antalet omförsök.

# Läs mer

- RFC 2681: A Round-trip Delay Metric for IPPM — fördröjning tur och retur definierad ned till biten, där förlorade svar och percentiler hanteras uttryckligen.
- RFC 7679: A One-Way Delay Metric for IP Performance Metrics — motsvarigheten i en riktning, publicerad som internetstandard.
- Server Timing — W3C:s header som låter en server tala om för webbläsaren vart tiden tog vägen.
- Histograms and summaries — varför kvantiler från en summary inte kan kombineras, medan histogrammets hinkar kan det.
- RFC 8446: TLS 1.3 — handskakningens utformning, inklusive den rundtur som den tar bort.

Ordlista

Välj utifrån era faktiska förutsättningar

Ta med driftsättningsvägen, driftsriskerna och dem som har jour. Vi kan avgränsa en förändring med kontroller, en väg tillbaka och tydligt ansvar.

**Se vad det kostar** Hör av er →

## Gör begreppen praktiska

Fråga hur ”Vad är latens?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

**Se våra tjänster**

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

**Omsorgsfullt skapad med kärlek av D.**

**CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N**

**Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.**

**Utskrivet från https://engineer.company/sv/glossary/latency/**