Dokumentation Projekt Datenbanken

Lukas Maier

Sebastian Hoch

Rik Hopfensitz

Victoria Eisenhut

Öffentliches Github Repository:

https://github.com/hochs02/RedisAuction

Installation:

1. Terminal öffnen und in den Ordner /backend wechseln
2. Befehl ausführen: "npm i"
3. Befehl ausführen: "Docker Compose Up"
4. Terminal öffnen und in den Ordner /backend wechseln
5. Befehl ausführen: "npm start"
6. Terminal öffnen und in den Ordner/frontend wechseln
7. Befehl ausführen: "npm i"
8. Befehl ausführen: "npm start"

Redis

Die Wahl der Datenbank fiel speziell für dieses Projekt auf Redis. Redis ist eine In-Memory-Datenbank, die über eine Vielzahl von Vorteilen verfügt, die im Folgenden näher beleuchtet werden.

Redis ist nicht nur einfach einzurichten und zu verwenden, es erfordert zudem keine spezielle Konfiguration oder Software, um mit der Entwicklung zu starten. Darüber hinaus verfügt die NoSQL Datenbank über eine sehr hohe Leistung, sie ist extrem schnell und kann Millionen von Operationen pro Sekunde verarbeiten. Ein weiterer Vorteil ist die hohe Verfügbarkeit. Redis ist in der Lage, Daten über mehrere Knoten zu replizieren, was eben diese hohe Verfügbarkeit und zusätzlich auch eine hohe Fehlertoleranz ermöglicht. Zusätzlich gibt es den Aspekt der hohen Skalierbarkeit. Redis ist hoch skalierbar und kann problemlos über mehrere Knoten skaliert werden. Auch die Datenpersistenz wird unterstützt, was bedeutet, dass Daten auch nach einem Neustart des Servers gespeichert und abgerufen werden können.

Ein umfangreicher Anwendungsbereich wird durch die flexiblen Datentypen garantiert, es werden Datenstrukturen wie Hashes, Listen, “Sets” und “Sorted Sets” unterstützt. Ein letzter wichtiger Aspekt für das Entwickeln von neuen Projekten ist die Möglichkeit, Feedback und Erfahrung aus einer aktiven Gemeinschaft zu beziehen, was mit dem reichhaltigen Redis-Ökosystem ebenfalls gegeben ist.

Aus den genannten Vorteilen lässt sich schließen, dass Redis eine Vielzahl an Möglichkeiten für neue Projekte bietet, welche in unserem Fall zu sehr großen Teilen kompatibel mit der gegebenen Aufgabenstellung sind.

So ist für eine Auktionsseite besonders Schnelligkeit und Verfügbarkeit zentral. Gebote müssen nicht nur in Echtzeit empfangen, sondern auch angezeigt werden. Der Aspekt der Echtzeit ist einer der zentralsten Faktoren bei einer Auktionswebsite. Für deren Betrieb ist eine Echtzeitanzeige der Gebote essentiell, da eine Auktion nur so sinnvoll und rechtlich umgesetzt werden kann. Nur so wird ermöglicht, dass die Auktion fair und transparent bleibt, und gleichzeitig gewährleistet, dass die Bieter fundierte Entscheidungen darüber treffen können, wann sie bieten, wie viel sie

