Paralel Hesaplama

Öğr. Gör. Zafer SERİN

Hata Ayıklama için Thread Bilgileri ve Yönetimi

- t.name özelliği ile Thread'lere anlamlı isimler vermek, log takibi ve hata ayıklamayı kolaylaştırır.
- threading.current_thread(): Şu anda kodu yürüten Thread nesnesini döndürür.
- threading.active_count(): Yorumlayıcı içinde şu anda aktif olan (çalışan veya beklemede olan) tüm Thread sayısını verir.

Aktif Thread'leri Listeleme: enumerate()

- Şu anda aktif durumda olan tüm Thread nesnelerinin bir listesini döndürür.
- Hata ayıklama veya bir uygulamanın o anda hangi Thread'leri çalıştırdığını kontrol etmek için kullanılır.

Thread'in Canlılık Durumunu Kontrol Etmek

- Thread'in başlatılıp başlatılmadığını ve işini bitirip bitirmediğini kontrol eder.
- Eğer Thread, start() ile çağrılmış ve henüz Terminated durumuna geçmemişse (Runnable, Running, Blocked durumlarında) True döndürür.

Aktif Thread'leri Listeleme: enumerate()

```
import threading
import time

def isci():
    time.sleep(2)

# 2 tane isimsiz, 1 tane isimli thread başlatalım
threading.Thread(target=isci).start()
threading.Thread(target=isci).start()
threading.Thread(target=isci, name="ÖZEL_THREAD_X").start()

print(f"Toplam Aktif Thread Sayısı: {threading.active_count()}")

print("\n--- Thread Listesi ---")
for t in threading.enumerate():
    print(f"Thread Adı: {t.name}, Canlı mı?: {t.is_alive()}")
```

Thread Havuzuna Giriş

- Uygulamanın başlangıcında belirli sayıda (genellikle CPU sayısına eşit) Thread'in yaratılıp, kuyruktan iş almak üzere sürekli hazır bekletilmesidir.
- Her iş geldiğinde Thread yaratmanın getirdiği ek yükü (overhead) ortadan kaldırır. Thread yaratma maliyeti yüksek bir iştir.

