Engineer Mjukvara

Prestandaoptimering

21 resultat

Koppla samman systemets delar

Arbetet under ”Prestandaoptimering” visar en del av ett fungerande system. Vi kan hjälpa er att koppla den delen till resten av verksamheten. Allt som händer på en plats — fartyg, fordon, tomter, sensorer — på en karta du kan söka i, mäta på och dela.

Se våra tjänster Hör av er →

Hitta en jämförbar utmaning

Öppna ett exempel som liknar er situation. Genomgången förklarar besluten bakom resultatet, utöver teknikens namn. Gjorde pipelinen för geodata 50 gånger snabbare — 50x snabbare datapipelines för geodata

Gjorde pipelinen för geodata 50 gånger snabbare Hör av er →

Gjorde pipelinen för geodata 50 gånger snabbare

Dokumenterade en förbättring på 50 gånger i en geografisk datapipeline i grundarens arbete hos Utiligize genom SQL- och datamodelländringar i PostgreSQL, MS SQL och Google Cloud BigQuery.

Gjorde GIS-kartan 10 gånger snabbare i PostgreSQL

Förbättrade prestandan i en kartapplikation för geodata 10 gånger genom en strategisk databasövergång från MSSQL till PostgreSQL, inklusive själva omställningen av data, och optimerade bearbetningen och datasäkerheten.

Löste 1 000 problem i geodata och tidsseriedata

Löste 1 000 problem i geodata och tidsseriedata med GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) och Bash, vilket säkrade en högkvalitativ bearbetning av stora datamängder.

Drev 100 datalager i PostgreSQL, MS SQL, BigQuery

Utformade, byggde upp och förvaltade 100 datalagerdatabaser i PostgreSQL, MS SQL och Google BigQuery, främst med GIS- och tidsseriedata, och optimerade prestanda och skalbarhet.

Styrde full stack-utvecklingen av GIS-kartor

Styrde full stack-utvecklingen av GIS-kartor med ansvar för PostgreSQL, Mapbox, ReactJS och NodeJS för att leverera en integrerad lösning.

Förnyade interna processer, sparade 8 000 timmar

Gjorde en genomgripande översyn av de interna processerna och sparade 8 000 timmar genom att förbättra mjukvaruarkitekturen, systemen och effektiviteten i schemaläggningen.

Administrerade 40 webbplatser på Apache och Nginx

Administrerade 40 webbplatser på värdservrar med Ubuntu Linux, Apache och Nginx och säkerställde hög tillgänglighet och prestanda.

Utformade en JSON-väg rakt från Postgres till API

Utformade en arkitektur för direkt vidarebefordran av JSON, där PostgreSQL-funktioner returnerar komplett JSON som Go-API:t skickar vidare ordagrant, vilket tog bort mellanliggande deserialisering och frikopplade frontenden från schemaändringar.

Modellerade 348 tabeller i 34 Postgres-scheman

Modellerade sjöfartsdomänen i 348 normaliserade tabeller i 34 PostgreSQL-scheman — yrkesverksamma, företag, fartyg, jobb, recensioner och resten — med SMALLINT-uppslag och UUID v7-nycklar.

Byggde Next.js 16-frontenden: SSR, SSG och CSR

Byggde frontenden i Next.js 16 med medvetna SSR-, SSG- och CSR-strategier och en återanvändbar fabrik för förhämtade serversidor som tar bort N+1-kaskaden av hämtningar från varje inloggad sida som flyttas över till den.

Levererade fullt stöd för Progressive Web App

Levererade fullt stöd för Progressive Web App — installerbar och med offlinestöd — med körtidscachning i Workbox via next-pwa.

Flyttade långsamt arbete till en jobbkö i River

Flyttade långsamt arbete från anropsvägen till en jobbkö i River — 15 workermoduler, 8 schemalagda uppgifter och 20 pg_cron-jobb — så att ett anrop returnerar medan arbetet bakom det fortsätter.

Byggde hastighetsbegränsning som stänger vid fel

Byggde skydd mot missbruk som stänger vid fel — 22 hastighetsbegränsare i Redis, Cloudflare Turnstile, idempotenta anrop och ett ursprungslås — så att plattformen avvisar botar och anropsfloder i stället för att lita på sina anropare.

Byggde en lojalitetsplånbok, ligor och märken

Byggde lojalitets- och ryktessystemet — 67 funktioner över en huvudbok i 31 tabeller, med ligor, märken och en inlösenbutik — som tar ett radlås på saldot för att stänga fönstret för dubbelspendering.

Offentliga tjänster på en liten server

Drev företagets offentliga tjänster på en värd med en kärna och 512 MB — en git-forge, en webbserver för sju domäner, Tor, två servrar för alternativa protokoll och intrångsblockering — med 464 MB användbart minne som designbegränsning; backupautomatik är en separat implementation.

Byggde en flerspråkig Hugo-webbplats på 3 språk

Byggde en flerspråkig statisk webbplats i Hugo med 108 mallfiler, varav 53 partials, som publicerar varje sida i fyra representationer från ett innehållsträd på tre språk, och en gång till under sju fokuserade underdomäner byggda från samma träd.

Ett skyddsnät som inte längre kostar en halvtimme

Kortade webbplatsens webbläsarstyrda kvalitetsgrind från 1 636 sekunder till 615 genom att schemalägga kontrollerna med den längsta först i en workerpool begränsad till fyra spår, efter att ha mätt att alfabetisk ordning kostade 320 sekunder mot 224.

Krympte stilmallspaketet från 87 KB till 31 KB

Krympte stilmallspaketet från 87 KB till 31 KB genom att ta ut ett base64-kodat typsnitt ur det, och tog bort 756 KB i 22 filer som publicerades vid varje driftsättning utan att något refererade till dem.

Byggde in en QR-kodare på 522 rader

Byggde in en QR-kodare — Reed-Solomon över GF(256), fast modullayout, åtta maskmönster — på 522 rader i stället för att ta in ett tredje körtidsberoende, och verifierade den genom att läsa tillbaka den färdiga matrisen med en oberoende skriven avkodare.

Ett begränsat händelseflöde i en macOS-app

Begränsade händelsebuffertar och loggning efter en dokumenterad macOS-incident med en tillväxt på 41 MB i sekunden och 111 GB komprimerade sidor på en maskin med 36 GB; en dokumenterad fullständig testkörning minskade kärnans loggrader från 610 996 till 1 411.

Införde strikt samtidighet i Swift 6 ovanpå C

Införde fullständig strikt samtidighet i Swift 6 utan actors och överbryggade en blockerande händelseloop i C till main actor genom en producent, en konsument och en ordning — efter att ha konstaterat att en task per händelse förlorar den ordning som gränssnittet är beroende av.

Fokusera den tekniska insatsen

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Sätt ämnet i ert sammanhang

Berätta hur ”Prestandaoptimering” ingår i ert system. Vi kan hjälpa er att välja rätt tjänst och avgränsa arbetet.

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://software.engineer.company/sv/categories/performance/