Engineer

Latenstid

⏱ ventetiden mellem spørgsmålet og svarets begyndelse

Hvad det er

Latenstid, ofte bare latens, er den tid, et system er om at begynde at svare: fra en forespørgsel forlader klienten, til den første byte af svaret er tilbage. Man taler også om forsinkelse, om rundturstid (round-trip time, RTT), når en netværksstrækning måles frem og tilbage, og i webperformance om time to first byte (TTFB). Svartid er storebroren, der løber helt til svarets sidste byte.

Latenstid forveksles let med gennemløb og båndbredde, men de tre besvarer hvert sit spørgsmål. Båndbredde er, hvor mange bit en forbindelse kan bære i sekundet. Gennemløb er, hvor meget arbejde et system gør færdigt i sekundet. Latenstid er, hvor længe ét stykke arbejde venter. Et natligt batchjob kan have glimrende gennemløb og elendig latenstid, og en chatserver omvendt.

Hvert millisekund stammer fra et af få steder:

- **udbredelse**, signalets vej over afstanden: lys i optisk fiber tilbagelægger omkring 200 km pr. millisekund, så en strækning på 1.000 km koster mindst 10 ms frem og tilbage, før noget software overhovedet kører;
- **transmission**, at få bytes ud på ledningen, hvilket betyder noget for store datamængder på langsomme forbindelser;
- **kø**, ventetid bag andet arbejde i en routerbuffer, en trådpulje, en forbindelsespulje eller en lås, og den eneste del, der vokser med belastningen;
- **behandling**, selve arbejdet: forespørgslen, skabelonen, krypteringen;
- **protokolrundture**, samtalen før den første nyttige byte. En ny HTTPS-forbindelse over TCP kræver først et TCP-håndtryk og et TLS-håndtryk, og TLS 1.3 skar det fulde håndtryk ned til én rundtur.

IETF's arbejdsgruppe for IP Performance Metrics definerede netværksudgaven præcist i RFC 2681: rundturstiden løber fra den første bit af en pakke sendes, til den sidste bit af svaret er modtaget. To af valgene i den gælder for enhver tjeneste. Et svar, der aldrig kommer, tælles som udefineret, uformelt uendeligt, i stedet for at forsvinde stille ud af stikprøven. Og de statistikker, RFC'en beskriver, er percentiler, medianen og minimum. Den bemærker også, at minimum afspejler udbredelse og transmission alene, mens alt over minimum skyldes trængsel, en opdeling der holder lige så godt for et API som for en vej mellem routere.

For en enkelt webforespørgsel kan curl dele ventetiden op i faser:

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

Hvert tal er sekunder siden start, så afstanden mellem dem er prisen for hvert trin: navneopslaget, forbindelsen, TLS-håndtrykket og til sidst serverens egen betænkningstid frem til den første byte.

Brug begrebet i en beslutning

Brug »Hvad er latenstid?« i en konkret beslutning. Vi kan knytte begrebet til det system, I bygger eller køber. Overvågning og alarmer, der siger til, mens fejlen stadig er lille, og tallene, der viser, hvor tit den bliver ved med at være det.

Se vores ydelser Kontakt os →

Undersøg antagelserne bag ordet

Brug forklaringen og det tilknyttede arbejde til at formulere spørgsmål til jeres projekt. Vi kan undersøge antagelserne sammen, før et værktøj vælges. Har bygget det første observerbarhedsdashboard — virksomhedens første overblik over sine egne systemer

Har bygget det første observerbarhedsdashboard Kontakt os →

Hvorfor det betyder noget

Latenstid er det, brugeren mærker; gennemløb er det, serverregningen afspejler. Når en side er langsom, er spørgsmålet, hvor pengene skal bruges. De fem komponenter ovenfor kræver forskellige løsninger; fire af dem viser, hvorfor mere hardware ikke er en generel løsning. Afstand mindskes ved at flytte tættere på: en cache, et CDN, en region nær brugerne. Ventetiden ved rundture

mindskes ved at reducere deres antal: genbruge forbindelser, samle forespørgsler, hente parallelt. Behandlingstiden forkortes ved at lave mindre arbejde: et indeks, en bedre forespørgselsplan. Mere kapacitet kan nedbringe køventetiden; hurtigere processorer, mere hukommelse eller bedre I/O kan også forkorte behandlingstiden, når disse ressourcer er flaskehalsen. En større server er ikke automatisk hurtigere: mål, hvor tiden går, før du vælger hardware.

Gennemsnit får problemet til at se mindre ud, end det er, for brugerne møder halen. En side, der kalder ti backend-tjenester parallelt, venter på den langsomste. Hvis hvert kald uafhængigt af de andre har 1 % sandsynlighed for at være langsomt, rammer $1 - 0{,}99^{10}$, omkring 9,6 % af sidevisningerne, mindst ét langsomt kald. En sjælden forsinkelse i delene bliver en almindelig forsinkelse i helheden.

