Bitirme Projesi

Konu: Doğru ve Yanlış Yapılandırılmış PostgreSQL Sunucularında Paralel Programlamanın Performansı

Hazırlayan: Defne Turğut

Projenin Amacı:

PostgreSQL veritabanı sunucularının farklı yapılandırma ayarlarının sunucu performansı üzerindeki etkisini test ederek gözlemlemektir. Veritabanı performanslarında donanım özelliklerinin büyük bir etkisi olsa da, aynı zamanda doğru yapılandırma ile performans da doğrudan ilişkilidir.
Doğru bir test ortamında çalışmak için aynı donanım özelliklerine sahip iki sanal sunucu kurulmuştur.

RAM : 8GB RAM

CPU: 4 CPU
Her iki sunucu üzerinde PostgreSQL veritabanı kurulumu gerçekleştirilmiş, Ubuntu 24.04 LTS işletim sistemi ile yapılandırılmıştır.

Ancak bu iki sunucudan biri sanal makinenin ve veritabanının sistem kaynaklarına göre optimize edilmiş doğru konfigürasyon değerleri ile ayarlanırken, diğeri hatalı parametreler ile konfigüre edilmiştir.

Değiştirilen Konfigürasyon Parametreleri:

- **shared_buffers:** PostgreSQL'in sorgular sırasında verileri bellek içinde tutmak için ayırdığı alanı belirler. Veritabanı sunucusu, diskten okunan verileri ilk olarak bu bellekte saklar.
  Etkisi: Bu alan yeterli büyüklükte ayarlanırsa, sık erişilen veriler bellekte tutulur ve disk erişimi azalır, böylece performans artar.
  Düşükse: Her sorguda disk erişimi yapılır, bu da sistemi yavaşlatır.
  Yüksekse: Sistem belleğini aşabilir, işletim sisteminin performansını düşürür.
- **work_mem:** PostgreSQL'in sıralama, birleşim (JOIN), filtreleme gibi işlemler sırasında her bağlantı başına kullandığı geçici bellek miktarını belirler.
  Etkisi: Yeterli büyüklükte ayarlanırsa, bu işlemler bellekte tamamlanabilir ve disk üzerinde geçici dosya oluşturulmasına gerek kalmaz. Bu da sorgu sürelerini önemli ölçüde azaltır.
  Düşükse: Diskte geçici dosyalar oluşur, performans düşer.
  Yüksekse: Eşzamanlı bağlantı sayısı fazlaysa toplam bellek kullanımı aşırı artar ve sistem çökebilir.
- **effective_cache_size:** PostgreSQL'e işletim sisteminin disk önbelleği olarak kullanabileceği toplam RAM miktarının tahmini bir değerini bildirir.
  Etkisi: Sorgu planlayıcısı, verilerin bellekte olma olasılığına göre daha etkili sorgu yolları seçebilir. Yüksek bir değer, daha agresif ve hızlı index kullanımı sağlar.
  Düşükse: PostgreSQL sorguların daha yavaş yollarını tercih edebilir.

Yüksekse (gerçekçi olmayan değer): Sorgu planlayıcısı yanıltılır ve yanlış planlar seçebilir.

