Engineer

Skalierbarkeit

↗ Wie weit mehr Maschinen allein zusätzliche Last bewältigen

Was es ist

Skalierbarkeit beschreibt, wie die Kapazität eines Systems wächst, wenn Ressourcen hinzukommen. Ein System skaliert gut, wenn doppelt so viele Maschinen bei gleichen Antwortzeiten ungefähr die doppelte Last tragen. Es skaliert schlecht, wenn die zusätzliche Hardware meist auf etwas wartet, bei dem sie nicht helfen kann. Das Wort beschreibt ein Verhältnis zwischen Last und Kosten, keine Eigenschaft, die ein System entweder hat oder nicht hat.

Es gibt zwei Wachstumsrichtungen. Vertikales Skalieren bedeutet eine größere Maschine: mehr Kerne, mehr Speicher, schnellere Datenträger. Es erfordert keine Softwareänderung und endet bei der größten Maschine, deren Kauf sich lohnt. Horizontales Skalieren bedeutet mehr Maschinen, die sich die Arbeit teilen. Es hat diese Obergrenze nicht, nützt aber nur teilbarer Arbeit.

HTTP wurde mit Blick auf horizontales Skalieren gestaltet. RFC 9110 merkt an, dass viele Implementierungen die Zustandslosigkeit von HTTP nutzen, um über Proxys geführte Verbindungen wiederzuverwenden oder Anfragen dynamisch auf mehrere Server zu verteilen. Eine Webschicht, die zwischen Anfragen nichts speichert, kann durch weitere Kopien hinter einem Loadbalancer wachsen. Jede Kopie kann jede Anfrage beantworten.

Last hat mehrere Dimensionen. Ein System kann entlang einer skalieren und entlang einer anderen stocken: Anfragen pro Sekunde, gespeichertes Datenvolumen, Zahl der Mandanten und Zahl der Schreiber, die um denselben Datensatz konkurrieren. Die letzte ist am schwierigsten. Eine einzelne Zeile kann immer nur von einer Transaktion gleichzeitig geändert werden, gleichgültig, wie viele Server darum stehen.

Dahinter steht Amdahls Gesetz. Muss ein Teil der Arbeit nacheinander laufen, beschleunigt parallele Hardware nur den Rest. Der sequenzielle Anteil setzt deshalb die Grenze: Bei 5 % serieller Arbeit kann die ganze Aufgabe nie mehr als 20-mal schneller laufen, selbst auf tausend Maschinen. In einer Datenbank ist der serielle Teil oft eine Sperre. In einer Anwendung ist es oft ein gemeinsam genutztes Element: ein Zähler, eine Warteschlange oder ein Sitzungsspeicher.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Skalierbarkeit?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Monitoring und Alarme melden sich, solange ein Fehler noch klein ist; Kennzahlen zeigen, wie oft das gelingt.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Das erste Observability-Dashboard aufgebaut — der erste Einblick des Unternehmens in die eigenen Systeme

Das erste Observability-Dashboard aufgebaut Kontakt aufnehmen →

Warum es wichtig ist

Skalierbarkeit entscheidet, was Wachstum kostet. Skaliert ein System gut, ist die Verdoppelung des Verkehrs eine Zeile auf einer Rechnung. Skaliert es schlecht, wächst die Rechnung schneller als die Last, oder zusätzliche Hardware hilft überhaupt nicht mehr. Die Lösung wird dann ein Umbau, während der Verkehr weiter eintrifft.

Hardware kann Engpässe bei Prozessoren, Speicher oder Datenträgern lindern. Ineffiziente Arbeit und erzwungene sequenzielle Verarbeitung können aber bleiben. Eine Abfrage, die unnötig eine ganze Tabelle durchsucht, kann auch auf einem größeren Server Arbeit verschwenden. Eine größere Maschine beseitigt nicht die sequenzielle Ausführung von Aktualisierungen einer gemeinsamen Zeile. Mehr Anwendungsserver können weitere Schreiber in die Warteschlange stellen. Die meisten Systeme, die scheinbar mehr Maschinen brauchen, müssen zuerst genauer untersucht werden.

