Engineer

Redis

Speicher im RAM für kurzlebige Daten

Was es ist

Redis ist ein Datenbankserver, der seinen gesamten Datenbestand als typisierte Strukturen im Arbeitsspeicher hält. Viele Anwendungsprozesse können sie über das Netzwerk mit jeweils einem kurzen Befehl lesen und verändern. Das Projekt bezeichnet sich als Datenstrukturserver. Die frei verfügbare Ausgabe heißt Redis Open Source. Redis Ltd verkauft daneben Redis Cloud und Redis Software unter kommerziellen Lizenzen.

Jeder Schlüssel hält einen Wert eines Typs. Zeichenketten sind einfache Bytes und dienen auch als Zähler. Hashes sind Datensätze aus Feldern und Werten, Listen bewahren die Ankunftsreihenfolge, Mengen enthalten eindeutige Mitglieder und sortierte Mengen ordnen ihre Mitglieder nach einem Zahlenwert. Die Typenliste reicht weiter zu Streams, JSON-Dokumenten, Geoindizes, Zeitreihen, Vektormengen und probabilistischen Strukturen wie Bloom-Filtern und HyperLogLog.

Befehle arbeiten direkt auf diesen Typen. `INCR` erhöht einen Zähler um eins und gibt den neuen Wert zurück, `SADD` fügt einer Menge Mitglieder hinzu und `SPOP` nimmt eines heraus. Jeder Schlüssel kann eine Ablauffrist erhalten, nach der Redis ihn selbst löscht. Redis nennt einen solchen Schlüssel flüchtig (volatile). Müssen mehrere Schritte als Einheit erfolgen, läuft ein Lua-Skript im Server. Redis garantiert dabei, dass währenddessen nichts anderes läuft.

Die Daten liegen im Arbeitsspeicher; der Datenträger ist optional. Redis kann Momentaufnahmen als RDB-Dateien schreiben oder jede Schreiboperation in einer nur durch Anhängen erweiterbaren Datei (AOF) protokollieren und die gespeicherten Schreiboperationen beim Start erneut ausführen, beides tun oder keines von beidem. Replikate kopieren einen Primärserver und können

Lesezugriffe bedienen. Eine Speicherobergrenze, `maxmemory`, und eine Verdrängungsrichtlinie entscheiden, was geschieht, wenn der Datenbestand den zugeteilten Platz übersteigt.

Mit Redis 8 änderte sich die Lizenz. Seit dieser Version wird Redis Open Source unter drei wählbaren Lizenzen angeboten: RSALv2, SSPLv1 und AGPLv3. Die ersten beiden sind Lizenzen mit einsehbarem Quellcode; AGPLv3 ist die Open-Source-Option.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Redis?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Die Abfragen und fehlenden Indizes hinter einem von Jahr zu Jahr schwerfälligeren System finden und die entscheidenden Probleme beheben.

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. 30 Tage Stromdaten in 180 ms — 180 ms für die Aggregation eines Monats an Zählerständen

30 Tage Stromdaten in 180 ms Kontakt aufnehmen →

Warum es wichtig ist

Eine Webanwendung läuft selten als einzelner Prozess. Hinter einem Lastverteiler stehen mehrere Instanzen mit jeweils eigenem Speicher. Alles darin — die Zahl der Anmeldeversuche, eine Sitzung, eine gecachte Seite — existiert nur in dem Prozess, der die Anfrage gerade bearbeitet hat. Ein so umgesetztes Limit lässt sich durch den Wechsel zu einer anderen Instanz umgehen. Redis stellt allen Instanzen für diesen Zustand einen gemeinsamen Speicher bereit, der sich schnell genug bei jeder Anfrage abfragen lässt.

Die atomaren Befehle zählen ebenso wie die Geschwindigkeit. Lesen zwei Instanzen jeweils einen Zähler, addieren eins und schreiben zurück, können beide dieselbe Zahl schreiben: Ein Inkrement geht verloren. `INCR` erledigt Lesen und Schreiben im Server als einen Schritt. Hundert gleichzeitige Anfragen erhöhen den Zähler deshalb um genau hundert. Ein Lua-Skript erweitert dieselbe Garantie auf eine kurze Folge: Eine Prüfung und Änderung, die nicht getrennt werden dürfen, etwa einen freien Platz zu prüfen und zu belegen, laufen als Einheit.

