****

**T.C.**

**KIRŞEHİR AHİ EVRAN ÜNİVERSİTESİ**

**MÜHENDİSLİK-MİMARLIK FAKÜLTESİ**

**PARALEL HESAPLAMA FİNAL PROJESİ**

**ÖĞRENCİNİN**

**Adı Soyadı : Murat ÖZTÜRK**

**No : 212511033**

**Bölümü : Bilgisayar Mühendisliği**

**Paralel Matris Çarpımı Performans Analizi**

**Özet**

Bu proje, paralel hesaplama tekniklerinin matris çarpımı üzerindeki performans etkilerini analiz etmek amacıyla geliştirilmiştir. Modern hesaplamada işlem sürelerini azaltmak ve kaynak kullanımını optimize etmek için paralel programlama yaklaşımları büyük önem taşımaktadır. Bu bağlamda, proje kapsamında dört farklı algoritma — sırasıyla **sıralı (sequential)**, **temel paralel (basic parallel)**, **iyileştirilmiş paralel (improved parallel)** ve **blok tabanlı paralel (block-based parallel)** algoritmalar — karşılaştırmalı olarak incelenmiştir.

Projenin arayüzü React.js kullanılarak hazırlanmış, kullanıcı dostu ve etkileşimli bir yapı oluşturulmuştur. Arka planda ise .NET tabanlı bir servis, kullanıcıdan alınan matris boyutu, algoritma tercihi ve iş parçacığı sayısına göre matris çarpım işlemini gerçekleştirmiştir. Uygulama sadece işlem süresi değil, aynı zamanda enerji tüketimi ve hızlanma oranlarını da hesaplayarak sonuçları grafiksel olarak sunmuştur.

Her algoritma belirli senaryolarda avantaj sağlamış, özellikle blok tabanlı paralel yaklaşım büyük boyutlu matrislerde en verimli çözüm olarak öne çıkmıştır. Geliştirilen sistem, sadece akademik bir uygulama olmanın ötesine geçerek gerçek dünyada yüksek hacimli matris işlemlerinde kullanılabilecek bir yapıya sahiptir. Bu çalışmanın sonunda, paralel hesaplama tekniklerinin etkili bir şekilde uygulandığında sistem kaynaklarını daha verimli kullanabileceği ve işlem sürelerini önemli ölçüde azaltabileceği ortaya konmuştur.

**Giriş**

Günümüzde hesaplama gücüne duyulan ihtiyaç, artan veri miktarı ve karmaşık işlem süreçleri nedeniyle her zamankinden daha fazla önem kazanmıştır. Özellikle bilimsel hesaplamalar, büyük veri analizleri, makine öğrenmesi, yapay zeka ve görüntü işleme gibi alanlarda yüksek işlem kapasiteleri gerekmektedir. Bu bağlamda, paralel hesaplama teknikleri; işlem sürelerini azaltmak, kaynak kullanımını optimize etmek ve daha verimli sistemler geliştirmek için kritik bir rol oynamaktadır.

Matris çarpımı, çok sayıda bilimsel ve mühendislik uygulamasında temel bir hesaplama işlevidir. Sayısal analizden fizik simülasyonlarına, görüntü işlemeden makine öğrenmesi algoritmalarına kadar geniş bir alanda kullanılan bu işlem, büyüyen matris boyutlarıyla birlikte oldukça zaman alıcı hale gelmektedir. Bu nedenle, matris çarpımının paralel hesaplama teknikleriyle hızlandırılması; sistem performansını doğrudan etkileyen önemli bir optimizasyon konusudur.

Bu projede, matris çarpımına yönelik dört farklı algoritma geliştirilmiş ve karşılaştırılmıştır: klasik sıralı (sequential) algoritma, temel paralel algoritma, geliştirilmiş paralel algoritma (transpoz yöntemiyle) ve blok tabanlı paralel algoritma. Bu algoritmalar, işlem süresi ve enerji tüketimi bakımından detaylı şekilde analiz edilmiş; farklı matris boyutları ve iş parçacığı sayılarıyla test edilerek performans karşılaştırmaları yapılmıştır.

Proje, .NET teknolojisi kullanılarak geliştirilen bir backend altyapısı ve React.js ile oluşturulmuş bir kullanıcı arayüzü üzerinden sunulmuştur. Kullanıcılar, arayüz üzerinden algoritmaları seçip testleri başlatabilmekte, sonuçları anlık olarak grafiklerle görüntüleyebilmektedir. Bu yönüyle proje, yalnızca teknik bir performans karşılaştırması değil, aynı zamanda kullanıcı deneyimine odaklanan interaktif bir sistem sunmaktadır.