In PostgreSQL beginnt diese Untersuchung bei `pg_stat_statements`. Die Erweiterung zeichnet jede normalisierte Anweisung samt Aufrufzahl und Ausführungszeit auf. Nach Gesamtzeit statt Mittelwert sortiert zeigt sie, womit der Server seine Zeit tatsächlich verbringt: Eine 2 ms dauernde Abfrage, die eine Million Mal pro Stunde läuft, kostet 2.000 Sekunden dieser Stunde. Ein zweimal ausgeführter Bericht mit zwei Sekunden Laufzeit kostet vier.

```
SELECT calls, round(total_exec_time) AS total_ms,
       left(query, 60) AS query
  FROM pg_stat_statements
 ORDER BY total_exec_time DESC
 LIMIT 10;
```

Jeder Schritt zu mehr Maschinen bringt eigene Betriebskosten mit. Deshalb zählt die Reihenfolge. Eine effizientere Anweisung oder ein besserer Index kann die Leistung verbessern, ohne einen weiteren Server hinzuzufügen. Eine Lesereplik einzurichten ist Konfigurationsarbeit. Partitionierung ändert eine Tabelle. Sharding, das Aufteilen der Daten auf getrennte Datenbankserver, verändert die umgebenden Abläufe: Migrationen, Sicherungen, Wiederherstellungen und Berichte müssen getrennte Shards berücksichtigen. Joins über Shard-Grenzen hinweg können in Anwendungscode wandern, und eine Abfrage ohne Kenntnis ihres Shard-Schlüssels muss möglicherweise alle Shards fragen.

Wie es in der Praxis aussieht

Eine PostgreSQL-Datenbank hinter einer mittelgroßen Anwendung enthielt Benutzerkonten, Mandanten, geografische Daten und täglichen Stromverbrauch. Sie lief bei geringem Verkehr, wurde mit wachsenden Datenmengen aber langsamer. Der Auftrag war, sie zehnmal schneller zu machen, ohne die Architektur umzubauen. Spielraum für künftiges Datenwachstum gehörte zum Ziel. Die Arbeit umfasste GiST-Indizes für Geodaten, zusammengesetzte Indizes auf Mandant und Datumsbereich, eine nach Zeit partitionierte Stromverbrauchstabelle, Hash-Partitionierung geografischer Daten, mit `EXPLAIN ANALYZE` geprüfte umgeschriebene Abfragen und Connection-Pooling. Die Antwortzeiten verbesserten sich um den Faktor zehn, der Ressourcenverbrauch der Datenbank sank um 40 %, und das System nahm wachsende Datenmengen ohne Verlangsamung auf. Der Bedarf an Hardwareaufrüstungen sank.

Bei einem mittelgroßen Unternehmen mit zehn PostgreSQL- und SQL-Server-Datenbankservern wurde die Infrastruktur für 40 % mehr Benutzerverkehr ohne Leistungseinbußen des Dienstes skaliert. Daneben verbesserten sich die Antwortzeiten der Abfragen durch Konfigurationsanpassung und Indexoptimierung um 30 %.

Die Reihenfolge der Arbeit beschreibt eine Notiz zum Optimieren vor dem Sharding: ansehen, straffen, Lesezugriffe vom Primärserver nehmen, die Tabellen um die Engstelle partitionieren und erst sharden, wenn eine Maschine

mit sinnvoller Arbeit ausgelastet ist. Die Notiz nennt drei begründete Fälle für Sharding: Der Primärserver ist durch bereits schlanke Anweisungen gesättigt, die Daten sind über ihre Wartbarkeit hinausgewachsen, oder die Trennung selbst ist gefordert.