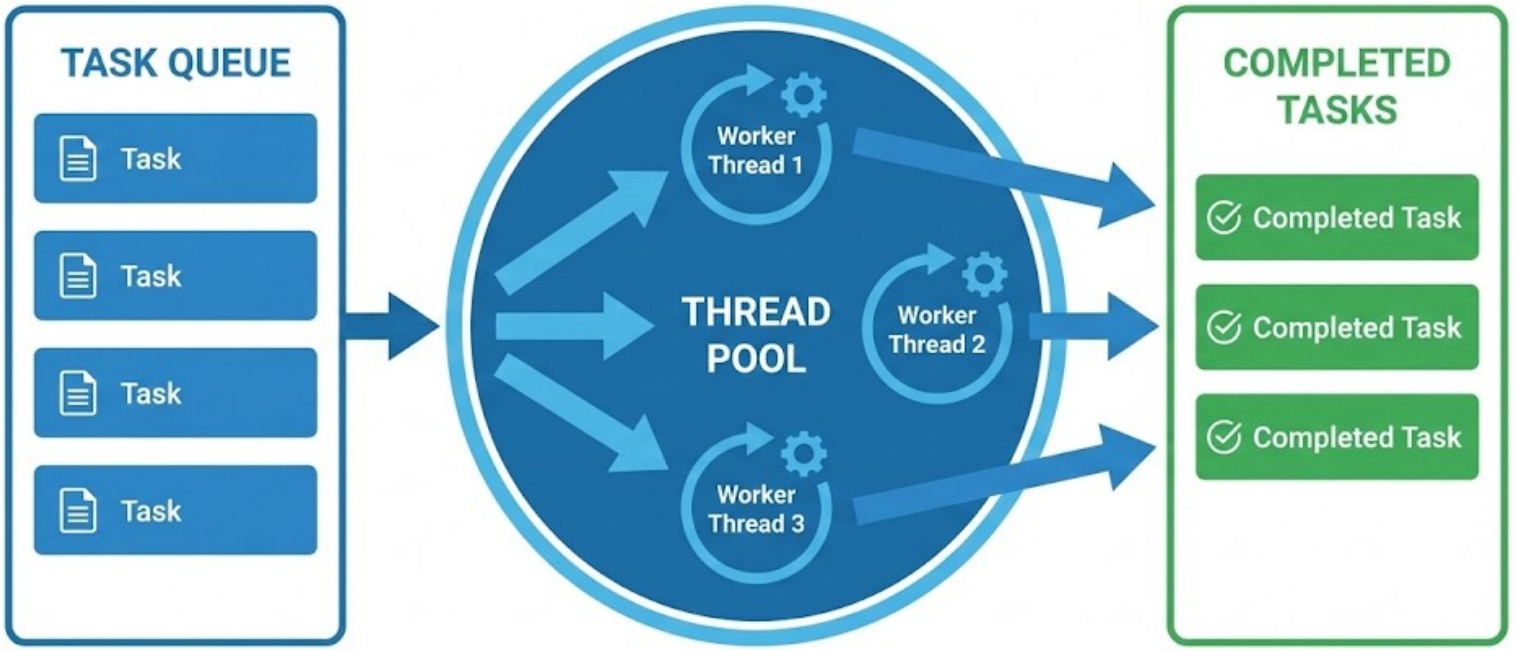
Thread Havuzu (Thread Pool) Mimarisi

- Veznedar Analojisi: Bir bankada 100 müşteri (Görev) olduğunu düşünün.
  - Havuzsuz Sistem: Her müşteri için yeni bir veznedar işe alınır, işlem bitince kovulur. (Çok maliyetli ve yavaş).
  - Havuzlu Sistem: Sadece 3 veznedar (Thread) vardır. Müşteriler sıraya girer. İşi biten veznedar, sıradaki müşteriyi alır.

Thread Havuzu (Thread Pool) Mimarisi

- Nasıl Çalışır?
  - Work Queue (İş Kuyruğu): Yapılacak görevler bir kuyrukta birikir.
  - Worker Threads (İşçi Thread'ler): Havuzdaki sabit sayıdaki Thread, kuyruktan sürekli iş çeker.
  - Reuse (Yeniden Kullanım): Görevi biten Thread yok olmaz, yeni görev almak üzere havuza döner.

Thread Havuzu (Thread Pool) Mimarisi İçerik:



THREAD POOL ARCHITECTURE

Python'da Modern Thread Havuzu Yönetimi

- Python'da Thread havuzlarını yönetmek için threading modülü yerine, daha yüksek seviyeli (high-level) olan concurrent.futures kütüphanesi tercih edilir.
- ThreadPoolExecutor Sınıfı:
  - Havuzun yöneticisidir. Kaç işçi çalışacağını (max_workers) belirler.
  - with bloğu ile kullanıldığında, tüm işler bitene kadar programın kapanmamasını ve kaynakların temizlenmesini otomatik yönetir.
- Temel Metotlar:
  - submit(fn, *args): Tek bir işi havuza atar.
  - map(fn, list): Bir listeyi (örn: dosya isimleri) sırayla işçilere dağıtır.

Python'da Modern Thread Havuzu Yönetimi

- Thread Havuzu ile Görev Dağıtımı

```
from concurrent.futures import ThreadPoolExecutor
import time
import threading

def gorev_yap(is_no):
    # Hangi Thread'in çalıştığını görelim
    thread_adi = threading.current_thread().name
    print(f"[{thread_adi}] Görev {is_no} başladı...")
    time.sleep(1) # İşlem simülasyonu
    return f"Görev {is_no} Bitti"

# Havuz oluşturuluyor (Maksimum 2 İşçi)
print("--- Havuz Açılıyor (Sadece 2 İşçi Var) ---")
with ThreadPoolExecutor(max_workers=2) as havuz:
    # 5 adet görevi havuza gönderiyoruz
    # map fonksiyonu, for döngüsü gibi çalışır ve işleri dağıtır
    sonuclar = havuz.map(gorev_yap, range(1, 6))

print("--- Tüm İşler Tamamlandı ---")
```

Performans ve Kaynak Yönetimi Avantajları

1. Overhead (Ek Yük) Azalması:
   - Thread oluşturmak (New -> Runnable) ve yok etmek (Terminated) işletim sistemi için maliyetli bir işlemdir. Havuz, bu maliyeti sadece başlangıçta bir kez öder.
2. Kaynak Kontrolü:
   - Sisteme aynı anda 10.000 istek gelirse ve her biri için Thread açarsanız sunucu çöker (Memory Error).
   - Havuz ile "Aynı anda en fazla 50 kişi çalışsın, diğerleri sırada beklesin" diyerek sistemi korursunuz.

- Sonuç: Daha kararlı (stable), tahmin edilebilir ve performanslı uygulamalar.