Engineer

Performanceoptimierung

30 Leistungen

Die Teile Ihres Systems verbinden

Die unter „Performanceoptimierung“ gesammelten Arbeiten zeigen einen Teil eines funktionierenden Systems. Wir können helfen, diesen Teil mit dem Rest Ihres Betriebs zu verbinden. Die Abfragen und fehlenden Indizes hinter einem von Jahr zu Jahr schwerfälligeren System finden und die entscheidenden Probleme beheben.

Unsere Leistungen entdecken Kontakt aufnehmen →

Eine vergleichbare Herausforderung finden

Folgen Sie einem Beispiel, das Ihrer Situation ähnelt. Die Rückschau beschreibt die Entscheidungen hinter dem Ergebnis und geht über die bloße Technologiebezeichnung hinaus. 30 Tage Stromdaten in 180 ms — 180 ms für die Aggregation eines Monats an Zählerständen

30 Tage Stromdaten in 180 ms Kontakt aufnehmen →

30 Tage Stromdaten in 180 ms

Varianten zur stündlichen Aggregation des Stromverbrauchs in Python, SQL und Bash + Jq entwickelt; der Python-Benchmark verarbeitete einen 30-Tage-JSONL-Datensatz in 180 ms.

Geografische Datenpipeline 50-mal schneller

Eine 50-fache Verbesserung der geografischen Datenpipeline bei der Arbeit der Gründerin für Utiligize durch Änderungen an SQL und Datenmodellen in PostgreSQL, MS SQL und Google Cloud BigQuery dokumentiert.

GIS-Karte mit PostgreSQL 10-mal schneller

Die Performance einer geografischen Kartenanwendung durch einen gezielten Datenbankwechsel von MSSQL zu PostgreSQL einschließlich Datenumschaltung um das 10-Fache verbessert und dabei Verarbeitung und Datensicherheit optimiert.

1.000 Fehler in geografischen Daten behoben

1.000 Probleme in geografischen Daten und Zeitreihendaten mit GDAL, ArcGIS, PostGIS, Mapbox, QGIS, SQL (PL/pgSQL, Transact-SQL) und Bash behoben und eine hochwertige Big-Data-Verarbeitung sichergestellt.

100 PostgreSQL/MS-SQL/BigQuery-Warehouses

100 Data-Warehouse-Datenbanken in PostgreSQL, MS SQL und Google BigQuery mit überwiegend GIS- und Zeitreihendaten konzipiert, aufgebaut und verwaltet sowie Performance und Skalierbarkeit optimiert.

10 PostgreSQL-/MS-SQL-Server auf Ubuntu

10 PostgreSQL- und MS-SQL-Server auf Ubuntu-Linux-VPS entworfen, bereitgestellt und gewartet und dabei Serverperformance und Zuverlässigkeit sichergestellt.

PostgreSQL um das 10-Fache beschleunigt

PostgreSQL durch gezielte Indizierung, Partitionierung und Abfrageoptimierung um das 10-Fache beschleunigt und die Datenbankeffizienz für Nutzer-, Mandanten-, Geodaten und Strom-Zeitreihendaten verbessert.

Full-Stack-GIS-Kartenentwicklung geleitet

Die Full-Stack-Entwicklung einer GIS-Kartenanwendung mit PostgreSQL, Mapbox, ReactJS und NodeJS geleitet und eine integrierte Lösung umgesetzt.

Abläufe überarbeitet und 8.000 Stunden eingespart

Interne Abläufe überarbeitet und durch bessere Softwarearchitektur, Systeme und Terminplanung 8.000 Stunden eingespart.

150 Stunden monatlich in 30 Projekten gespart

Die Projekteffizienz durch optimierte Arbeitsabläufe und Ressourcenplanung verbessert und über 30 Projekte hinweg monatlich 150 Stunden eingespart.

40 Websites mit Apache und Nginx administriert

40 Websites auf Ubuntu-Linux-Hostingservern mit Apache und Nginx administriert und hohe Verfügbarkeit und Performance sichergestellt.

600 PL/pgSQL-ETL/ELT-Pipelines entwickelt

600 PL/pgSQL-basierte ETL/ELT-Pipelines für effizientere komplexe Datenverarbeitung in mehreren PostgreSQL-Entwicklungs- und Produktivumgebungen entwickelt.

JSON direkt von Postgres an die API geleitet

Eine JSON-Passthrough-Architektur entworfen, in der PostgreSQL-Funktionen vollständiges JSON zurückgeben, das die Go-API unverändert weiterreicht; damit entfällt die zwischenzeitliche Deserialisierung und das Frontend bleibt von Schemaänderungen entkoppelt.

Zeitlich geordnete Schlüssel für ein wachsendes Datenmodell

Zeitlich geordnete UUID-v7-Identifikatoren (PostgreSQL 18) als Entitätsschlüssel eingeführt, um die Fragmentierung von B-Baum-Indizes zu verringern und Abfragen zu beschleunigen.

348 Tabellen in 34 Postgres-Schemata modelliert

Die maritime Domäne in 348 normalisierten Tabellen über 34 PostgreSQL-Schemata modelliert — Fachkräfte, Unternehmen, Schiffe, Stellen, Bewertungen und weitere Bereiche — mit SMALLINT-Nachschlagetabellen und UUID-v7-Schlüsseln.

Next.js-16-Frontend: SSR, SSG und CSR entwickelt

Das Next.js-16-Frontend mit bewusst gewählten SSR-, SSG- und CSR-Strategien sowie einer wiederverwendbaren Factory für vorab geladene Serverseiten entwickelt, die die N+1-Abfragekaskade auf jeder umgestellten authentifizierten Seite beseitigt.

