Transferarbeit Chat Applikation

Von René Striby

Für das Fach Parallele und verteilte Systeme

Das Github Repository ist unter folgendem Link verfügbar:

https://github.com/rene8619/reneschat

Das Docker Image ist auf Dockerhub verfügbar rstriby/reneschat

https://hub.docker.com/repository/docker/rstriby/reneschat

Der Chat ist unter folgendem Link erreichbar:

http://striby.teko.hackerman.ch/

Ausfälle von Teilsystemen

Client:

Der Client ist eine HTML-Seite, die im Benutzerbrowser aufgerufen wird. Der Webserver ist mit Node.js und Express erstellt und befindet sich zusammen mit dem Backend in einem Dockercontainer. Dieser Container wird in einem Kubernetes-Cluster betrieben und ist so konfiguriert, dass immer zwei Replikationen (Pods) ausgeführt werden. Dies bedeutet, dass der Chat auch dann weiterhin über den anderen Webserver erreichbar ist, wenn einer der beiden ausfällt. Die Steuerung erfolgt durch den integrierten Load Balancer im Kubernetes-Cluster. Bei einem Ausfall startet Kubernetes automatisch einen neuen Pod, um die Verfügbarkeit sicherzustellen. Der Kubernetes-Cluster besteht aus drei Nodes, wobei Kubernetes sicherstellt, dass die beiden Pods auf unterschiedlichen Nodes laufen. Somit bleibt der Betrieb auch bei einem Ausfall eines Nodes gewährleistet. Wenn der Client im Browser abstürzt, verliert der Benutzer vorübergehend die Verbindung. Er kann sich jedoch jederzeit neu verbinden. Der Ausfall eines Benutzers hat keine Auswirkungen auf den Rest des Systems. In der Zwischenzeit von anderen Benutzern gesendete Nachrichten werden dem Benutzer angezeigt, sobald er sich erneut verbindet, da alle alten Nachrichten beim erneuten Verbinden angezeigt werden. Um die Ausfallsicherheit weiter zu erhöhen, könnten zusätzliche Replikationen des Containers verwendet werden.

Backend:

Das Backend befindet sich ebenfalls in einem Container mit zwei Replikationen und verfügt somit über die gleiche Ausfallsicherheit wie der Client-Teil. Wenn ein Backend ausfällt, verlieren alle Benutzer, die mit diesem Backend verbunden sind, vorübergehend die Verbindung. Durch das Neuladen der Seite werden die Benutzer automatisch mit einem verfügbaren Backend verbunden. Der Ausfall eines Backends hat keine Auswirkungen auf Benutzer, die mit einem anderen Backend verbunden sind. Die Nachrichten gehen nicht verloren, da das Backend den Nachrichtenverlauf in der Redis-Datenbank speichert. Wenn ein neues Backend gestartet wird, lädt es den Nachrichtenverlauf aus der Datenbank. Für den Benutzer spielt es keine Rolle, mit welchem Backend er verbunden ist, da die Nachrichten und Benutzerlisten zwischen den Backends über die Redis-Datenbank synchronisiert werden. Das Backend ist so konzipiert, dass es horizontal skalierbar ist. Um die Ausfallsicherheit weiter zu erhöhen, könnten zusätzliche Replikationen des Containers verwendet werden.

Redis-Datenbank:

Die Redis-Datenbank speichert den Nachrichtenverlauf und die Benutzerlisten und fungiert als Message Broker, um Nachrichten an alle Backends zu senden. Ohne Redis funktioniert die Synchronisierung der Backends nicht. Theoretisch könnte der Chat so gestaltet werden, dass er auch ohne Redis für die direkt verbundenen Benutzer weiter funktioniert. Wie der Chat jedoch derzeit programmiert ist, würde ein Ausfall von Redis den gesamten Chat zum Stillstand bringen. Um die Ausfallsicherheit zu erhöhen, könnte man Redis ebenfalls horizontal skalieren. Da Redis jedoch kein zustandsloses System ist, müsste zusätzlich eine Synchronisierung der Redis-Datenbanken implementiert werden. Es wäre auch möglich, den Chat so zu gestalten, dass er für die direkt verbundenen Benutzer weiterläuft und alle Daten nachträglich synchronisiert, sobald Redis wieder verfügbar ist.

Beim Horizontal skalieren der einzelnen Teile sollte man das gesamte System im Auge behalten. Es nützt wenig, zum Beispiel vom Backend 10 Container laufen zu lassen und dann beim Redis einen Flaschenhals oder Single Points of Failure zu haben.

Testen des Clients

Testfälle

Testfall Nr.	Was machen	Erwartetes Ergebnis
1	Den Chat unter der Adresse http://striby.teko.hackerman.ch/ öffnen.	Die Chat-Oberfläche wird vollständig geladen, und es sind keine Fehler sichtbar.
2	Den Chat mit einem anderen Browser öffnen.	Die Chat-Oberfläche wird vollständig geladen, und es sind keine Fehler sichtbar.
3	Kontrollieren, ob ein zufälliger Benutzername vergeben wurde.	Hinter aktueller Name steht Gastxxxxx
4	Den Benutzernamen ändern.	Hinter aktueller Name steht der neue Benutzername
5	Eine Chatnachricht eingeben und auf Senden drücken.	Die Nachricht taucht im Chatfenster nach folgendem Schema auf: Benutzername: Nachricht
6	Das Browserfenster mit dem Chat schliessen. Danach den Chat erneut öffnen.	Die alten Chatnachrichten werden wieder angezeigt.
7	Den Chat in mindestens 4 Browserfenstern/Tabs öffnen	Bei allen Fenstern ist der Chat mit allen Nachrichten sichtbar.
8	Ein einem Browserfenster eine Nachricht Senden.	Die Nachricht wird sofort in den anderen geöffneten Chats angezeigt.
9	Den Benutzernamen in einem der geöffneten Chatfenster ändern.	Der Benutzername des entsprechenden Benutzers wird auch in der Liste eingeloggten Benutzer der anderen geöffneten Chats aktualisiert
10	In einem Browserfenster den Chat schliessen.	Der entsprechende Benutzername versschwindet aus der Liste Eingeloggte Benutzer bei dennoch geöffneten Chats.
11	In einem Browserfenster den Chat verlassen, indem man eine andere Website öffnet.	Der entsprechende Benutzername versschwindet aus der Liste Eingeloggte Benutzer bei dennoch geöffneten Chats.

Testresultate

Testdatum 22.09.2023

Testfall Nr.	Test erfolgreich ✓/X	Bemerkungen
1	✓	Firefox verwendet
2	✓	Chrome verwendet
3	✓	
4	✓	
5	✓	
6	✓	
7	✓	
8	✓	
9	✓	
10	✓	
11	✓	