Senkronizasyon

Semafor nedir ?

Semafor, iş parçacıkları (thread) ya da işlemler (process) arasında senkronizasyonu sağlamak için kullanılan bir kontrol mekanizmasıdır. Temel amacı, aynı anda birden fazla işlemin paylaşılan bir kaynağa erişmesini kontrol altına almak ve çakışmaları önlemektir. İki tür Semafor vardır ilki binary(ikili) kaynağa sadece tek işlemin erişmesini sağlar. Mutex(kilit) gibi davranır. İkincisi ise Counting semafor (saymalı semafor) değeri sıfırdan büyük herhangi bir sayı alabilir birden fazla işlemi sınırlı sayıda kaynağı kullabilir.

Aşağıda basit 2 ipliğin kullanabildiği bir semafor python kodu bulunmakatadır.

Semafor.py

```python
import threading

semafor = threading.Semaphore(2)  # En fazla 2 işlem girebilir

def kaynak_kullan():
    with semafor:
        print(f"{threading.current_thread().name} kaynağı kullanıyor")
        import time
        time.sleep(2)
        print(f"{threading.current_thread().name} işi bitirdi")

for i in range(4):
    threading.Thread(target=kaynak_kullan).start()
```

Mutex nedir ?

Mutex (Mutual Exclusion - Karşılıklı Dışlama), çoklu iş parçacığı (thread) ya da işlem (process) kullanılan ortamlarda aynı anda yalnızca birinin belirli bir kritik bölgeye (örneğin bir dosya, bellek alanı, veri tabanı satırı vb.) erişmesini sağlamak için kullanılan bir eşzamanlama (senkronizasyon) aracıdır. Mutex, bir çeşit kilittir. Sadece bir thread kilidi alabilir, diğerleri beklemek zorundadır.

Aşağıda basit bir Mutex örneği verilmiştir.

Mutex.py

```python
import threading
import time

# Ortak kaynak
sayac = 0

# Mutex (kilit nesnesi)
mutex = threading.Lock()

def sayac_artir():
    global sayac
    for _ in range(5):
        mutex.acquire()  # Kilidi alma kodu
        temp = sayac
        time.sleep(1)  # işlem uzun sürüyormuş gibi davran
        sayac = temp + 1
        print(f"{threading.current_thread().name}: sayac = {sayac}")
        mutex.release()  # Kilidi bırakma kodu
```

```
# İki iş parçacığı oluştur
thread1 = threading.Thread(target=sayac_artir, name="Thread-1")
thread2 = threading.Thread(target=sayac_artir, name="Thread-2")

# Başlat
thread1.start()
thread2.start()

# Ana thread, diğerlerinin bitmesini bekler
thread1.join()
thread2.join()

print(f"Final sayaç değeri: {sayac}")
```

Uygulama: Python'da semafor ve mutex ile yarış koşullarını engelleme kodları aşağıda verilmiştir.

Mutex_race_conditions.py

```
import threading
import time

sayac = 0
lock = threading.Lock()

def sayac_artir():
    global sayac
    for _ in range(1000):
        with lock:
            temp = sayac
            temp += 1
            sayac = temp

thread1 = threading.Thread(target=sayac_artir)
thread2 = threading.Thread(target=sayac_artir)

thread1.start()
thread2.start()
thread1.join()
thread2.join()

print(f" Mutex ile sayaç sonucu: {sayac}")
```

Semafor_ race_conditions.py

```
import threading
import time

sayac = 0
semafor = threading.Semaphore(1)  # 1 → binary semaphore (mutex gibi
davranır)

def sayac_artir():
    global sayac
    for _ in range(1000):
        semafor.acquire()
        temp = sayac
        temp += 1
        sayac = temp
        semafor.release()
```

```
thread1 = threading.Thread(target=sayac_artir)
thread2 = threading.Thread(target=sayac_artir)

thread1.start()
thread2.start()
thread1.join()
thread2.join()

print(f"Semaphore ile sayaç sonucu: {sayac}")
```

Tutarlılık

Eventual Consistency (Sonunda Tutarlılık), dağıtık sistemlerde kullanılan bir veri tutarlılığı modelidir. Bu modelde, tüm düğümler başlangıçta aynı veriye sahip olmayabilir, ancak zamanla hepsi tutarlı hale gelir.

a) Redis ve Cassandra ile Eventual Consistency testi nasıl yapılır ?

Gerekli ortamların kurulu olması yani redis ve cassandra ortamlarının kurulu olması gerekmektedir.

** Redis ile Eventual Consistency Testi :

Uygulama: Aşağıda python ile yazılmış test kodu bulunmaktadır.

Redis_cons.py

```
import redis
import time

clientA = redis.Redis(host='localhost', port=7000)  # Redis Cluster portu
örneği
clientB = redis.Redis(host='localhost', port=7001)  # Farklı node

# Client A yazıyor
clientA.set("test_key", "value1")
print("Client A wrote: value1")

# Client B hemen okuyor
print("Client B reads immediately:", clientB.get("test_key"))

# 2 saniye bekleyip tekrar okuma
time.sleep(2)
print("Client B reads after 2s:", clientB.get("test_key"))
```

** Cassandra ile Eventual Consistency Testi

Uygulama: Aşağıda python ile yazılmış test kodu bulunmaktadır.

Cassandra_cons.py

```
from cassandra.cluster import Cluster
from cassandra import ConsistencyLevel
from cassandra.query import SimpleStatement
import time
```

```
cluster = Cluster(['127.0.0.1'])  # Cassandra node IP
session = cluster.connect()

session.execute("CREATE KEYSPACE IF NOT EXISTS testks WITH replication =
{'class':'SimpleStrategy', 'replication_factor' : 2};")
session.set_keyspace('testks')
session.execute("CREATE TABLE IF NOT EXISTS data (id int PRIMARY KEY,
value text);")

# Client A: Yazma
stmt_write = SimpleStatement("INSERT INTO data (id, value) VALUES (1,
'eventual')", consistency_level=ConsistencyLevel.ONE)
session.execute(stmt_write)
print("Client A wrote with CL=ONE")

# Client B: Hemen okuma
stmt_read = SimpleStatement("SELECT value FROM data WHERE id=1",
consistency_level=ConsistencyLevel.ONE)
print("Client B reads immediately:", session.execute(stmt_read).one())

# Bekleme
time.sleep(2)
print("Client B reads after 2s:", session.execute(stmt_read).one())
```

****Not: Bu uygulamaları gerçekleştirmek için cassandra ve redis yüklü olması gerek aşağıda DynamoDB uygulaması olacak onun içinde DynamoDB indirmeniz şart.

** Amazon DynamoDB replikasyon deneyleri

Uygulama : Aşağıda DynamoDB kod örneği bulunmaktadır.

dynamoDB.py

```
import boto3
import time

# 1. US bölgesine veri yaz
us_client = boto3.resource('dynamodb', region_name='us-east-1')
eu_client = boto3.resource('dynamodb', region_name='eu-west-1')

table_name = "DeneyTablosu"

us_table = us_client.Table(table_name)
eu_table = eu_client.Table(table_name)

# Veri ekleme
us_table.put_item(Item={"id": "test1", "value": "replication test"})
print("US region wrote data.")

# Hemen okumaya çalış (diğer bölge)
response = eu_table.get_item(Key={"id": "test1"})
item = response.get("Item")
print("EU read immediately:", item)

# Bekle ve tekrar oku
time.sleep(5)
response = eu_table.get_item(Key={"id": "test1"})
item = response.get("Item")
print("EU read after 5s:", item)
```