Aufgabe 1 Modelierung

Das XmasWishes-Projekt hat zum Ziel, weltweit Weihnachtswünsche von Kindern zu sammeln und zu verarbeiten. Dabei wird angenommen, dass der Weihnachtsmann für jedes Kind einen Wunsch entgegennimmt. Insgesamt gibt es weltweit 2,2 Milliarden Kinder unter 12 Jahren, wobei nicht alle Weihnachten feiern – diese Schätzung geht jedoch von allen Kindern aus. Die Wünsche werden über eine API an den Weihnachtsmann gesendet und müssen über das letzte Quartal des Jahres verarbeitet werden. Bei einer Normalverteilung wird die maximale Last auf ca. 570 Anfragen pro Sekunde geschätzt.

Architekturentwurf: Das System setzt auf eine Microservice-Architektur, bei der jeder Arbeitsschritt durch einen eigenen Microservice abgewickelt wird. Dies ermöglicht eine einfache Wartung und skalierbare Erweiterung. Für die Nachrichtenverarbeitung wird eine Message Queue verwendet, die als zentrales Transportmittel dient und Load Balancing ermöglicht.

Komponenten:

- Message Queue zur Verteilung von Nachrichten.
- Datenbank zur Speicherung der Wünsche.
- Microservices zur Verarbeitung der Wünsche und Statusänderungen.
- Microservices als API Gateway

Skalierung und Ausfallsicherheit: Das System setzt auf horizontale Skalierung, wobei bei steigender Last zusätzliche Instanzen hinzugefügt werden können, um die Last zu bewältigen und Ausfallsicherheit zu garantieren.

Technische Herausforderungen: Das System muss rund 2 Milliarden Anfragen in drei Monaten empfangen, was zu einer durchschnittlichen Last von 280 Anfragen pro Sekunde führt. Das Gateway muss in der Lage sein, Spitzenlasten von bis zu 570 Anfragen pro Sekunde zu verarbeiten.

Der große Vorteil an der Microservice Architektur, dass sie Leicht skalierbar ist. Es kommt lediglich ein Kleiner Orchestrierungsaufwand hinzu, wenn zusätzliche Server genutzt werden.

Das ermöglicht es dem Weihnachtsmann für die verschiedenen Kontinente oder Länder unterschiedlich viele Server zur Verfügung zu stellen und diese an die Last anzupassen. Auch kann er einen Server bis zum letzten Quartal laufen lassen, um Wünsche über das Jahr bereits in der Datenbank zu sammeln.

Aufgabe 2 Konkretisierung

Für die Umsetzung des XmasWishes-Projekts werden folgende Haupttechnologien eingesetzt:

- Microservices: Jeder Microservice wird in einem Docker-Container ausgeführt, was eine einfache Bereitstellung und Skalierung ermöglicht.
- Messaging System: Apache Kafka wird als skalierbares Messaging-System genutzt, um eine zuverlässige Kommunikation zwischen den Microservices zu gewährleisten.
- Datenbank: MongoDB, eine NoSQL-Datenbank, wird verwendet, um große Mengen unstrukturierter Daten (Wünsche) effizient zu speichern.

Aufgabe 3 Prototyp

Für den Prototypen habe ich die interne Verarbeiteungspipeline als Microservices in Docker implementiert. Die Microservices erwarten eine JSON und verändern ediglich den Status, um zu zeigen, dass das Geschenk einen Schritt weiter ist.

Kafka sorgt für die Kommunikation zwischen den Microservices. Es existieren 4 Topics, die mit dem Status der Wünsche korrelieren. Die Microservices sind jeweils zu dem Topic subscribed für das sie zuständig sind. Beim Hochfahren eines Microservices werden diese auch direkt einer Consumer-Group hinzugefügt, damit sich die Last gleichmäßig auf die Container verteilen kann.

Für den Prototypen habe ich auf die Datenbank verzichtet, weil ich vor allem den API Gateway Microservice testen möchte. Dieser nimmt die Anfragen an und schickt sie direkt in die interne Pipeline an das erste Topic. In einem fertigen System würde die API den Wunsch in der Datenbank speichern und die Wünsche würden in der Nacht vor Weihnachten als Batch verarbeitet werden. Die maximale Verarbeitungsgeschwindigkeit der internen Pipeline würde eigene Lasttests rechtfertigen.

Zum Testen meines API Gateways habe ich Jmeter benutzt. Jmeter erlaubt es mir eine hohe Anzahl an Threads zu erstellen und per API Wünsche an mein System zu senden. Um die Tests zu vereinfachen verschicke ich den wiederholt den gleichen Wunsch.
Für die Tests werde ich die Anzahl an Threads stetig erhöhen und den Durchsatz messen. Jeder Test hat einen Ramp Up von 30 Sekunden und läuft 5 Minuten.

Test 1: 10 Threads Durchsatz: 261/s

Test 2: 50 Threads Durchsatz: 675/s

Test 3: 75 Threads Durchsatz: 728/s

Test 4: 100 Threads Durchsatz: 749/s

Test 5: 125 Threads Durchsatz: 765/s

Test 6: 200 Threads Durchsatz: 764/s

Von 125 zu 200 Threads gab es keine Erhöhung mehr. Der Arbeitsspeicher meines Laptops kam auch an seine Grenzen. Für meine Maschine bedeutet das, dass ca. 765 Anfragen/s das Maximum sind, das ich mit den Jmeter Tests ermitteln konnte.

Alle Tests liefen mit einer Errorquote von 0,1% durch. Das müssen wohl die Kinder sein die an Weihnachten nur Kohle kriegen 😊.