OMÜ, BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ, VERİTABANI LABORATUVARI, 6.DENEY HAZIRLIĞI
2024, BAHAR

Öğr_Numarası, İsim	20060355, Aleyna KAHRAMAN

Deney Sorusu 1' i cevaplandırınız. (En fazla 150 kelime)

Bu senaryoda, geçici veri saklama ihtiyacını çözmek için bellekte tutulan listeler yerine, kalıcı veri saklama çözümleri gereklidir. Özellikle, ilişkisel veya belge tabanlı veri tabanları gibi kalıcı depolama mekanizmaları tercih edilebilir. Bu veritabanları, uygulama durdurulduğunda bile verilerin korunmasını sağlar, böylece yeni bir versiyona geçiş veya uygulamanın yeniden başlatılması durumunda veriler kaybolmaz. Veritabanlarının ACID özellikleri, veri bütünlüğünü ve tutarlılığını garanti eder. Ayrıca, yedekleme ve geri alma özellikleri sayesinde veri kaybı riski minimize edilir. Bu yaklaşım, uygulamanın performansını artırırken aynı zamanda veri yönetimini daha etkili hale getirir. Örneğin, ilişkisel veri tabanlarından MySQL veya PostgreSQL, NoSQL çözümlerinden MongoDB veya Cassandra gibi seçenekler kullanılabilir. Bu sayede, onbinlerce SMS gönderimi gibi yoğun iş yükleriyle başa çıkılabilir ve veri kaybı riski minimize edilir. Veritabanı kullanımı, her yeni versiyonla birlikte kaybolan ve yüksek erişim gereksinimleri olan verinin kalıcı olarak saklanmasını sağlar, böylece hizmetin kesintisiz olarak sağlanmasına yardımcı olur.

Deney Sorusu 2' yi cevaplandırınız. (En fazla 500 kelime)

IoT cihazlarından gelen büyük hacimli veriyi işlemek ve depolamak için ölçeklenebilir ve yüksek erişilebilir bir çözüm gerekmektedir. Bu bağlamda, NoSQL veritabanları, özellikle de MongoDB gibi doküman tabanlı veritabanları ideal bir seçenektir. MongoDB, esnek şema yapısı sayesinde değişen ve genişleyen veri yapılarına uygun olarak veri depolama imkanı sunar.

Veritabanı Teknolojisi

MongoDB: 50.000 cihazdan gelen verilerin karşılanabilmesi için ölçeklenebilir bir yapı gereklidir ve MongoDB, dağıtık mimarisi ile bu gereksinimi karşılayabilir. MongoDB, veri dokümanlarını esnek bir şekilde saklar ve yüksek hacimli veri işlemede oldukça etkilidir. Her 5 dakikada bir gelen verileri kaydetmek için, MongoDB'de her cihaz için bir doküman oluşturulabilir. Bu durumda, 50.000 cihaz için 50.000 doküman oluşturulur ve her doküman yaklaşık 10 MB yer kaplar.

Veri İletimi

MQTT Protokolü: Verilerin toplu olarak web servisine iletilmesi için MQTT protokolü kullanılır. MQTT, yayın/abone modeliyle çalışır ve IoT uygulamaları için optimize edilmiştir. Bu protokol, düşük bant genişliği ve düşük güç tüketimi ile veri iletimini sağlar, bu da IoT cihazları için idealdir.

Yük Dengeleyici ve Ölçeklenebilirlik

Yük Dengeleyici: 50.000 cihazdan gelen istekleri işlemek için yük dengeleyici kullanımı önemlidir. Yük dengeleyici, gelen istekleri farklı sunuculara dağıtarak yükü dengeler ve hizmetin sürekliliğini sağlar. AWS Elastic Load Balancer (ELB) veya NGINX gibi çözümler kullanılabilir. Yük dengeleyici, veritabanı sunucuları ve veri işleme katmanları arasında yükü dengeler.

Sunucu Yapısı ve Sayısı

Kafka Broker'ları: IoT cihazlarından gelen veriyi almak ve işlemek için Apache Kafka kullanılabilir. Kafka, büyük veri akışlarını yönetmek için idealdir. 50.000 cihazdan gelen veri akışını karşılamak için en az 5 Kafka broker önerilir. Bu brokerlar, veriyi alıp tüketicilere iletmekle sorumludur.

Kafka Consumers: Kafka'daki mesajları tüketip MongoDB'ye kaydeden tüketiciler için 10-15 sunucu yeterli olabilir. Bu tüketiciler, veriyi alıp işleyerek MongoDB'ye yazarlar. Tüketici sayısı, veri işleme kapasitesine göre ayarlanabilir.

MongoDB Cluster: Verinin kalıcı olarak saklanması için MongoDB cluster yapısı kullanılır. Başlangıç olarak, her biri bir shard olacak şekilde en az 3 MongoDB sunucusu önerilir. Bu yapı, veriyi yatayda bölerek ölçeklenebilirliği sağlar. Shard sayısı, veri hacmine ve sorgu performansına göre artırılabilir.

Yedekleme ve Erişilebilirlik

Replication: MongoDB'nin replika setleri kullanılarak verinin yedeklenmesi ve yüksek erişilebilirlik sağlanabilir. Replika setleri, verilerin birden fazla kopyasını tutarak veri kaybını önler ve hizmetin devamlılığını sağlar.

Fault Tolerance: Kafka ve MongoDB'nin hata toleransı mekanizmaları, veri kaybını minimuma indirir ve sistem bileşenlerinde oluşabilecek hatalara karşı dayanıklılığı artırır.

Ölçeklenebilirlik ve Yönetim

Kubernetes: Mikroservis mimarisiyle tasarlanan servislerin yönetimi için Kubernetes gibi konteyner orkestrasyon sistemleri kullanılabilir. Kubernetes, otomatik ölçeklenebilirlik ve dağıtım kolaylığı sunar. Artan veri hacmi ve kullanıcı sayısına göre sistem kapasitesi kolayca artırılabilir.

Sonuç

Bu mimari, 50.000 cihazdan gelen verileri etkin bir şekilde işleyip depolayabilir. Apache Kafka, verilerin anlık işlenmesini sağlarken, MongoDB verilerin kalıcı olarak saklanmasını sağlar. Yük dengeleyici, veri işleme ve depolama katmanlarının dengeli çalışmasını sağlar. Bu yapı, gelecekteki büyüme ve ölçeklenebilirlik ihtiyaçlarını karşılayabilir ve Türkiye'nin ormanlık bölgelerine yerleştirilen IoT cihazlarından gelen verilerin güvenli, hızlı ve verimli bir şekilde işlenmesini ve saklanmasını sağlar.