Der dritte Punkt sind die Ablauffristen. Ein Zeitfenster für die Ratenbegrenzung, eine Sitzung oder ein Einmalcode sollte nach Ablauf seiner Zeit verschwinden. Mit einer Ablauffrist am Schlüssel übernimmt der Speicher die Bereinigung statt

einer geplanten Aufgabe.

Der Preis ist Dauerhaftigkeit. Die Zahlen sollten vor der Entscheidung bekannt sein. Mit AOF und seiner Standardrichtlinie von einem fsync pro Sekunde kann ein Absturz ungefähr eine Sekunde an Schreiboperationen kosten. Bei alleinigen Momentaufnahmen, die üblicherweise alle paar Minuten entstehen, warnt die Dokumentation, man „sollte darauf vorbereitet sein, die letzten Minuten an Daten zu verlieren“. Replikation ist standardmäßig asynchron: Ein Primärserver kann eine Schreiboperation bestätigen und anhalten, bevor irgendein Replikat sie erhalten hat. Redis eignet sich daher gut für Zustand, der sich wiederherstellen lässt oder dessen Verlust keinen Schaden anrichtet, und schlecht für die einzige Kopie einer Bestellung, einer Zahlung oder eines Saldos. Die wichtigste Entwurfsfrage lautet, was die Anwendung tut, wenn Redis unerreichbar ist oder etwas ihm Anvertrautes vergessen hat.

Wie es in der Praxis aussieht

22 Ratenbegrenzer halten ihren Zustand in Redis für ein öffentliches Verzeichnis von Unternehmen und Fachleuten. Eine Grenze gilt dadurch über alle Instanzen hinweg, statt in einem Prozess zu liegen, wo sie leicht zu umgehen wäre. Jeder Begrenzer wird für seinen geschützten Pfad gebaut, weil Anmeldeversuch, Suche und Massenexport bei sehr unterschiedlichen Raten missbräuchlich werden. Entscheidend ist das Verhalten bei fehlendem Redis: Die Begrenzer schließen den Zugriff bei einem Ausfall, und Verkehr wird abgelehnt statt durchgelassen. Das hat einen sichtbaren Preis. Ein Redis-Problem wird für Nutzer sichtbar. Das wurde bewusst akzeptiert, weil andernfalls aus einem Redis-Problem ein Abrechnungsproblem würde.

Beim Verkauf begrenzter Bestände steht Redis als schnelle Zugangsschranke vor einer Datenbank: Ein von einem Lua-Skript verringerter Redis-Zähler lässt ungefähr so viele Käufer wie vorhandene Einheiten zu und weist alle anderen in Mikrosekunden ab. Das Skript macht Prüfung und Verringerung zu einem Schritt. Die Schranke verspricht weniger, als sie vermuten lässt. Redis repliziert asynchron. Nach einem Failover kann der neue Primärserver einige Zulassungen vergessen haben und ein paar Käufer zu viel durchlassen. Ein bedingtes `UPDATE` in der Datenbank entscheidet weiterhin über jeden Verkauf; ein Abgleichsprozess setzt die Schranke anhand des Datenbankwerts zurück. Bei wenigen, unterschiedlichen Einheiten wie nummerierten Sitzplätzen können ihre Kennungen in einer Redis-Menge liegen. `SPOP` teilt sie einzeln zu, ohne Konkurrenz um einen Zähler.

Die gewöhnlichste Rolle ist ein einfacher Cache. Eine in Bicep definierte Azure-Infrastruktur erzeugt jede Umgebung aus denselben Vorlagen, für Tests, Staging und den Produktivbetrieb gleichermaßen. Jede enthält Redis als Cache neben einem PostgreSQL Flexible Server, den Container Apps für API und Website sowie Front Door mit einer WAF-Richtlinie. Weil der Cache zusammen mit allem anderen in den Vorlagen steht, ist seine Änderung ein Pull Request wie jede andere Umgebungsänderung.