- **max_connections:** Aynı anda veritabanına kaç istemcinin bağlanabileceğini belirler.
  Etkisi:Yüksek kullanıcı trafiği olan sistemlerde yeterince yüksek tutulmalıdır ki bağlantılar reddedilmesin. Makinenin donanımsal özelliklerine göre kapasitesine uygun bağlantı sayısı optimum değerdir.
  Çok düşükse: Bağlantılar reddedilir, uygulama hataları oluşabilir.
  Çok yüksekse: Her bağlantı için bellek ayrıldığı için toplam RAM hızla tükenir, sistem aşırı yüklenebilir.
- **wal_buffers:** PostgreSQL'in Write-Ahead Logging (WAL) sırasında kullanılan geçici belleği belirler. Bu loglar, verilerin kalıcılığını ve tutarlılığını sağlar.
  Etkisi: Doğru ayarlanırsa, veri değişiklikleri bellekte toplanıp disk yazımı daha verimli hale gelir. Bu da yazma işlemlerinin hızını artırır.
  Düşükse: Çok sık disk yazımı olur, I/O yükü artar.
  Yüksekse: Bellek israfı olabilir, özellikle düşük trafikli sistemlerde.
- **checkpoint_segments**: PostgreSQL 9.5+ sürümlerinde bu parametrenin yerine **max_wal_size**, **min_wal_size** ve **checkpoint_timeout** kullanılmaktadır.
  - **max_wal_size:** Checkpoint işlemi başlamadan önce WAL (Write-Ahead Log) dosyalarının toplam büyüklüğünün ulaşabileceği maksimum değeri belirler.
    Etkisi: Yüksek ayarlanırsa, checkpoint işlemleri daha seyrek yapılır, böylece disk I/O azaltılır ve yazma performansı artar.
    Düşükse: Çok sık checkpoint yapılır, bu da diske aşırı yazma ve yavaşlama demektir.
    Yüksekse: Sistem çökmesinde kurtarma süresi uzayabilir, veri kaybı riski artar.
  - **min_wal_size:** Checkpoint'ler arası tutulması gereken minimum WAL dosyası miktarını belirler.
    Etkisi: Sistemin yük durumuna göre WAL dosyalarının sıkça oluşturulup silinmesini engeller, böylece I/O yükünü dengeler.
    Düşükse: WAL dosyaları sürekli silinip yeniden oluşturulabilir, bu da disk kullanımı ve sistem performansını olumsuz etkiler.
    Yüksekse: Gereksiz yere fazla disk alanı kullanılabilir.
  - **checkpoint_timeout:** Checkpoint işlemleri arasında geçmesi gereken maksimum süredir. Belirtilen süre dolmadan önce max_wal_size aşılmamışsa bile checkpoint tetiklenir.
    Etkisi: Uzun süreli işlemlerde veri kaybını önlemek ve kurtarma süresini kısaltmak için kritik bir parametredir.
    Kısa süre: Çok sık checkpoint yapılır, bu da sistem kaynaklarını zorlar.
    Uzun süre: Olası bir sistem arızasında kurtarma süresi artar.

PostgreSQL Yapılandırma Parametreleri

Parametre	Doğru Yapılandırma	Yanlış Yapılandırma
shared_buffers	2GB	128kB
work_mem	20164kB	64kB
max_connections	100	1000
wal_buffers	16MB	32kB
effective_cache_size	6GB	128MB
checkpoint_timeout	15min	30s
max_wal_size	4GB	64MB
min_wal_size	1GB	32MB

Yukarıda parametrelerle ilgili verilen bilgiler doğrultusunda iki sunucu için de optimize ve yanlış değerler test amaçlı **postgresql.conf** dosyasından değiştirilmiştir.
Max_connection değerinin belirli bir optimize değeri olmayıp sanal makinenin özelliklerine göre en iyi performansını ortalama **100** bağlantı için verebileceği ön görülmüş, yanlış konfigürasyondan gereğinden fazla değer alarak **1000** olarak ayarlanmıştır. Doğru konfigürasyon için diğer parametreler max_connection değerine bağlı olarak optimize edilmiştir.

Veri Üretimi:

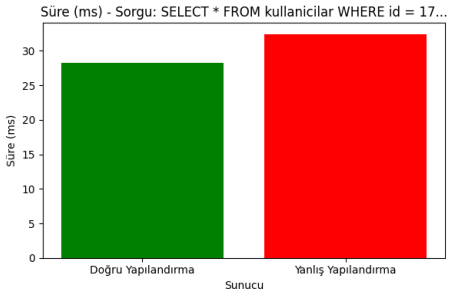
Her iki sunucudaki veritabanlarına Python ile bağlanarak **kullanicilar** tablosu oluşturulmuştur. Tablo bilgisi;