Progressive Web App vollständig unterstützt

Vollständige Progressive-Web-App-Unterstützung umgesetzt — installierbar und offlinefähig — mit Workbox-Laufzeitcaching über next-pwa.

Langsame Aufgaben in River-Jobqueue verlagert

Langsame Verarbeitung aus dem Anfragepfad in eine River-Jobqueue verlagert — 15 Worker-Module, 8 geplante Aufgaben und 20 pg_cron-Jobs — sodass Anfragen zurückkehren, während die zugehörige Arbeit weiterläuft.

Bei Fehlern sperrenden Missbrauchsschutz gebaut

Bei Fehlern sperrende Missbrauchsschutzmechanismen aufgebaut — 22 Redis-gestützte Ratenbegrenzer, Cloudflare Turnstile, Anfrage-Idempotenz und eine Ursprungssperre — damit die Plattform Bots und Überlastungsangriffe abweist, statt ihren Aufrufern zu vertrauen.

Eigene Analysen in PostgreSQL entwickelt

Eigene Produktanalysen in PostgreSQL entwickelt — 47 Funktionen über partitionierten Ereignistabellen mit automatischer Bereinigung — pseudonymisiert mit wechselndem Salt und nur mit Einwilligung des Besuchers.

Loyalitätskonto, Ligen und Abzeichen entwickelt

Das Loyalitäts- und Reputationssystem entwickelt — 67 Funktionen über ein Hauptbuch mit 31 Tabellen, mit Ligen, Abzeichen und einem Prämienshop — und den Kontostand durch eine Zeilensperre gegen doppelte Ausgaben abgesichert.

Öffentliche Dienste auf einem kleinen Server

Die öffentlichen Unternehmensdienste auf einem Single-Core-Host mit 512 MB gehostet — Git-Forge, Webserver für sieben Domains, Tor, zwei Server für alternative Protokolle und Angriffssperren — mit 464 MB nutzbarem Arbeitsspeicher als Entwurfsgrenze; Sicherungsautomatisierung ist eine getrennte Implementierung.

Totmannüberwachung für Hosts entwickelt

Eine Totmannüberwachung entwickelt, die nur bei gesundem Speicher- und Plattenzustand Signale sendet, sodass ein beeinträchtigter Host durch sein Verstummen alarmiert — und sechs Variablennamen mit "free" gefunden, deren Prüfungen korrekt "available" maßen, auf einem Rechner mit 464 MB um eine Größenordnung verschieden.

Hugo-Website mehrsprachig in 3 Sprachen aufgebaut

Eine mehrsprachige statische Hugo-Website mit 108 Vorlagendateien einschließlich 53 Partials aufgebaut, die jede Seite aus einem Inhaltsbaum in drei Sprachen und vier Darstellungsformen veröffentlicht, zusätzlich auf sieben spezialisierten Subdomains aus demselben Baum.

Ein Sicherheitsnetz, das keine halbe Stunde mehr kostet

Die browsergestützte Qualitätsprüfung der Website von 1.636 auf 615 Sekunden verkürzt, indem die Prüfungen in einem auf vier parallele Ausführungen begrenzten Worker-Pool nach längster Laufzeit zuerst eingeplant werden, nachdem die alphabetische Reihenfolge 320 statt 224 Sekunden benötigt hatte.

Stylesheet von 87 KB auf 31 KB verkleinert

Das Stylesheet-Bündel durch das Auslagern einer Base64-Schrift von 87 KB auf 31 KB verkleinert und 756 KB in 22 bei jeder Bereitstellung veröffentlichten, aber nirgends referenzierten Dateien entfernt.

QR-Encoder mit 522 Zeilen eingebunden

Einen QR-Encoder mit Reed-Solomon über GF(256), festem Modullayout und acht Maskenmustern in 522 Zeilen eingebunden, statt eine dritte Laufzeitabhängigkeit einzuführen, und durch Rücklesen der fertigen Matrix mit einem unabhängig geschriebenen Decoder geprüft.

Eine begrenzte Ereignispipeline für eine macOS-App

Ereignispuffer und Protokollierung nach einem dokumentierten macOS-Vorfall mit 41 MB Wachstum pro Sekunde und 111 GB komprimierten Speicherseiten auf einem Rechner mit 36 GB begrenzt; ein dokumentierter vollständiger Testlauf reduzierte Kernprotokollzeilen von 610.996 auf 1.411.

Swift-6-Nebenläufigkeit über C abgesichert

Vollständige strikte Nebenläufigkeit von Swift 6 ohne eigene Actors eingeführt und eine blockierende C-Ereignisschleife über genau einen Produzenten, einen Verbraucher und eine Reihenfolge an den Main Actor angebunden, nachdem ein Task je Ereignis die für die Oberfläche nötige Reihenfolge verloren hatte.

Testsuite auf 8,9 Sekunden beschleunigt

Die Testsuite von sechs Tests jenseits eines Limits von sechzig Sekunden auf 135 erfolgreiche Tests in 8,9 Sekunden gebracht, indem der Hauptthread profiliert und die zwei Aufrufe entfernt wurden, in denen er bei 3.989 von 4.017 Stichproben festhing.

Die technische Arbeit gezielt einsetzen

Bringen Sie einen repräsentativen Datensatz, die Fragen, die er beantworten muss, und den aktuellen Engpass mit. Wir können vereinbaren, wie Qualität, Durchsatz und Wiederherstellung gemessen werden.

Preise ansehen Kontakt aufnehmen →

Das Thema in Ihren Kontext einordnen

Erzählen Sie uns, welche Rolle „Performanceoptimierung“ in Ihrem eigenen System spielt. Wir können helfen, die passende Leistung auszuwählen und die damit verbundene Arbeit zu definieren.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/categories/performance/