**DUMLUPINAR ÜNİVERSİTESİ**

**TEZ RAPORU**

Seyit Ahmet ÖZDEMİR

11.06.2025

Dr. Öğr. Üyesi Muammer Akçay

# BLOCKİNG I/O İÇEREN SÜREÇLERİN, JAVA PLATFORMU VE SANAL THREAD KULLANIM PERFORMANSLARI ÜZERİNE ETKİLERİ: BİR ARAŞTIRMA

Seyit Ahmet ÖZDEMİR, 202285151045, 536 508 1057

*Kütahya Dumlupınar Üniversitesi Lisansüstü Eğitim Enstitüsü Evliya Çelebi Yerleşkesi Tavşanlı Yolu 10. km. Kütahya/Türkiye*

## 1. ÖZET

Bu rapor, Blocking I/O içeren süreçler için Java, Platform ve Sanal Thread kullanımının performans farklarını incelemektedir. Java'nın klasik tek thread (Single Thread), çoklu platform thread (Multi Platform Thread) ve sanal thread (Virtual Thread) kullanımı karşılaştırılarak, her bir yöntemin avantajları ve dezavantajları analiz edilmiştir. Bu çalışma, özellikle yüksek iş yükü altında çalışan sistemler için en verimli çözümü belirlemeyi amaçlamaktadır.

## 2. TEORİK ÇALIŞMALAR

Java gibi ardışık programlama dillerinde ilk zamanlarda program akışının kesintiye uğramaması için Thread modeli ortaya çıkmıştır. Bilgisayar işlemcilerindeki çekirdek sayısının artması ile multi thread mimarisi eş zamanlı programlama yeteneği ile birlikte gelmiştir [1].

### 2.1. Single Thread

Single Thread, tek bir işlemci çekirdeğinde çalışan ve tüm işlemleri sırayla gerçekleştiren bir işleme modelidir. Bu model, basitliği ve kolay yönetilebilirliği nedeniyle bazı uygulamalarda tercih edilir. Ancak, Blocking IO işlemleri sırasında CPU'nun bekleme süresi arttıkça performans düşer [3].

### 2.2. Multi Platform Thread

Multi Platform Thread, aynı anda birden fazla iş parçacığının (thread) çalıştırılmasına izin verir ve genellikle işletim sistemi seviyesinde yönetilir. Bu yöntem, CPU'nun daha verimli kullanılmasını sağlar ve Blocking I/O işlemleri sırasında diğer thread'lerin çalışmaya devam etmesine olanak tanır. Bununla birlikte, thread yönetimi ve bağlam değiştirme maliyetleri performansı olumsuz etkileyebilir​ [5], [6].

### 2.3. Virtual Thread

Virtual Thread, Java'nın Project Loom girişimi kapsamında geliştirilen hafif iş parçacıklarıdır. Sanal thread'ler, platform thread'lerin aksine, düşük bellek kullanımı ve hızlı başlatma süreleri ile dikkat çeker. Bu iş parçacıkları, yüksek iş yükü altında daha yüksek verimlilik ve daha düşük gecikme süresi sağlar. Bununla birlikte, sanal thread'lerin veri yapılarını yönetme ve oluşturma maliyetleri CPU-bağlı iş yüklerinde performans düşüşüne neden olabilir [6],[7].

### 2.4. Thread Mimarileri

Bilgisayar çekirdekleri çekirdek sayısı kadar işletim sistemi thread’i (OS Thread) kullanacak altyapıya sahiptir. Java platform threadleri ise oluşturulduklarında bir işletim sistemi thread’i ile birebir işlenecek mimaride tasarlanmıştır (Şekil-1). Bu tasarım kaynak kullanımı kısıtlarını beraberinde getirmektedir.



**Şekil-1** Platform Thread – OS Thread ve CPU İlişkisi

Platform threadleri kullanımı sırasında giriş/çıkış bloklama (Blockin I/O) işlemi ile karşılaşıldığında işlem tamamlanana kadar bu thread, OS thread’ini kullanmaya devam etmekte ve bunun negatif sonucu olarak performans kayıplarına neden olmaktadır.