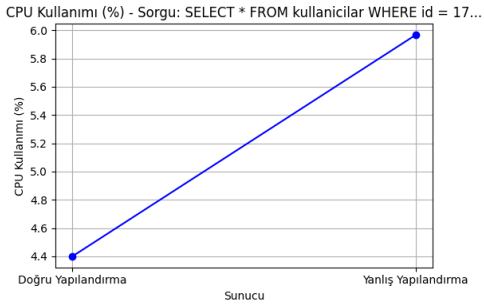
```
CREATE TABLE kullanicilar (
    id SERIAL PRIMARY KEY,
    name TEXT,
    surname TEXT,
    eposta TEXT,
    dogum_tarihi DATE,
    olusturma_zamani TIMESTAMP
);
```

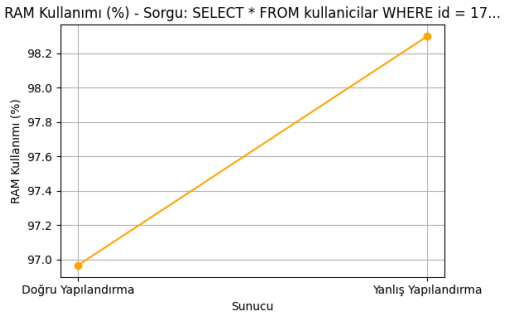
İki veritabanındaki tablolara **INSERT** komutu ile 10 milyon **sahte(dummy)** veri üretilmiştir.

Sorgu Performans Analizi:

1. **Sorgu: SELECT * FROM kullanicilar WHERE id = 172;**



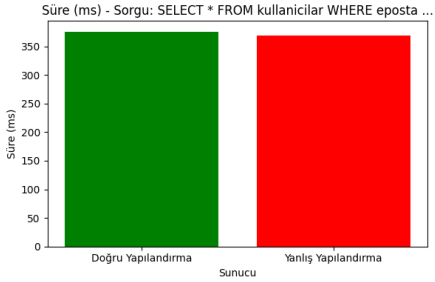


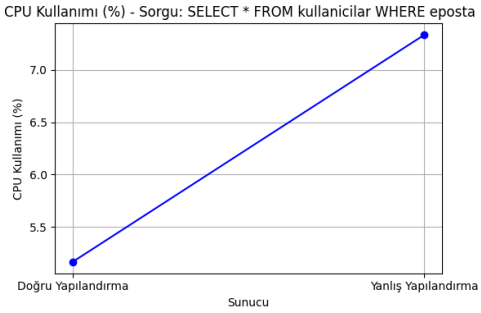


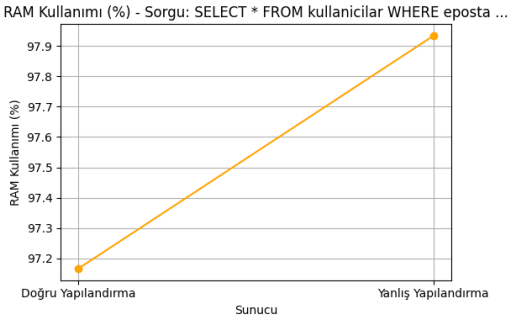
- Bu sorgu id üzerinden arama yaptığı için iki sunucuda da hızlı çalışmıştır. Doğru yapılandırma, sorgunun ortalama süresini yaklaşık %13 oranında azaltmıştır (28.2 ms vs 32.4 ms).
- CPU kullanımı yanlış konfigürasyonda daha yüksek (5.97% vs 4.40%), bu da verimsiz kaynak kullanımına işaret eder.
- RAM kullanımı her iki durumda da oldukça yüksek, ancak yanlış konfigürasyonda biraz daha fazla (%98.3 vs %96.97), bu da sistemin aşırı yüklenebileceğini gösterir.

Yorum:
Doğru yapılandırma, birincil anahtar sorgularında daha hızlı ve verimli çalışıyor. Yanlış ayarlarda, basit bir id araması bile gereksiz CPU ve RAM kullanımına sebep olabiliyor. Bu durum büyük ölçekli uygulamalarda performans azalmasına yol açabilir.