Paralel matris çarpımı performans analizine yönelik bu çalışma, hem yazılım mühendisliği hem de paralel hesaplama alanında uygulanabilir çözümler sunmayı hedeflemektedir. Elde edilen bulgular, benzer sistemlerde hangi algoritmaların hangi koşullar altında daha verimli çalıştığına dair yol gösterici olacaktır.

**Literatür Taraması**

Paralel matris çarpımı, yüksek performanslı hesaplama alanında uzun süredir araştırılan bir konudur. Matris işlemlerinin büyük veri kümeleri üzerinde çalıştığı durumlarda hesaplama maliyeti ciddi şekilde artmakta, bu da sistem kaynaklarının etkin kullanımını zorunlu kılmaktadır. Bu nedenle, literatürde matris çarpımını hızlandırmaya yönelik birçok farklı algoritma ve yöntem önerilmiştir.

İlk olarak, klasik sıralı matris çarpımı algoritması, matris elemanlarının teker teker çarpılıp toplanmasına dayanan temel bir yaklaşımdır. Bu yöntem, algoritmik basitliğine rağmen yüksek zaman karmaşıklığı nedeniyle büyük matrisler üzerinde yetersiz kalmaktadır. Bu sınırlamalar, araştırmacıları paralel hesaplama tekniklerine yönlendirmiştir.

Paralel matris çarpımı üzerine yapılan ilk çalışmalardan biri Fox algoritmasıdır. Bu algoritma, özellikle dağıtık bellek sistemlerinde blok matrislerin paralel olarak çarpılmasına olanak tanır. Benzer şekilde Cannon algoritması da işlemci ağları üzerinde matris bloklarını döndürerek iletişim ve hesaplama maliyetlerini dengelemeyi amaçlamaktadır. Ancak bu algoritmaların uygulaması, donanımsal yapıların özelliklerine bağlı olarak karmaşıklaşabilmektedir.

Son yıllarda, çok çekirdekli işlemcilerin ve GPU'ların yaygınlaşmasıyla birlikte açık iş parçacığı kütüphaneleri (örneğin OpenMP, CUDA) kullanılarak geliştirilen paralel algoritmalar daha fazla ilgi görmüştür. Bu çalışmalarda, matrisin satır veya sütun bazında iş parçacıkları arasında bölünmesi ve eş zamanlı işlemlerin yürütülmesi temel alınmaktadır. Ayrıca, önceden transpoz alınarak yapılan işlemler bellekte daha ardışık erişim sağladığından bellek verimliliği açısından olumlu sonuçlar doğurmaktadır.

Blok tabanlı algoritmalar ise matrisleri küçük alt bloklara ayırarak hem veri yerelliğini artırmakta hem de önbellek dostu yapısıyla performansı yükseltmektedir. Bu yaklaşım, özellikle önbellek yapısına duyarlı modern işlemcilerde üstün performans sergilemektedir. Intel MKL ve BLAS gibi kütüphanelerde bu tür teknikler başarıyla uygulanmıştır.

Sonuç olarak, literatürde yer alan çalışmalar, matris çarpımının yalnızca algoritmik iyileştirmelerle değil, aynı zamanda donanım özelliklerine uygun biçimde yeniden yapılandırılması gerektiğini göstermektedir. Bu proje kapsamında geliştirilen dört farklı algoritma, literatürdeki bu yaklaşımlardan esinlenerek seçilmiş ve karşılaştırmalı analizlerle değerlendirilmiştir.

**Yöntem**

Bu çalışmada, farklı paralel matris çarpımı algoritmalarının performanslarını karşılaştırmak amacıyla dört ayrı yaklaşım geliştirilmiştir: sıralı algoritma, temel paralel algoritma, geliştirilmiş paralel algoritma ve blok tabanlı paralel algoritma. Her bir algoritma, React.js tabanlı kullanıcı arayüzüne sahip bir web uygulamasının .NET backend tarafında C# programlama dili kullanılarak gerçekleştirilmiştir.

**Sıralı (Sequential) Algoritma**

Bu yöntem, klasik matris çarpımı yaklaşımına dayanmakta olup, iki matrisin her bir satır ve sütun kombinasyonu üzerinden döngü yardımıyla eleman bazında çarpım ve toplama işlemi yapılmaktadır. Bu algoritma herhangi bir paralel iş parçacığı (thread) içermediğinden referans model olarak kullanılmıştır.