Sanal (Virtual) thread mimarisi ise bu gereksiz bekleme ve performans kayıplarının önüne gererek, eş zamanlı programlama performansını artırmayı amaçlamaktadır [7].

Sana (Virtual) thread kullanım mimarisi platform threadleri gibi birebir OS threadleri ile eşlenerek çalışmamaktadır. Bunun yerine bir sanal thread gurubu, platform thread’i ile eşlenir platform threadleri üzerinde işlemlerini yapar. I/O blok ile karşılaşıldığı zaman bu sanal thread heap’de beklemeye alınır ve yerine bloklanan zamanda başka bir sanal thread işleme alınmaktadır [2] (Şekil-2). Kullanım performansını en fazla artıran yeteneği ise bu planlama yapısı gösterilmektedir.



**Şekil-2** Virtual Thread - Platform Thread - OS Thread ve CPU İlişkisi

Sanal threadler platform threadleri gibi çekirdek sayısına bağımlı olmadığı için çok sayıda üretilip kullanıma olanak sağlamaktadır.

## 3. YAPILAN ÇALIŞMALAR

### 3.1. Thread Performans Analizleri ve Karşılaştırmalı Değerlendirme

Teorik araştırması tamamlanan thread performans ölçümlerinin pratik uygulamasını gerçekleştirmek amacıyla, Java SpringBoot framework'ü kullanılarak kapsamlı bir test ortamı tasarlanmıştır. Geliştirilen uygulama kapsamında, hem I/O yoğun işlemler hem de CPU bağımlı operasyonlar için ayrı ayrı olmak üzere iki farklı thread türü (virtual ve platform thread) üzerinde çalışan RESTful web servis endpoint'leri implemente edilmiştir.

Performans ölçümlerinin nesnel bir şekilde yapılabilmesi için Apache JMeter test aracı kullanılarak çeşitli yük test senaryoları oluşturulmuş ve sistem davranışını görselleştiren detaylı performans grafikleri elde edilmiştir. Uygulamanın açık kaynak kodları, araştırmacıların erişimine sunulmak üzere GitHub platformunda paylaşılmıştır.

Proje Kaynak Kodu:

[SpringBoot Thread Performans Karşılaştırma Uygulaması](https://github.com/saozdemir/SpringBootWorkspace/tree/main/thread-performance)

### 3.2. Gerçek Hayat Senaryolarını Simüle Eden Uygulama Geliştirme

Thread performans analizlerini desteklemek amacıyla, gerçek dünya koşullarını yansıtan iki ayrı RESTful web servis mimarisi tasarlanmıştır. Birinci sistemde, personel yönetimi ve eğitim süreçlerini modelleyen; personel bilgileri, eğitim programları ve bu programlara ait ödevlerin sorgulanması gibi I/O yoğun işlemler içeren entegre bir veri modeli oluşturulmuştur. İkinci sistem ise, dağıtık mimariyi simüle etmek amacıyla personel görev yönetimini merkeze alan bağımsız bir servis olarak tasarlanmıştır.

Sistemler arası etkileşimi gerçekçi kılmak için:

* Her iki uygulama farklı veritabanları ve bağlantı portları üzerinde çalışacak şekilde yapılandırılmıştır
* Test verisi üreten otomatik metodlar geliştirilerek tutarlı test ortamı sağlanmıştır
* Dağıtık sistem davranışlarını ölçmeye yönelik özel sorgu senaryoları tasarlanmıştır
* Uygulama mimarilerinin kaynak kodları ve teknik dokümantasyon GitHub platformunda paylaşıma sunulmuştur.

Proje Kaynak Kodları:

[Personel Eğitim Yönetim Sistemi](https://github.com/saozdemir/SpringBootWorkspace/tree/main/personneleducation)

[Personel Görev Takip Sistemi](https://github.com/saozdemir/SpringBootWorkspace/tree/main/personneltask)