2. **Sorgu: SELECT * FROM kullanicilar WHERE eposta = 'user_b76abc58@example.com';**





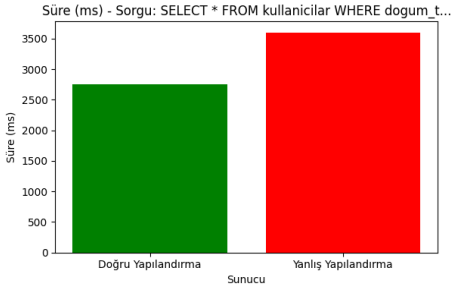


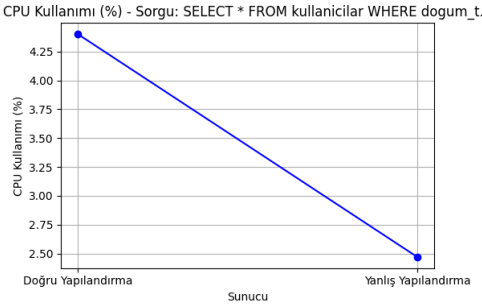
- Eposta ile araması id aramasına göre daha karmaşık sorgu olduğu için iki sunucuda da daha uzun sürmüştür.
- İlginç şekilde, yanlış yapılandırmada sorgu süresi %2 civarında daha düşüktür (368 ms vs 375 ms). Bunun sebebi yanlış sunucuda bellekte aranan sorgunun beklenenden daha hızlı bulunması olabilir.
- Ancak CPU kullanımı yanlış konfigürasyonda oldukça yüksek. Yani süreler benzer çıksa da yanlış konfigürasyon bu sürede bulmak için daha fazla kaynak tüketmiştir. Bu da yanlış ve doğru sunucu aynı sürede çalıştıkları durumda yanlış sunucunun bu işlem için daha fazla kaynak tükettiğini göstermiştir.(7.33% vs 5.17%).
- RAM kullanımı yine her iki durumda yüksek, ancak yanlış konfigürasyon biraz daha fazla RAM kullanıyor.

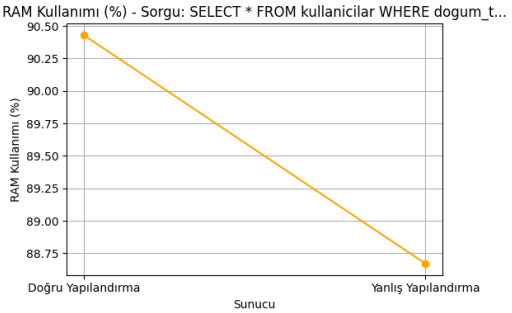
Yorum:

Sorgu süresi açısından büyük fark yok; ancak yanlış konfigürasyon daha fazla CPU ve RAM tüketiyor. Bu da kaynakların gereksiz yere kullanıldığı anlamına gelir. Yanlış yapılandırma, optimize edilmemiş ayarlarla performansı düşürmese de kaynak verimliliğini kötü etkiliyor.

3. **Sorgu: SELECT * FROM kullanicilar WHERE dogum_tarihi BETWEEN '2001-07-13' AND '2024-07-13';**







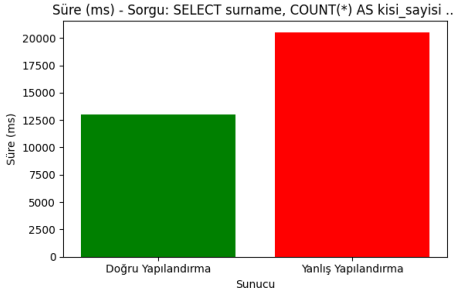
- Tarih aralığında filtreleme büyük veri setlerinde oldukça maliyetli bir sorgudur. Burada dikkat çekici olan, yanlış yapılandırmada sorgu süresinin yaklaşık %15 daha