Wo es schiefgeht

Mehr Webserver vor einer Datenbank. Jeder Anwendungsserver öffnet eigene Verbindungen. Sind um ein Vielfaches mehr Sitzungen aktiv, als die Datenbank Kerne hat, verbringt der Server seine Zeit mit dem Wechsel zwischen ihnen. Ein Verbindungspool löst das, bevor Hardware es tut, und ein weiterer Webserver macht es schlimmer.

Eine häufig umkämpfte Zeile. Schreibt jede Anfrage in denselben Datensatz, etwa einen Bestandszähler oder einen Kontostand, ist der Durchsatz ungefähr auf die von einer Sperre tragbaren Schreibvorgänge geteilt durch deren Haltedauer begrenzt, auf beliebig vielen Maschinen. Alles Langsame innerhalb dieser Sperre senkt die Obergrenze. Halten Sie die Transaktion kurz und externe Aufrufe außerhalb. Wächst ein Artikel über seine Maschine hinaus, teilen Sie seinen Bestand in kleinere Stücke, jedes mit genau einem Besitzer.

Zustand auf nur einem Server. Sitzungen im Speicher eines Prozesses, Uploads auf seinem lokalen Datenträger oder ein geplanter Job, der voraussetzt, dass keine weitere Kopie läuft, verhindern eine gleichwertige zweite Instanz. Halten Sie diesen Zustand in einem gemeinsamen Speicher, damit jede Instanz jede Anfrage beantworten kann.

Repliken werden wie Kopien des Primärservers behandelt. Eine Streaming-Replik liegt etwas zurück. Wer gerade etwas geschrieben hat und es dann von der Replik liest, sieht es deshalb womöglich nicht. Lesen Sie kürzlich geschriebene Daten vom Primärserver. Auch ein langer Bericht auf einem Standby kann abgebrochen werden, wenn er die dort eingespielten Änderungen aufhält. PostgreSQL dokumentiert dies als Abfragekonflikte samt Einstellungen, mit denen sich zwischen solchen Konflikten und Replikationsverzögerung abwägen lässt.

Abfragen lassen den Partitionsschlüssel weg. Partitionierung teilt eine Tabelle nach einem Schlüssel, etwa Monat oder Mandant. Eine Abfrage, die den Schlüssel nennt, liest nur die benötigten Partitionen. Ohne ihn liest sie jede Partition und gewinnt durch die Aufteilung nichts. Partitionieren Sie nach dem, worauf die

Abfragen bereits filtern. Erwarten Sie Nutzen nur bei großen Tabellen: Als Faustregel nennt das PostgreSQL-Handbuch eine Tabelle, die größer als der physische Speicher des Datenbankservers ist.

Weiterführende Lektüre

- RFC 9110: HTTP-Semantik, Abschnitt 3.3 — Verbindungen, Clients und Server sowie der zustandslose Entwurf, der Anfragen auf mehrere Server verteilen lässt.
- PostgreSQL: Tabellenpartitionierung — Was Partitionierung ist, wann sie sich lohnt und wie Partition Pruning nicht benötigte Partitionen überspringt.
- PostgreSQL: pg_stat_statements — Die Erweiterung, die Aufrufzahlen und Zeiten jeder normalisierten Anweisung speichert.
- PostgreSQL: Hot Standby — Nur lesende Abfragen auf einem Standby und Konflikte, die sie abbrechen können.
- RFC 9111: HTTP-Caching — Wie HTTP-Caches Antworten speichern und wiederverwenden, sodass wiederholte Anfragen nicht zum Ursprungsserver gelangen müssen.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Bereitstellungsweg, die Betriebsrisiken und die Bereitschaftsverantwortlichen. Wir können eine Änderung mit Prüfungen, einem Wiederherstellungsweg und klaren Zuständigkeiten abgrenzen.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Skalierbarkeit?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

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/glossary/scalability/