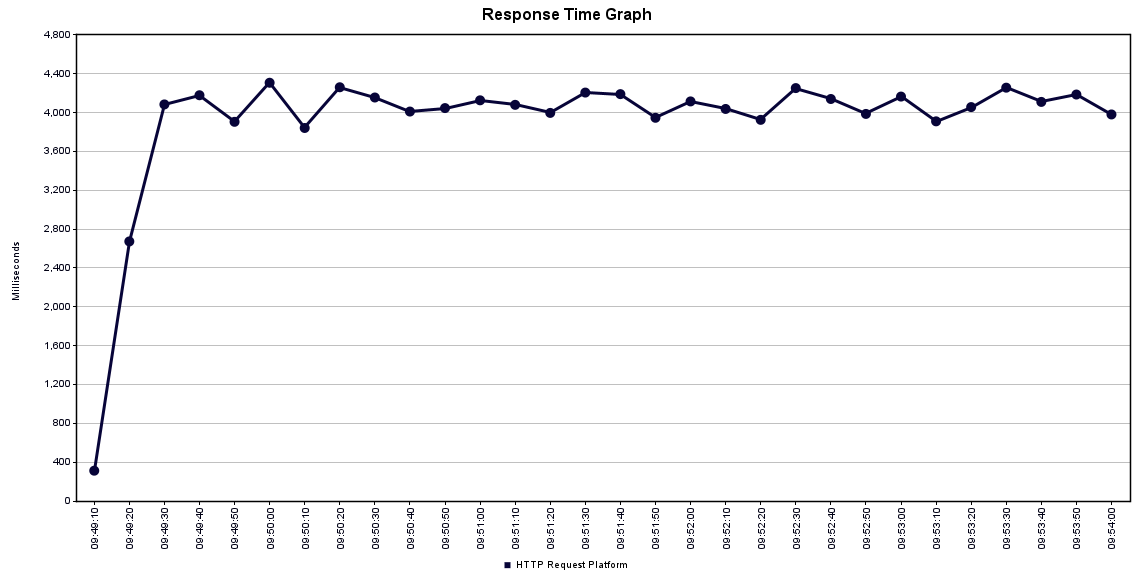
Bu kapsamlı test ortamı sayesinde, virtual thread'lerin gerçek dünya uygulamalarında I/O bağımlı işlemlerde nasıl bir performans avantajı sağladığı bilimsel verilerle ortaya konulması amaçlanmaktadır.

## 4. SONUÇLAR

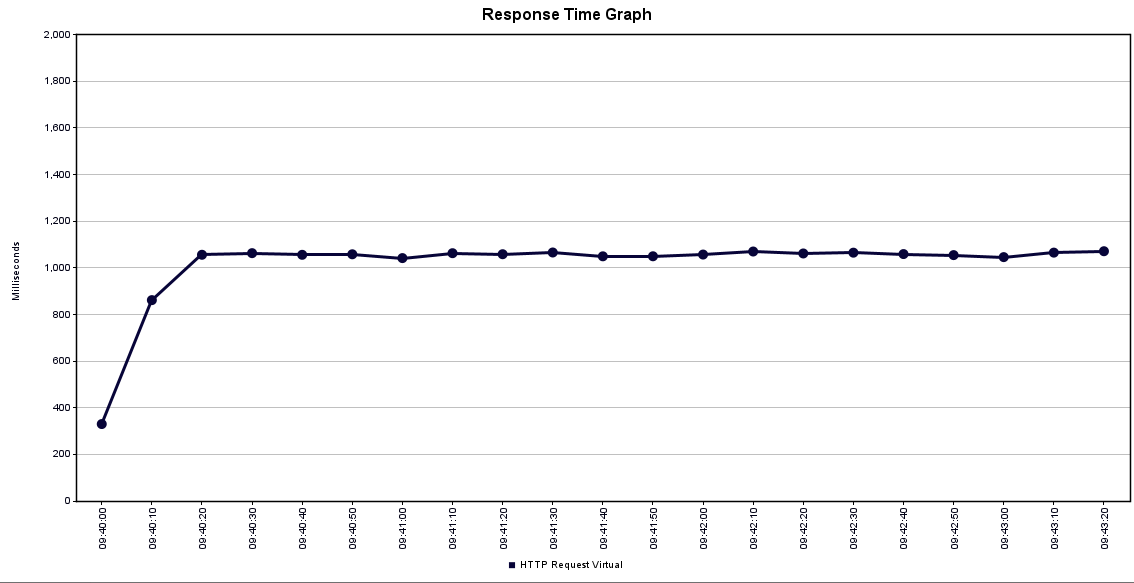
Tüm sonuçlar için ayrı ayrı RESTful servisler oluşturulmuş ve bu servisler için ayrı ayrı JMeter test senaryoları hazırlanmıştır. Senaryolar 1000 kullanıcı için 10. Saniyede maksimum istek oluşturacak şekilde 5dk süre ile ayrı ayrı test edilmiştir.

