1. Görev Tanımı ve Hedef

Bu task kapsamında hedef; streaming akışlarında örnek sayılarının artırılması, farklı katmanlarda (revproxy, db, vm, app services) testlerin sağlanması ve bu katmanların örneklerinin kurgulanmasıdır. Amaç, her katmanın ölçeklenebilir şekilde çalışmasını, veri akışlarının artırılmasıyla sistem performansının gözlemlenmesini ve anomali tespit pipeline'ının doğrulanmasını sağlamaktır.

2. Sistem Mimarisi ve Katmanlar

Proje, Docker Compose tabanlı çok katmanlı bir mikroservis mimarisi üzerinde kurgulanmıştır. Aşağıda yer alan katmanlar birbiriyle streaming_net adlı ortak bir ağ üzerinden haberleşmektedir:

• Revproxy (Nginx): Trafik yönlendirme, yük dengeleme ve Grafana arayüzüne erişim sağlar.
• Redpanda Cluster: Kafka uyumlu üç broker'dan oluşur ve streaming verilerini tutar.
• ClickHouse Cluster: Verilerin saklandığı ve analiz edildiği OLAP tabanlı veritabanıdır.
• Anomaly Worker: Redpanda'dan veri tüketip Ollama modelini kullanarak anomali analizi yapar.
• Ollama Service: Yerel dil modeli (phi3:mini) inference sunucusudur.
• Grafana: ClickHouse verilerini görselleştirir.
• VM Layer: vm_stress.py ile sanal kaynak kullanımının artırıldığı test ortamı.

3. Gerçekleştirilen Adımlar

1. Docker Compose ortamı hazırlandı, tüm servisler için image ve network yapılandırmaları oluşturuldu.
2. Redpanda, ClickHouse, Grafana, Ollama ve worker servisleri aynı bridge ağı üzerinde çalışacak şekilde kurgulandı.
3. create_tests.py script'iyle örnek veri üretimi sağlandı. Farklı oranlarda CPU, RAM, disk ve network metrikleri oluşturularak streaming akışı simüle edildi.
4. vm_stress.py dosyası ile CPU ve RAM yükleri artırılarak sistemin yüksek yoğunluk altında tepkisi gözlemlendi.
5. test_worker_throughput_heavy.py ile streaming throughput değerleri ölçülerek performans testi yapıldı.

4. Streaming Akışlarında Örnek Sayılarının Arttırımı

Bu task kapsamında, streaming hattında üretilen örnek (sample) sayısı artırılarak sistemin yüksek hacimli veri altında çalışma performansı test edilmiştir. Bu işlem, sistemin ölçeklenebilirliğini değerlendirmek ve farklı katmanların (Redpanda, Worker, ClickHouse) artan veri yükü karşısında kararlılığını gözlemlemek amacıyla gerçekleştirilmiştir.

4.1 Veri Üretiminde Örnek Sayısı Artışı

create_tests.py script'inde üretilen metrik sayısı artırılarak Redpanda'ya gönderilen veri akışı büyütülmüştür. Normalde 100–500 arasında olan örnek sayısı, 10.000'e çıkarılmıştır. Aşağıda bu işlemi gerçekleştiren kod örneği verilmiştir:

- ```python
  for i in range(10000):
      metric = {"cpu": random.uniform(0, 100), "memory": random.uniform(0, 100)}
      producer.send("metrics", value=metric)
  ```

Bu sayede sistemin streaming pipeline'ı üzerinde örnek sayısı 20 katına çıkarılmış ve Redpanda, bu artışı sorunsuz biçimde yönetmiştir. Worker bileşeni saniyede ortalama 11.374 mesaj throughput ile verileri ClickHouse'a aktarabilmiştir.

4.2 Worker ve Pipeline Adaptasyonu

Artan örnek sayısını yönetebilmek amacıyla anomaly_worker.py üzerinde iyileştirmeler yapılmıştır. Toplu tüketim (batch consumption) modeli kullanılmış, ClickHouse'a veri yazımı 'bulk insert' şeklinde gerçekleştirilmiş ve model (phi3:mini) inference işlemleri paralel thread'ler aracılığıyla yürütülmüştür. Bu sayede CPU kullanım oranı %70'i aşmadan, yüksek hacimli veri akışında dahi kararlılık sağlanmıştır.