bieten und wann sie aufhören sollten zu bieten. Die Echtzeit-Anzeige versetzt die einzelnen Bieter unter Handlungsdruck, da diese sehen können, wie schnell das Bieten voranschreitet und wie viel Konkurrenz sie haben. Dabei gilt es, das Benutzererlebnis durch maximale Verfügbarkeit bestmöglich zu gestalten. Die Verfügbarkeit ist für eine Auktionswebsite wichtig, da sie es Käufern und Verkäufern garantiert, jederzeit und von jedem Ort aus auf die Website zuzugreifen. Dies fördert sowohl die Beteiligung als auch das Engagement. Außerdem können die Teilnehmer so die Vorteile des kontinuierlichen Bietprozesses nutzen, was zu höheren Preisen und größeren Gewinnen führen kann. Schließlich können die Veranstalter durch die Verfügbarkeit ein größeres Publikum erreichen, was die Sichtbarkeit und den Bekanntheitsgrad ihrer Auktion erhöhen kann. Hinzu kommt auch die hohe Leistung. Da Redis auch in dieser Hinsicht überdurchschnittlich gut abschneidet, wird dem Nutzer so ein angenehmes Nutzererlebnis ermöglicht. Insbesondere soll durch das Vermeiden von schlechter Leistung präventiv gegen zu langsame Ladezeiten, Verzögerungen oder sogar Offline-Phasen vorgegangen werden. Um eben solche negativen Effekte zu vermeiden, lässt sich schließen, dass eine gute Leistung eine weitere Grundvoraussetzung für die Auktionswebsite darstellt. Jedoch gibt es zwei Nachteile, die es bei Redis zu beachten gibt. Da es sich hierbei um eine In-Memory-Datenbank handelt, werden die Daten lediglich im Arbeitsspeicher gespeichert, was bedeutet, dass die Menge der Daten, die es speichern kann, durch die Menge an Primary-Storage des Server begrenzt ist. Aus diesem Grund wird Redis im Allgemeinen für Anwendungen verwendet, die eine hohe Leistung erfordern und relativ kleine Datensätze haben. Bei der Auktionswebsite werden dabei nur die Höchstgebote der einzelnen Nutzer zu den verschiedenen Objekten gespeichert.

In unserem Fall ist dieser Nachteil jedoch vernachlässigbar, da weitere benötigte Daten bei Bedarf in anderen DBMS gespeichert werden können. Der Fokus liegt auf der Leistung und die Datensätze sind limitiert, weshalb eine In-Memory-Datenbank der richtige Schritt für unser Projekt ist.

Sorted Sets und Key-Value

In Redis ist ein Sorted Set eine Datenstruktur, die ähnlich wie ein reguläres Set funktioniert, jedoch zusätzlich zu jedem Element Scores bzw. eine Gewichtung enthält. Sorted Sets werden durch eine Kombination aus einer Hash-Tabelle und einer Skip-Liste implementiert, was schnelle Einfüge-, Lösch- und Durchlauf-Operationen ermöglicht.

In einem Sorted Set ist jeder Eintrag sowohl mit einem Value als auch mit einem Score versehen. Der Value ist das eigentliche Element, das im Set gespeichert wird, während der Score eine numerische Bewertung ist, die diesem Element zugeordnet wird.

Die Score-Werte dienen dazu, die Reihenfolge der Elemente im Set festzulegen. Elemente mit höheren Scores werden im Set weiter vorne platziert als Elemente mit niedrigeren Scores. Wenn mehrere Elemente denselben Score haben, wird die lexikographische Ordnung der Value-Werte verwendet, um die Reihenfolge der Elemente festzulegen.

Das Hinzufügen von Elementen zu einem Sorted Set erfordert immer eine Kombination aus einem Value und einem Score. Wenn ein Element mit einem bereits vorhandenen Value hinzugefügt wird, kann der Score aktualisiert werden, um die Position des Elements im Set zu ändern. In ähnlicher Weise können Score-Updates auch dazu verwendet werden, die Reihenfolge der Elemente im Set zu ändern.

In unserer Anwendung sind die Values mit einem Nutzer (Bieter) und die Scores mit dem jeweiligen Gebot versehen. Der höchste Score führt hierbei die Reihenfolge an. Dadurch muss für das Höchstgebot immer nur die erste Position des Sorted Sets ausgegeben werden.

Socket

Socket.IO ist eine JavaScript-Bibliothek, welche die Erstellung von Echtzeit-Webanwendungen erleichtert. Sie ermöglicht die bidirektionale

Kommunikation zwischen Webbrowsern und Servern über WebSockets, einer Technologie, die es einem Client und einem Server ermöglicht, eine dauerhafte Verbindung aufrechtzuerhalten. Wenn WebSockets nicht verfügbar sind, bietet Socket.IO einen Fallback-Mechanismus, der automatisch auf alternative Übertragungsmechanismen wie AJAX oder JSONP umschaltet.