### 4.1. Thread Performans Analizleri ve Karşılaştırmalı Değerlendirme Sonuçları

İlk olarak I/O bağımlı işlemleri için virtual ve platform thread kullanımı ile alınan grafiksel sonuçlar Şekil-3 ve Şekil-4’te gösterilmiştir.



**Şekil-3** I/O Bağımlı İşlemlerde Platform Thread Kullanım Performansı



**Şekil-4** I/O Bağımlı İşlemlerde Virtual Thread Kullanım Performansı

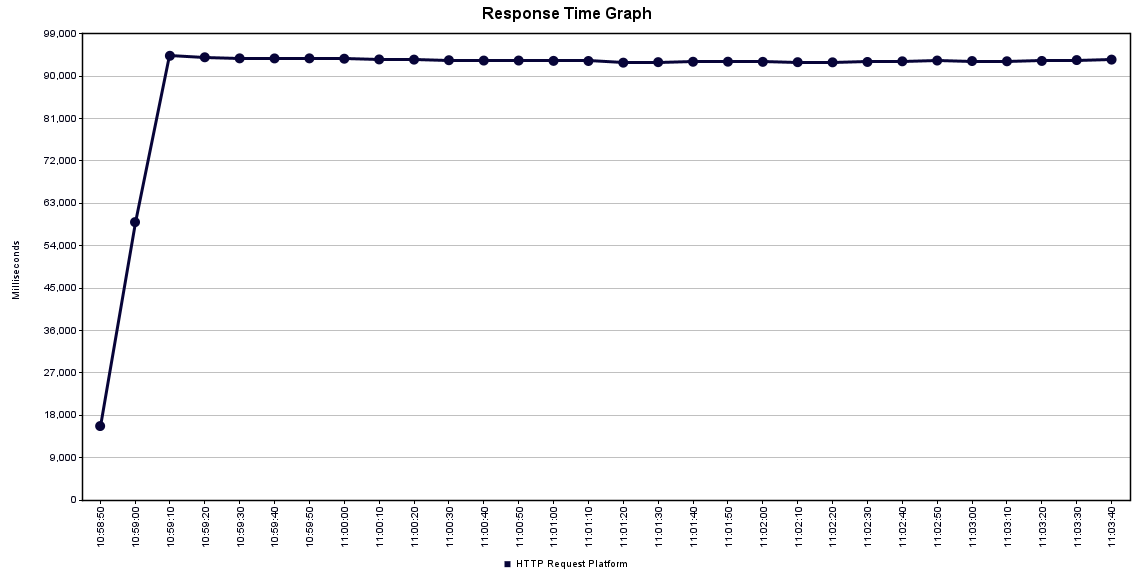
Bu performans grafikleri ışığında aşağıdaki tablo karşılaştırmalarını yaparak test sonuçlarının analizi tamamlanmıştır (Tablo-1).

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Platform Thread (I/O)** | **Virtual Thread (I/O)** | **Sonuç** |
| Gerçekleştirilen Toplam İstek | 74386 | 282231 | Virtual thread ile daha fazla kullanıcı/istek içeriyor. |
| Ortalama Yanıt Süresi (ms) | 3995 | 1048 | Ortalama yanıt süresi virtual thread kullanımında daha verimli. |
| En Kısa Yanıt Süresi (ms) | 201 | 200 | En kısa yanıt süresi iki teknoloji için de aynı. |
| En Uzun Yanıt Süresi (ms) | 7528 | 2032 | Virtual threadlerin en uzun yanıt süresi platform thread kullanımına göre daha verimli. |
| Standart Sapma | 2659,84 | 732,34 | Virtual thread teknolojisi daha istikrarlı çalışmaktadır. |
| Hata(%) | 0.0 | 0,0 |  |
| Saniyede Tamamlanan İstek Sayısı (s) | 244,5 | 936,8 | Virtual thread kullanılan sistem neredeyse 4 kat daha fazla işlem hacmine sahiptir. |