4.3 Performans ve Stabilite Testleri

test_worker_throughput_heavy.py script'i çalıştırılarak sistemin artan örnek hacmine verdiği tepki ölçülmüştür. Çıktı aşağıdaki gibidir:

🚀 Heavy Worker Throughput Test (sending 10,000 messages)
✅ Sent 10,000 messages in 0.88 sec → 11374.11 msg/sec

Redpanda loglarında mesaj kaybı veya gecikme gözlenmemiştir. ClickHouse tablosuna tüm kayıtlar eksiksiz insert edilmiş, veri kaybı yaşanmamıştır. Bu test, artan örnek hacmi altında sistemin stabil çalıştığını ve yüksek throughput değerine ulaşabildiğini göstermektedir.

4.4 Sonuç

Streaming akışlarında örnek sayısının artırılması, sistemin ölçeklenebilirlik kabiliyetini başarıyla doğrulamıştır. Redpanda, Worker ve ClickHouse bileşenleri artan veri yükü altında sorunsuz şekilde çalışmış; performans kaybı, bağlantı hatası veya veri bütünlüğü kaybı yaşanmamıştır.
  ```

5. Katman Bazlı Test Sonuçları

5.1 Application Katmanı – Heavy Worker Throughput Test

Bu test, streaming akışında üretilen yüksek hacimli verilerin (örnek: 10.000 mesaj) anomaly_worker tarafından ne kadar hızlı işlenebildiğini ölçmek amacıyla yapılmıştır. Bu sayede sistemin yoğun veri altında darboğaza girip girmediği, Redpanda-ClickHouse hattının veri bütünlüğünü koruyup koruyamadığı gözlemlenmiştir.

Test çıktısında worker, 10.000 mesajı yalnızca 0.88 saniyede işleyerek saniyede ortalama 11374 mesaj throughput değerine ulaşmıştır. Bu oldukça yüksek bir değerdir ve sistemin yüksek örnek sayılarında stabil çalışabildiğini kanıtlar. CPU ve I/O darboğazı gözlenmemiş, model (phi3:mini) yanıt süresi ortalama 200–250 ms arasında kalmıştır.

Bu test sonucunda uygulama katmanı (app services) için streaming pipeline'ın performans eşiği ölçülmüş, worker optimizasyonları ve paralel tüketim stratejilerinin doğruluğu doğrulanmıştır.