Beim Projekt wurde Socket.IO verwendet, um Ereignisse zu senden und zu empfangen, die in Echtzeit zwischen Clients und Server ausgetauscht werden. Diese Ereignisse können Daten in einer Vielzahl von Formaten enthalten, einschließlich JSON und Binärdaten. Socket.IO unterstützt auch mehrere Räume oder Kanäle, was die Verwaltung mehrerer Untergruppen innerhalb einer Anwendung vereinfacht.

Diese Übertragung in Echtzeit ist für unsere Anwendung sehr wichtig, da nur so eine reibungslose und faire Versteigerung ablaufen kann. Bieter müssen in Echtzeit sehen, was das momentane Gebot ist, um noch rechtzeitig vor Ablauf der Zeit reagieren zu können. In unserem Fall waren es somit die Gebote, welche in Echtzeit zwischen Client und Server ausgetauscht werden müssen. Die Übertragung in Echtzeit war für das Projekt die elementare Eigenschaft von Socket.IO, jedoch bringt diese Bibliothek noch weitere Vorteile mit sich wie beispielsweise:

1. Socket.IO unterstützt sowohl Server- als auch Client-gerichtete Kommunikation, so dass beide Seiten Daten an die jeweils andere Seite senden können (Bidirektionale Kommunikation).

2. Wenn WebSockets nicht verfügbar sind, bietet Socket.IO einen Fallback-Mechanismus, der automatisch zu alternativen Kommunikationsmodi wie AJAX und JSONP umschaltet. Dies erhöht die Robustheit der Anwendung.

3. Socket.IO unterstützt die Skalierung von Anwendungen, indem es zahlreiche Server einsetzt, die miteinander kommunizieren und die Last gleichmäßig verteilen können.

4. Socket.IO bietet eine breite Palette von Funktionen und Konfigurationsoptionen, die es Programmierern ermöglichen, die Bibliothek flexibel an ihre eigenen Bedürfnisse anzupassen.

NoSQL vs. SQL

NoSQL-Datenbanken sind nicht-relationale Datenbanken, die für die Speicherung und den Abruf von unstrukturierten Daten konzipiert sind. Die Daten werden dabei in einem Key-Value-Format gespeichert. Sie werden in der Regel für große Datensätze verwendet, die Skalierbarkeit, Flexibilität und Leistung erfordern. NoSQL-Datenbanken sind in der Regel einfacher zu skalieren und funktionieren besser in Kombination mit verteilten Systemen. Außerdem bieten sie eine Vielzahl von Funktionen wie hohe Verfügbarkeit, dynamische Schemata und eine gute Leistung.

SQL-Datenbanken sind relationale Datenbanken, die für das Speichern und Abrufen strukturierter Daten konzipiert sind. Anders als bei den NoSQL Datenbanken werden hier die Daten in Tabellen, das heißt Spalten und Zeilen gespeichert. Sie werden in der Regel für kleine Datenmengen verwendet. SQL-Datenbanken sind in der Regel strengeren Vorgaben unterworfen und erfordern mehr Einrichtungszeit, bieten aber ebenfalls unterschiedliche Funktionen wie ACID-Konformität und die Möglichkeit, SQL-Abfragen zur Interaktion mit der Datenbank zu verwenden. SQL-Datenbanken eignen sich dementsprechend besser für komplexe Abfragen und Transaktionen.

Eine NoSQL-Datenbank ist für eine Auktionswebsite wesentlich besser geeignet als eine SQL-Datenbank, da sie große Datenmengen effizienter verwalten kann, agiler und flexibler auf sich ändernde Anforderungen reagiert und große Mengen unstrukturierter Daten verarbeiten kann. Außerdem bieten NoSQL-Datenbanken in der Regel eine bessere Skalierbarkeit und Leistung als relationale Datenbanken, d. h. sie können viele gleichzeitig zugreifende Nutzer und daraus resultierende Transaktionen verarbeiten. Auch sind NoSQL-Datenbanken oft kostengünstiger als herkömmliche relationale Datenbanken. Aus diesen Gründen ist für den Bietvorgang beim Auktionsprojekt die Wahl einer NoSQL-Datenbank am sinnvollsten.