Derfor skrives seriøse mål i percentiler, for eksempel at 99 % af forespørgslerne besvares inden for 300 ms over 30 dage, og derfor er den 99. percentil som regel det tal, der er værd at holde øje med. Medianen fortæller, hvordan tjenesten føles på en god dag; de høje percentiler fortæller, hvordan den føles for de kunder, der er på vej væk.

Sådan ser det ud i praksis

I en geografisk kortapplikation gik latens ved rendering af kortlag og forespørgsler på lokationsdata ud over både brugerne og backendens pålidelighed. Arbejdet fokuserede på behandling af spatiale forespørgsler i en backend, der brugte en ældre database baseret på Microsoft SQL Server. Skiftet til PostgreSQL med PostGIS sænkede den gennemsnitlige svartid fra 2,5 sekunder til under 250 millisekunder, en forbedring af ydeevnen for spatiale forespørgsler med en faktor 10. Bag arbejdet lå GiST- og SP-GiST-indeks på de spatiale kolonner, over 50 millioner migrerede poster og en konfiguration tunet til samtidig adgang; backendens CPU-belastning faldt med 65 %.

Latenstid sætter også loft over gennemløbet, hvor flere transaktioner konkurrerer om én række. En note om flash-salg, hvor alle købere skriver til den samme række, sætter loftet for en enkelt vare til omtrent de salg, én lås bærer, divideret med den tid, låsen holdes. Alt langsomt inde i låsen sænker loftet: et betalingskald, mens låsen holdes, bringer det ned på omkring to salg i sekundet. Selv en sikkerhedsforanstaltning har en pris her. En synkron standby i PostgreSQL lægger latens på hver commit, og det forlænger præcis den holdetid, loftet divideres med.

Og for at finde ud af, hvor tiden går, registrerer den hjemmebyggede overvågning på én platform API-latenstid i platformens egen database og sporer forespørgsler over OpenTelemetry, så en langsom forespørgsel kan følges gennem frontend, API og database i stedet for at blive gættet på.

Hvor det går galt

Gennemsnittet skjuler halen. Et gennemsnit på 100 ms er foreneligt med, at hver halvtredsindstyvende forespørgsel tager tre sekunder. Registrér latenstid som et histogram, og læs percentilerne af det. Pas så på, hvordan de lægges sammen: den 99. percentil fra fem servere kan ikke gennemsnitsberegnes til hele flådens 99. percentil. Prometheus siger det selv i sin dokumentation: kvantiler beregnet af en summary kan ikke aggregeres, mens histogrammets spande kan lægges sammen først, hvorefter kvantilen beregnes.

Uret starter det forkerte sted. En server, der måler fra det øjeblik, dens handler begynder, overser den tid, forespørgslen brugte i load balanceren, i accept-køen eller på at vente på en forbindelse fra puljen. Belastningsværktøjer, der venter på hvert svar, før de sender den næste forespørgsel, sender færre forespørgsler, netop når systemet går i stå, så stoppet næsten ikke ses i resultaterne. Mål også fra klienten, og lad serveren melde sin andel i en standardheader:

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

Samtalen er for snakkesalig. En side, der kører én forespørgsel pr. række i en liste, betaler én rundtur pr. række. Står databasen i en anden region med 30 ms pr. rundtur, koster hundrede rækker tre sekunder, før nogen forespørgsel laver rigtigt arbejde. Én forespørgsel med en join, eller ét samlet kald, betaler afstanden én gang.

Genforsøg ganger belastningen op. En klient, der giver op efter et sekund og prøver igen tre gange, gør én langsom server til fire gange trafikken, hvilket gør den endnu langsommere. Giv hver forespørgsel en samlet tidsfrist, vent eksponentielt længere med tilfældig spredning mellem forsøgene, og sæt et loft over antallet af genforsøg.

Læs mere

- RFC 2681: A Round-trip Delay Metric for IPPM — rundturstid defineret ned til bitten, med tabte svar og percentiler behandlet eksplicit.
- RFC 7679: A One-Way Delay Metric for IP Performance Metrics — modstykket for envejsforsinkelse, udgivet som Internet Standard.
- Server Timing — W3C-headeren, der lader en server fortælle browseren, hvor tiden gik.
- Histograms and summaries — hvorfor kvantiler fra en summary ikke kan kombineres, og histogrammets spande kan.
- RFC 8446: TLS 1.3 — håndtrykkets design, herunder den rundtur, det sparer.

Ordliste

Vælg ud fra jeres faktiske vilkår

Medbring udrulningsforløbet, driftsrisiciene og dem, der har vagten. Vi kan afgrænse en ændring med kontroller, en vej tilbage og tydeligt ansvar.

Se, hvad det koster Kontakt os →

Gør begreberne praktiske

Spørg, hvordan »Hvad er latenstid?« gælder i jeres organisation. Et konkret eksempel fra jeres nuværende system er nok til at begynde samtalen.

Se vores ydelser

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

Tag denne side med dig

Visitkort

Lavet med kærlighed af D.

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

Copyright © 2025 – nu · Engineer ApS · Alle rettigheder forbeholdes.

Udskrevet fra https://engineer.company/da/glossary/latency/