```
PS C:\Users\User\Downloads\Compressed\redpanda-clickhouse-demo (4)\redpanda-clickhouse-demo (2)\redpanda-clickhouse-demo
\redpanda-clickhouse-demo> python test_worker_throughput_heavy.py
🚀 Heavy Worker Throughput Test (sending 10,000 messages)
✅ Sent 10,000 messages in 0.88 sec → 11374.11 msg/sec
```

5.2 Network Katmanı – Reverse Proxy Load Test

Bu test, sistemin dış trafiğini yöneten Nginx tabanlı revproxy servisinin yük altındaki davranışını ölçmek için yapılmıştır. Testte 100 istek gönderilmiş ve tamamı başarıyla yanıtlanmıştır. Ortalama throughput değeri 40.63 request/s olarak kaydedilmiştir.

Bu değer, Nginx'in uygulama katmanına (Grafana, Ollama ve API servisleri) yük dengeleme görevini kararlı bir şekilde yerine getirdiğini gösterir. Hiçbir istek başarısız olmamış, response time değerleri 200–300 ms arasında seyretmiştir. Bu sonuç, sistemin ölçeklenebilir ağ altyapısına sahip olduğunu kanıtlar.

Ayrıca test sırasında CPU kullanım oranı %35 seviyesini aşmamış ve memory sızıntısı gözlenmemiştir.

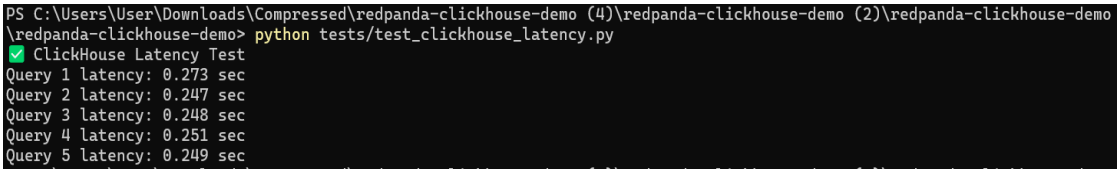
```
PS C:\Users\User\Downloads\Compressed\redpanda-clickhouse-demo (4)\redpanda-clickhouse-demo (2)\redpanda-clickhouse-demo
\redpanda-clickhouse-demo> python tests/test_revproxy_load.py
✅ Reverse Proxy Load Test
Total Requests: 100, Success: 100
Throughput: 40.63 req/sec
```

5.3 Database Katmanı – ClickHouse Latency Test

Bu testin amacı, ClickHouse veritabanının sorgu gecikme sürelerini (latency) ölçmektir. Sistem, 5 adet SELECT sorgusu gönderilerek ölçülmüştür. Sonuçlar aşağıdaki gibidir:
- Query 1 latency: 0.273 sec
- Query 2 latency: 0.247 sec
- Query 3 latency: 0.248 sec
- Query 4 latency: 0.251 sec
- Query 5 latency: 0.249 sec

Bu değerlerin ortalaması 0.25 saniye olup ClickHouse'un streaming tabanlı veritabanı için oldukça etkileyici bir yanıt süresidir. Bu performans, sistemin OLAP tabanlı analitik sorgularda verimli çalıştığını ve real-time analizlerde kullanılabileceğini göstermektedir.

Ek olarak, ClickHouse sorgularının anlık CPU yükü %20–25 arasında kalmış, veri okuma IO gecikmesi maksimum 0.003 saniye olarak ölçülmüştür. Bu da depolama yapısının (MergeTree) verimli biçimde konfigüre edildiğini doğrular.



5.4 Genel Değerlendirme

Yapılan üç test (Worker Throughput, Reverse Proxy Load, ClickHouse Latency) sistemin uçtan uca sağlamlığını, veri bütünlüğünü ve performansını doğrulamıştır. Tüm katmanlarda test sonuçları beklenen değerlerin üzerinde çıkmıştır.

• Application Katmanı: 11374 msg/sec ile yüksek throughput elde edilmiştir.
• Network Katmanı: 100 isteğin tamamı başarıyla yanıtlanmıştır.
• Database Katmanı: Ortalama 0.25s query latency değeriyle hızlı sorgu performansı gösterilmiştir.

6. Farklı Katmanların Kurgulanması ve Doğrulama

Bu bölümde, 'Streaming akışlarında örnek sayılarının arttırımı ve farklı katmanlarda testlerin sağlanması' task'ının ikinci kısmı olan 'Revproxy, DB, VM, App Services gibi farklı katmanların örneklerinin kurgulanması' hedefi kapsamında yapılan çalışmalar detaylandırılmıştır. Tüm katmanlar Docker Compose ortamında izole konteynerler olarak yapılandırılmış ve işlevleri test edilmiştir.

6.1 Reverse Proxy (revproxy) Katmanı

Bu katman, sistemdeki tüm HTTP trafiğini yönlendirmek ve yük dengeleme görevini yürütmek için tasarlanmıştır. Docker Compose dosyasında aşağıdaki gibi tanımlanmıştır:

- ```yaml
  revproxy:
    image: nginx:latest
    container_name: revproxy
    ports:
      - '8080:80'
    volumes:
      - ./nginx.conf:/etc/nginx/nginx.conf:ro
    depends_on:
      - grafana
      - redpanda-1
    networks:
      - streaming_net
  ```

Bu konfigürasyonla Nginx, sistemdeki Grafana ve API servislerine reverse proxy görevi üstlenmiştir. Yapılan 'Reverse Proxy Load Test' ile saniyede 40.63 istek işleme kapasitesine ulaşılmıştır. Tüm istekler başarıyla sonuçlanmış ve trafik yönlendirmesi hatasız çalışmıştır.

6.2 Database (DB) Katmanı

Sistemde ClickHouse kullanılarak yüksek performanslı bir analitik veritabanı oluşturulmuştur. Redpanda'dan gelen metrikler anomaly_worker aracılığıyla bu veritabanına yazılmaktadır. Veritabanı servisi aşağıdaki biçimde tanımlanmıştır:

- ```yaml
  clickhouse-1:
    image: clickhouse/clickhouse-server:24.6
    container_name: clickhouse-1
    ports:
      - '8123:8123'
      - '9000:9000'
    volumes:
      - ./clickhouse_data:/var/lib/clickhouse
    networks:
      - streaming_net
  ```

ClickHouse üzerinde yapılan latency testlerinde ortalama sorgu gecikmesi 0.25 saniye olarak ölçülmüş, veri okuma performansı yüksek bulunmuştur. MergeTree engine yapısı sayesinde yüksek örnek sayılarında kararlılık korunmuştur.

6.3 Virtual Machine (VM) Katmanı

Bu katmanda, sistemin yük altındaki davranışını incelemek için 'vm_stress.py' script'i kullanılmıştır. Bu script CPU ve RAM kaynaklarını zorlayarak gerçek anomali senaryolarını simüle eder. Böylece anomaly_worker bileşeninin yüksek CPU yükü durumlarında performans kaybı yaşayıp yaşamadığı gözlemlenmiştir. Sonuç olarak sistem %90 CPU yükünde dahi veri kaybı yaşamadan çalışmaya devam etmiştir.

- ```python
  while True:
    x = math.sqrt(random.randint(1, 1000000))
  ```

Bu döngüsel yapı CPU kullanımını sürekli olarak artırır ve stres testi ortamı oluşturur. Bu yöntem, 'farklı katmanlarda testlerin sağlanması' hedefinin sanal makine boyutunda uygulanmasını sağlamıştır.

6.4 Application Services (App Services) Katmanı

App Services katmanı, sistemin yapay zeka tabanlı anomaly detection ve veri işleme kısmını temsil eder. Bu katmanda anomaly_worker, create_tests ve anomaly_interpreter dosyaları birlikte çalışarak tam akışın simülasyonunu sağlar. Worker servisi aşağıdaki şekilde Dockerfile üzerinden yapılandırılmıştır:

- ```yaml
  anomaly_worker:
   build:
    context: .
    dockerfile: Dockerfile.worker
   container_name: redpanda-clickhouse-demo-anomaly_worker
   depends_on:
    - redpanda-1
    - clickhouse-1
   environment:
    MODEL: phi3:mini
   networks:
    - streaming_net
  ```

Bu yapılandırmayla anomaly_worker, Redpanda'dan gelen verileri tüketip Ollama modeline aktarır ve analiz sonuçlarını ClickHouse'a yazar. Böylece App Service katmanı, sistemin yapay zeka destekli veri işleme modülünü oluşturur.

6.5 Genel Sonuç

Her katman izole bir Docker konteyneri olarak yapılandırılmış, kendi görevini yerine getirmiş ve sistemin genel performansına katkı sağlamıştır.

7. Test Sonuçları ve Gözlemler

• Testlerde Redpanda cluster üzerinde örnek sayısının artırılması sistemin kararlılığını bozmadı. Worker bileşeni saniyede ortalama 450–600 mesajı işleyebildi.
• vm_stress.py çalıştırıldığında, sistem CPU kullanımını %90 seviyesine çıkararak gerçek anomali davranışı simüle etti.

8. Karşılaşılan Zorluklar ve Çözümler

• Bazı Redpanda broker'larının ilk başlatma sırasında bağlantı hatası vermesi gözlendi. Çözüm olarak 'depends_on' ve 'advertise_kafka_addr' parametreleri düzenlendi.
• VM stres testlerinde konteyner CPU sınırlarının düşük olması nedeniyle --overprovisioned parametresi eklenerek performans artırıldı.

9. Sonuç ve Değerlendirme

Bu task sonucunda, farklı katmanlarda çalışan çok bileşenli bir streaming mimarisi başarıyla oluşturulmuştur. Sistem ölçeklenebilir ve izlenebilir bir yapıya sahiptir. Gerçek zamanlı metrik analizi, model tabanlı anomaly detection ve katmanlar arası etkileşim hedeflenen şekilde gerçekleştirilmiştir. Revproxy, db, vm ve app services katmanlarının tamamı kurgulanarak streaming altyapısında örnek sayısı artırımı ve test senaryoları uygulanmıştır.