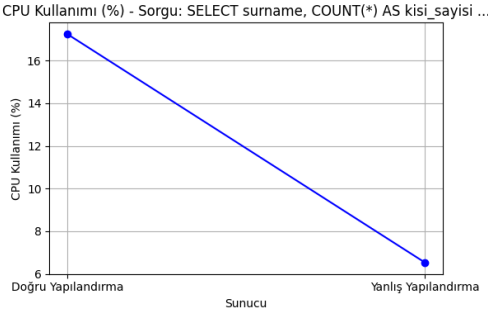
- kısa olması (3975 ms vs 4669 ms). Bu da yapılandırma ayarlarının böyle maliyetli bir sorgu için oldukça önemli olduğunu gösterir.
- Doğru konfigürasyonda daha hızlı işlem yapabilmek için CPU kullanımı iki katı kadar fazladır.
- Aynı şekilde RAM kullanımı da doğru konfigürasyonda daha fazladırç

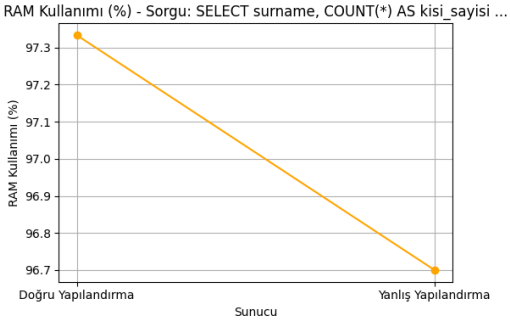
Yorum:

Böyle maliyetli bir sorgunun doğru konfigürasyonda daha hızlı çalışması beklenen bir durumdur. Hızlı çalışmak için kaynakları aktif kullanımı yüksek tutması da olası sonuçlardan biridir.

4. **Sorgu: SELECT surname, COUNT(*) AS kisi_sayisi FROM kullanicilar GROUP BY surname ORDER BY surname DESC;**







- Gruplama ve sıralama içeren bu sorgu, büyük veri setlerinde oldukça kaynak tüketen ve uzun süren bir işlemdir.
- Doğru konfigürasyon, sorgu süresini %36 oranında azaltarak yaklaşık 13 saniyede sonuç verirken, yanlış konfigürasyon 20.5 saniyeyi buluyor.
- CPU kullanımı doğru konfigürasyonda neredeyse 3 kat daha yüksek (%17.23 vs %6.53).
- RAM kullanımı her iki durumda da oldukça yüksek ve yakın seviyede (%97+).

Yorum:
Doğru yapılandırma, paralel işlemler ve bellek yönetimi sayesinde bu kompleks sorguda çok daha verimli çalışıyor. Yüksek CPU kullanımı, kaynakların aktif ve doğru şekilde kullanıldığı anlamına gelir. Yanlış yapılandırmada CPU kullanımının düşük olması, kaynakların tam anlamıyla kullanılmadığını ve bunun da sorgu süresinin uzamasına neden olduğunu gösterir. Bu sorgu, yapılandırma ayarlarının paralel işlem ve bellek yönetimindeki etkisini açıkça ortaya koymaktadır.

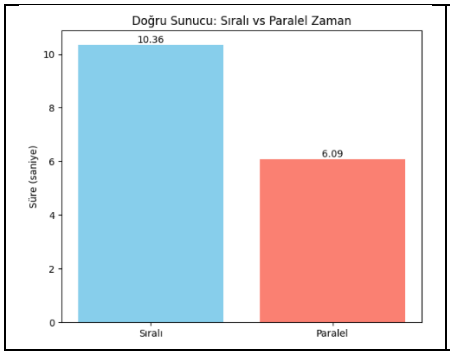
Paralel Programlama Testi:

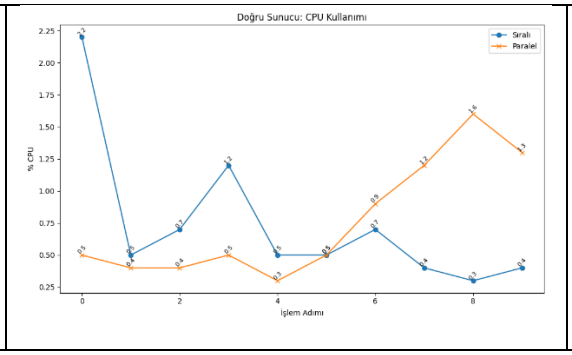
Sorgu: SELECT * FROM kullanicilar WHERE id = {user_id}