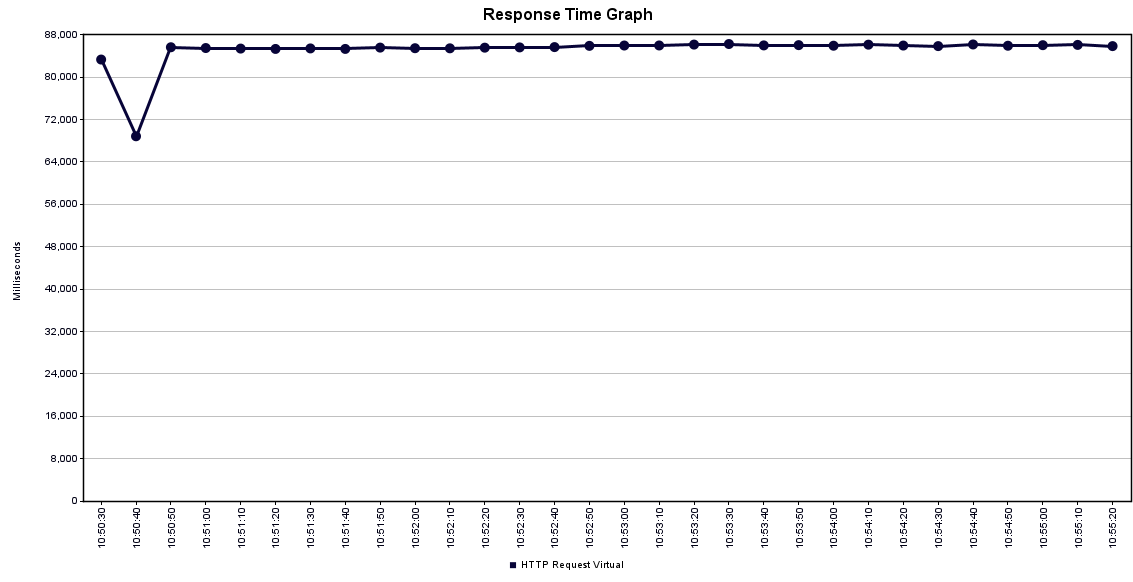
**Tablo-1** I/O Bağımlı İşlemlerde Virtual ve Platform Thread Kullanımının Performans Karşılaştırılması

Tablolardan da görüldüğü üzere I/O bağımlı işlemlerde virtual thread kullanımı daha verimli bir yöntem olduğu test ediliştir. Özellikle karşılanan istek, saniyede tamamlanan istek sayısı ve en uzun yanış süresinde virtual thread kullanımının verimli olduğu görülmektedir.

Ancak CPU bağımlı işlemler için aynı performans verileri elde edilememiştir. Bunun nedeni olarak virtual threadlerin işlem havuzundan bir platform threade atanmasıdır. CPU işlemi sırasında virtual thread platform thread gibi davranmaktadır.