Wo es schiefgeht

Die meisten Redis-Probleme entstehen, wenn ein schneller, vergesslicher Speicher wie ein dauerhafter behandelt wird oder wenn angenommen wird, er erinnere sich an alle Zeitvorgaben.

Ein Schlüssel soll ablaufen und bleibt für immer. `SET` ersetzt den Wert und verwirft jede vorhandene Ablauffrist. Aktualisiert Code einen Cachewert mit einem einfachen `SET`, wird deshalb aus einem Eintrag für zehn Minuten ein dauerhafter:

```
> SET session:42 "v1" EX 600
OK
> SET session:42 "v2"
OK
> TTL session:42
(integer) -1
```

Geben Sie die Ablauffrist bei jeder Schreiboperation an, die eine benötigt, oder ergänzen Sie `KEEPTTL`, verfügbar seit 6.0.0, um die alte zu erhalten. Derselbe Fehler steckt in einem Zähler aus `INCR` und einem getrennten `EXPIRE`: Kommt der zweite Befehl nie an, läuft der Schlüssel nie ab. Senden Sie beide innerhalb eines Blocks aus `MULTI` und `EXEC` oder in einem Lua-Skript.

Schreibzugriffe melden Fehler, während Lesezugriffe weiter funktionieren. Der Datenbestand hat `maxmemory` erreicht. Die Verdrängungsrichtlinie ist `noeviction` oder eine `volatile-`-Richtlinie für Daten, bei denen kein Schlüssel eine Ablauffrist hat; das wirkt gleich. Wählen Sie die Richtlinie passend zur Aufgabe: `allkeys-` für einen reinen Cache, in dem jeder Schlüssel entfallen darf, und `volatile-` nur, wenn Cacheschlüssel eine Instanz mit Schlüsseln teilen, die bleiben müssen. Für diese Mischung empfiehlt die Dokumentation, zwei getrennte Instanzen zu erwägen.

Nach Neustart oder Failover fehlen von Redis bestätigte Schreiboperationen. Ein Absturz fiel in das fsync-Zeitfenster, oder ein Replikat wurde vor dem Empfang der Schreiboperation befördert. So ist Redis entworfen; die Korrektur liegt deshalb darin, was dort gespeichert wird. Speichern Sie alles, was überleben muss, in einer Datenbank mit dauerhaften Commits. Setzen Sie Redis als Schranke oder Cache davor und gleichen Sie beides von der Datenbank aus ab, wie es die Notiz zu Beständen tut.

Alle Clients stocken gleichzeitig. Ein langes Lua-Skript läuft, und Redis blockiert während seiner gesamten Laufzeit jede andere Serveraktivität. Halten Sie Skripte kurz und begrenzt und lassen Sie Schleifen über große Sammlungen weg.

Ein Begrenzer lässt bei einem Ausfall Verkehr durch. Der Clientcode deutet einen Redis-Fehler als „Grenze nicht erreicht“. Der Begrenzer lässt damit genau dann Anfragen durch, wenn sein Speicher unerreichbar ist. Entscheiden Sie den Fehlermodus bewusst und prüfen Sie ihn durch Anhalten von Redis: Ein Begrenzer für einen teuren Pfad sollte ablehnen, wie die oben beschriebenen 22 Begrenzer, auch wenn dadurch ein Redis-Problem für die Nutzer sichtbar wird.

Weiterführende Lektüre

- Mit Redis entwickeln — der Projekteinstieg: Befehle zum Ausprobieren, Clientbibliotheken je Sprache und die Datentypen.
- Redis-Persistenz — Momentaufnahmen, die nur durch Anhängen erweiterbare Datei, fsync-Richtlinien und mögliche Verluste je Verfahren.
- Redis-Replikation — wie Replikate einem Primärserver folgen und warum ein Zeitfenster für Datenverlust bleibt.
- Schlüsselverdrängung — `maxmemory`, sämtliche Verdrängungsrichtlinien und Hinweise zur Auswahl.
- EXPIRE — Ablauffristen für Schlüssel und die Befehle, die sie entfernen oder erhalten.
- Lua-Skripte — atomare Ausführung im Server und die währenddessen blockierte Aktivität.



Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

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 →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Redis?“ 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/redis/