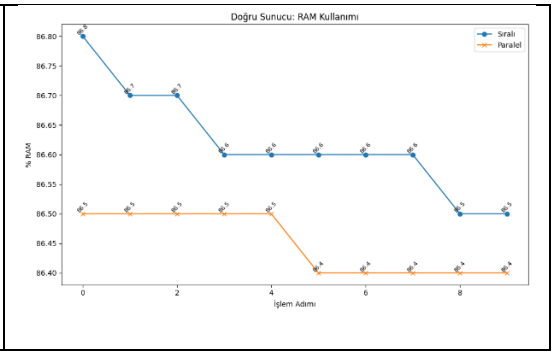
10 id için sırayla veya paralel olarak kullanıcıları getirme

Paralel programlama testlerinde paralel işlemleri yapmak için **concurrent.futures** kütüphanesinde faydalanılmıştır.

1. Doğru Yapılandırma İçin Sıralı vs Paralel Programlama





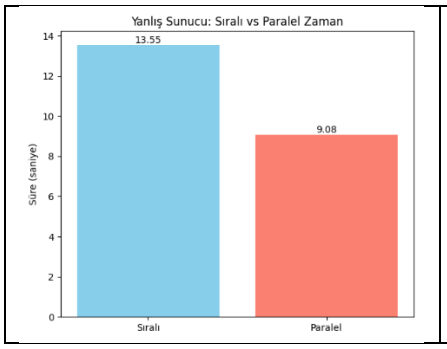


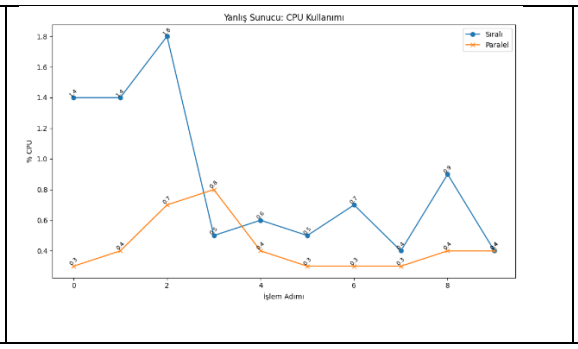
- Doğru sunucuda beklendiği gibi paralel programlamada sorgular daha hızlı çalışmıştır.
- Sıralı sorguda CPU kullanımı çok yüksekten başlayarak düşerek ara ara artarak devam etmiş ama paralel programlamada CPU kullanımı thread ler arasında artarak devam etmiştir.
- RAM kullanımı sıralı sorguda daha yüksek olduğu görülmüştür.

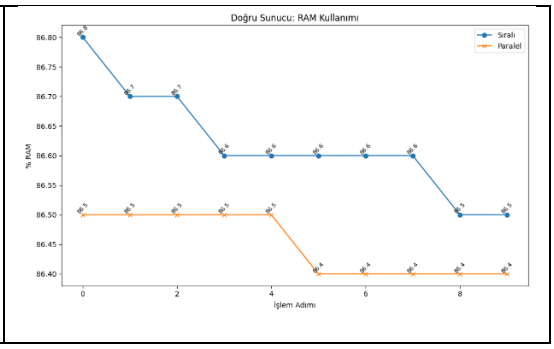
Yorum:

Paralel programlama, hem daha kısa işlem süresi, hem de daha dengeli kaynak kullanımı sağlamaktadır. Sıralı programlama, daha fazla zaman almakta ve kaynak kullanımı açısından dalgalı bir profile sahiptir. Test sonuçları, PostgreSQL gibi veritabanı sistemleri üzerinde çoklu sorgularla çalışan uygulamalar için paralel programlamanın açık bir tercih sebebi olduğunu ortaya koymaktadır.