**Şekil-5** CPU Bağımlı İşlemlerde Platform Thread Kullanım Performansı



**Şekil-6** CPU Bağımlı İşlemlerde Virtual Thread Kullanım Performansı

Şekil-5 ve Şekil-6’da da görüleceği üzere iki teknoloji de birbirine yakın grafiksel sonuçlar vermektedir.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Platform Thread (CPU)** | **Virtual Thread (CPU)** | **Sonuç** |
| Gerçekleştirilen Toplam İstek | 4170 | 4120 | Yakın sayıda istek gerçekleştirilmiş. |
| Ortalama Yanıt Süresi (ms) | 81910 | 83315 | Ortalama yanıt süresi Virtual Thread'te biraz daha yüksek; bu, CPU kaynaklarına daha yoğun ihtiyaç duyulan işlemlerde platform thread'lerin daha tutarlı olabileceğini göstermektedir. |
| En Kısa Yanıt Süresi (ms) | 878 | 1400 | Platform Thread daha kısa minimum yanıt süresi ile daha hızlı ilk tepkileri vermiştir. |
| En Uzun Yanıt Süresi (ms) | 95839 | 394025 | Virtual Thread'te önemli ölçüde daha uzun maksimum yanıt süresi gözlenmiş, bu da bazı thread’lerin uzun süre bloklandığını göstermektedir. |
| Standart Sapma | 23645,51 | 52659,49 | Virtual Thread yanıt sürelerinde çok daha fazla dalgalanma göstermiştir. |
| Hata(%) | 0.0 | 0,0 |  |
| Saniyede Tamamlanan İstek Sayısı (s) | 10,7 | 10,4 | Platform Thread biraz daha fazla istek tamamlamış, bu da CPU-bound işlemlerde onun daha verimli olduğunu desteklemektedir. |

**Tablo-2** CPU Bağımlı İşlemlerde Virtual ve Platform Thread Kullanımının Performans Karşılaştırılması

Yukardaki tablo sonuçlarına göre; CPU bağımlı işlemlerde Platform Thread, daha düşük ortalama ve maksimum yanıt süresi, daha az dalgalanma (standart sapma) ve daha yüksek istek tamamlama oranı ile daha stabil ve verimli bir performans sergilemiştir.

Virtual Thread yapısı ise CPU ağırlıklı görevlerde yüksek yanıt süresi ve dalgalanma göstererek, bu tür görevler için henüz platform thread’ler kadar uygun olmadığını ortaya koymuştur.

## 5. GELECEK ÇALIŞMALAR

Gelecek çalışmalarda gerçek hayatta kullanılacak senaryoya benzer test ortamında I/O bağımlı alanlar test verileri ile simüle edilerek thread performans analizlerindeki sonuçlar ile karşılaştırılması planlanmaktadır. Bu adımda sonuçlar elde edildikten sonra I/O bağımlı ve CPU bağımlı senaryoların birlikte kullanımına yönelik çalışmalara devam edilecektir.

## 6. KAYNAKLAR

[1] PrasadSrivastava, R., & CNandi, G. (2017). Controlling Multi Thread Execution using Single Thread Event Loop. 2017 International Conference on Innovations in Control, Communication and Information Systems (ICICCI). <https://doi.org/10.1109/iciccis.2017.8660809>

[2] KukYoon, M., Kim, K., Lee, S., WooRo, W., & Annavaram, M. (2016). Virtual Thread: Maximizing Thread-Level Parallelism beyond GPU Scheduling Limit. 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). <https://doi.org/10.1109/isca.2016.59>

[3] Beronić, D., Modrić, L., Mihaljević,B., Radovan, A. (2022). Comparison of Structured Concurrency Constructs in Java and Kotlin – Virtual Threads and Coroutines. 2022 45th Jubilee International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE. <https://doi.org/10.23919/MIPRO55190.2022.9803765>

[4] Navarro, A., Ponge, J., Mouël, F., & Escoffier,C. 2023. Considerations for integrating virtual threads in a Java framework: a Quarkus example in a resource-constrained environment. In Proceedings of the 17th ACM International Conference on Distributed and Event-based Systems (DEBS '23). Association for Computing Machinery, New York, NY, USA, 103–114. <https://doi.org/10.1145/3583678.3596895>

[5] Rosà, A., Basso, M., Bohnhoff, L., & Binder, W. (2023). Automated Runtime Transition between Virtual and Platform Threads in the Java Virtual Machine. 2023 30th Asia-Pacific Software Engineering Conference (APSEC). IEEE. <https://doi.org/10.1109/apsec60848.2023.00080>

[6] Pufek, P., Beronić, D., Mihaljević, B. & Radovan, A. (2020). Achieving Efficient Structured Concurrency through Lightweight Fibers in Java Virtual Machine. 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE. <https://doi.org/10.23919/MIPRO48935.2020.9245253>

[7] Beronić,D., Pufek, P., Mihaljević, B. & Radovan, A. (2021). On Analyzing Virtual Threads – a Structured Concurrency Model for Scalable Applications on the JVM. 2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE. <https://doi.org/10.23919/MIPRO52101.2021.9596855>