2. Yanlış Yapılandırma İçin Sıralı vs Paralel Programlama



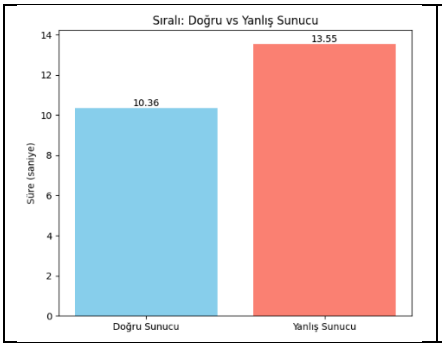


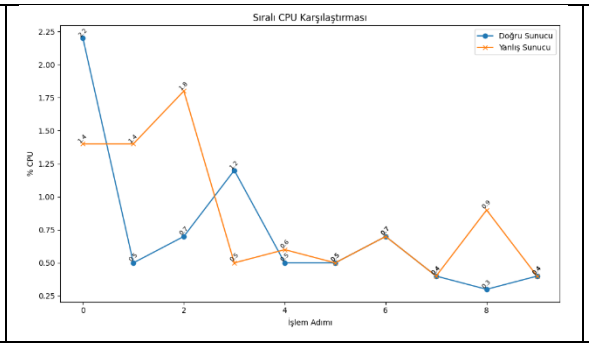


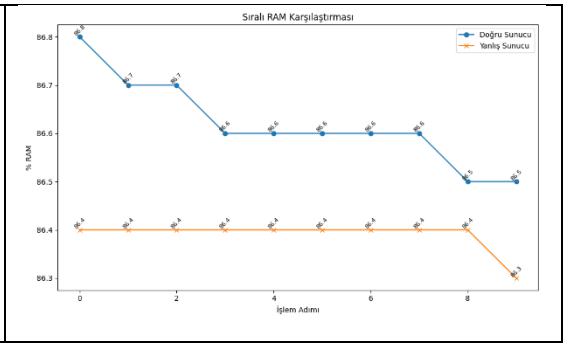
- Yanlış sunucuda kendi içinde paralel programlama %33 oranında daha hızlıdır.
- Kaynak kullanımı sıralı sorguda daha yüksektir.

Yorum:
Yanlış yapılandırılmış sunucuda paralel programlama sınırlı bir performans artışı sunsa da, kaynaklar (özellikle CPU) yeterince verimli kullanılamamaktadır. Bu da sistemin donanımsal veya konfigürasyonel darboğazlar nedeniyle paralel işlemlerde beklenen faydayı tam olarak sağlayamadığını göstermektedir.

3. Sıralı Testlerde Doğru vs Yanlış Yapılandırma



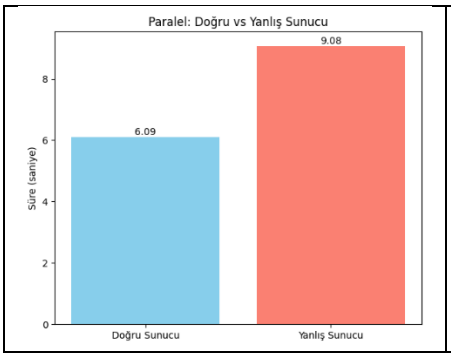


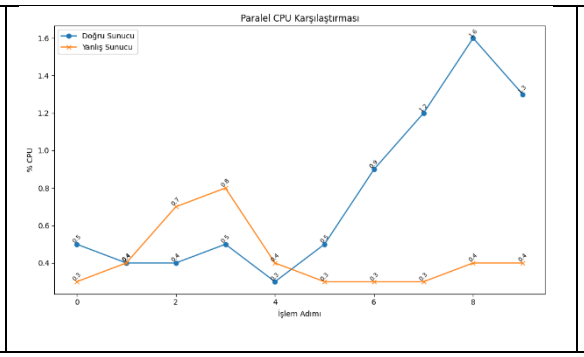


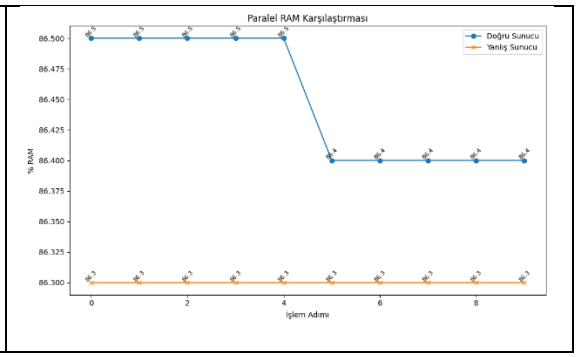
- Sıralı çalıştırma sırasında yanlış sunucu doğru sunucuya göre %31 daha yavaş çalışmıştır.
- Doğru sunucuda CPU kullanımı başlangıçta yüksek olsa da işlem adımları ilerledikçe daha düşük ve dengeli bir seviyeye gelmiştir. Yanlış sunucuda ise CPU kullanımı genellikle daha dalgalı ve zaman zaman doğru sunucudan daha yüksektir.
- Doğru sunucuda RAM kullanımı zamanla azalan bir eğilim gösterirken, yanlış sunucuda RAM kullanımı sabit kalmış ve daha düşük seviyededir.

Yorum:
Doğru yapılandırılmış sunucu, işlem süresi, CPU dengesi ve aktif RAM yönetimi açısından daha yüksek bir performans sergilemiştir. Yanlış yapılandırılmış sunucu, aynı görevleri daha uzun sürede tamamlamakta ve kaynak kullanımında dalgalanmalar göstermektedir. Bu verilere göre sistem yapılandırmasının sıralı işlemlerde dahi performans üzerinde kayda değer bir etkisi olduğunu gözlemlenir. Veritabanı performansını artırmak için yapılandırma doğruluğu, donanım ve kaynak yönetimi büyük önem taşır.

4. Paralel Programlamada Doğru vs Yanlış Yapılandırma







- Doğru yapılandırılmış sistemin paralel programlamayı yaklaşık %33 daha hızlı tamamlamıştır.
- Doğru sunucuda CPU kullanımı zamanla artan bir grafik izlemekte ve işlemlerin ilerleyen adımlarında daha yoğun kullanım göstermektedir (son adımda %1.6'ya kadar çıkmakta).Yanlış sunucuda ise CPU kullanımı daha sınırlı ve sabit kalmıştır (%0.3 - %0.6 aralığında), bu da kaynakların yeterince kullanılamadığını göstermektedir.
- Doğru sunucuda RAM kullanımı başlangıçta sabit olup 5. adım sonrası hafifçe düşüş göstermektedir (%86.5 → %86.4).Yanlış sunucuda RAM kullanımı tamamen sabit kalmış ve daha düşük seviyededir (%86.3).

Yorum:
Doğru yapılandırılmış sunucu, işlem süresi, CPU verimliliği ve RAM yönetimi açısından belirgin bir üstünlük sağlamaktadır. Yanlış yapılandırılmış sunucu, paralel işlemleri tam kapasiteyle çalıştıramamakta, bu da performans kazancını sınırlamaktadır. Genel olarak bu analiz, paralel programlamadan yüksek verim almak için yalnızca yazılım tarafında değil, donanım ve sistem yapılandırmasının da doğru yapılmasının hayati önem taşıdığını açıkça ortaya koymaktadır.

Özetle;

Yapılan dört testin (sıralı ve paralel programlama ile doğru ve yanlış yapılandırılmış sunuculardaki performans karşılaştırmaları) genel analizine göre:

- Paralel programlama, her iki sunucuda da sıralı programlamaya kıyasla daha kısa sürelerde sonuç vermiştir. Ancak bu avantaj, yalnızca doğru yapılandırılmış sunucuda tam anlamıyla ortaya çıkmıştır.
- Doğru yapılandırılmış sunucu, işlem süresi, CPU etkinliği ve RAM kullanımı açısından her iki modelde de daha üstün bir performans göstermiştir.
- Yanlış yapılandırılmış sunucuda ise paralel programlamanın etkisi sınırlı kalmış, sistem kaynakları yeterince kullanılamamıştır.

Sonuç olarak, en verimli senaryo:
Doğru yapılandırılmış sunucu + Paralel programlama kombinasyonudur.
Bu yapılandırma, hem işlem süresini minimuma indirirken hem de sistem kaynaklarını dengeli ve verimli şekilde kullanarak optimum performansı sağlamaktadır.