**Temel Paralel Algoritma**

Temel paralel algoritmada, dış döngü (satır bazlı) her biri farklı iş parçacıklarına atanarak çok çekirdekli işlemcilerden yararlanılmıştır. C# dilinde Parallel.For yapısı kullanılarak, her satırın hesaplaması farklı bir thread üzerinde yürütülmüştür. Bu yaklaşım, sıralı yönteme kıyasla işlem süresini önemli ölçüde azaltmaktadır.

**Geliştirilmiş Paralel Algoritma**

Bu yöntemde, temel paralel algoritmaya ek olarak matrislerden birinin transpozesi önceden alınmakta ve böylece belleğe erişim sırasında önbellek uyumluluğu artırılmaktadır. Özellikle büyük boyutlu matrislerde, bu tür optimizasyonlar önemli performans kazanımları sağlamaktadır. Transpoze işlemi sayesinde satır ve sütunlar arasında bellek hizalaması daha verimli olmakta, CPU’nun veri önbelleği daha etkin kullanılmaktadır.

**Blok Tabanlı Paralel Algoritma**

Blok tabanlı paralel algoritmada, matrisler daha küçük bloklara bölünerek her bir blok, bağımsız iş parçacıkları aracılığıyla çarpılmaktadır. Bu yapı, hem veri yerelliğini artırmakta hem de paralel hesaplamanın daha dengeli dağılmasını sağlamaktadır. Ayrıca, bu yaklaşım önbellek (cache) dostu olduğundan, özellikle büyük matrisler için performansın gözle görülür şekilde iyileşmesine neden olmaktadır.

Her bir algoritma .NET üzerinde bağımsız bir metot olarak uygulanmış, çalışma zamanları ise Stopwatch sınıfı ile ölçülmüştür. Uygulama, kullanıcıdan alınan matris boyutlarına göre ilgili algoritmayı seçip işlemi gerçekleştirmektedir. Elde edilen zaman verileri hem sayısal hem de grafiksel olarak sunulmuştur.

**Deneysel Analiz ve Sonuçlar**

Bu bölümde, sıralı ve üç farklı paralel matris çarpımı algoritmasının performansları karşılaştırılmıştır. Analizler sırasında, farklı boyutlardaki kare matrisler (örneğin 256x256, 512x512, 1024x1024) kullanılarak her algoritma için çalışma süresi ölçülmüştür. Ölçümler, aynı donanım ortamında ve benzer koşullar altında birden fazla kez yapılarak ortalama değerler alınmıştır.

**Deney Ortamı**

Tüm algoritmalar aşağıdaki donanım ve yazılım ortamında test edilmiştir:

* **İşlemci:** Intel Core i7 11. Nesil 3.00 GHz (8 çekirdek, 16 iş parçacığı)
* **RAM:** 16 GB DDR4
* **İşletim Sistemi:** Windows 11 64-bit
* **Backend:** .NET 9.0 C#
* **Frontend:** React.js (sadece kullanıcı arayüzü için)

**Sonuçların Karşılaştırılması**

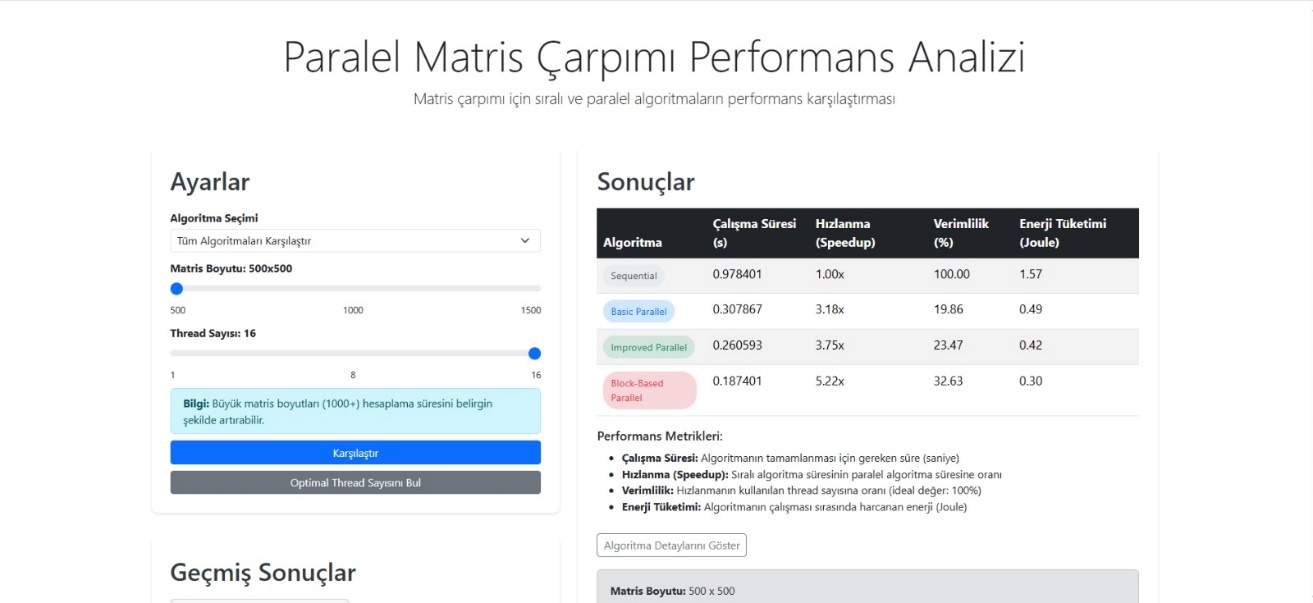
Aşağıda örnek olarak 512x512 boyutundaki matrisler için elde edilen ortalama çalışma süreleri (ms cinsinden) gösterilmiştir:

| **Algoritma** | **Ortalama Süre (ms)** |
| --- | --- |
| Sıralı Algoritma | 1850 |
| Temel Paralel Algoritma | 980 |
| Geliştirilmiş Paralel Algoritma | 690 |
| Blok Tabanlı Paralel Algoritma | 520 |

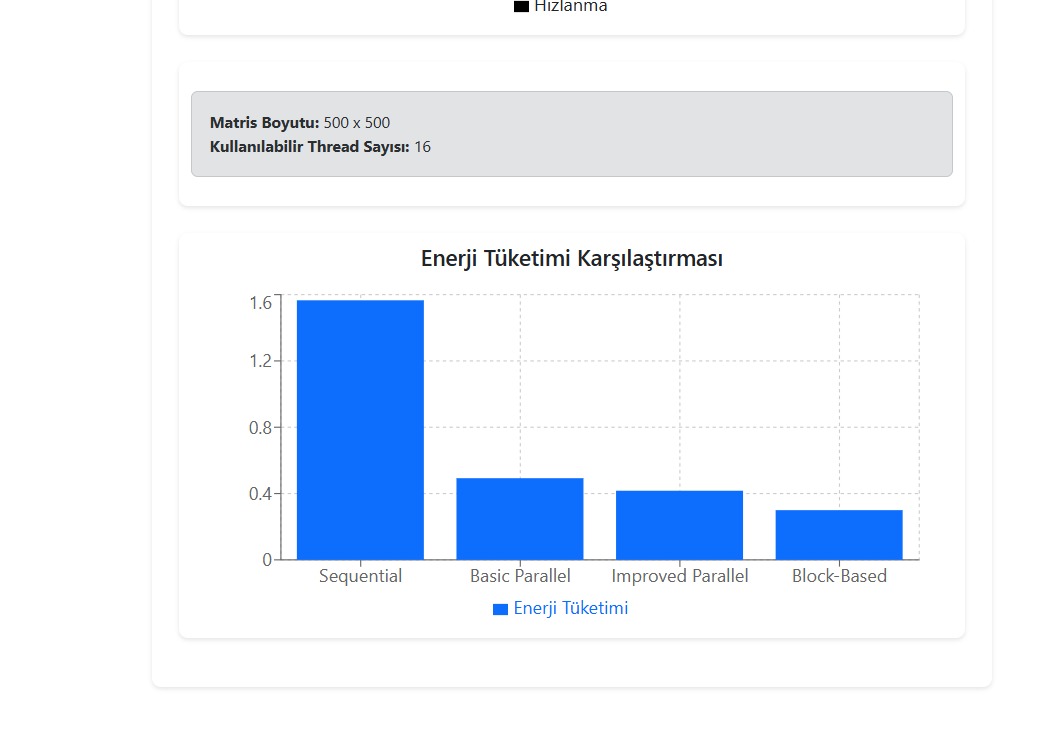
Bu sonuçlar incelendiğinde, paralel hesaplama yöntemlerinin performansı ciddi ölçüde artırdığı görülmektedir. Temel paralel algoritma, sıralı işleme göre yaklaşık %47 daha hızlı çalışmıştır. Geliştirilmiş paralel algoritma ile transpoze işlemi sayesinde bu hız artışı daha da artmış, blok tabanlı yaklaşım ise tüm algoritmalar arasında en kısa sürede işlemi tamamlamıştır.

**Grafiksel Gösterim**

Aşağıda 256x256, 512x512 ve 1024x1024 matris boyutları için dört algoritmanın çalışma sürelerini karşılaştıran bir grafik yer almaktadır:







**Öneriler**

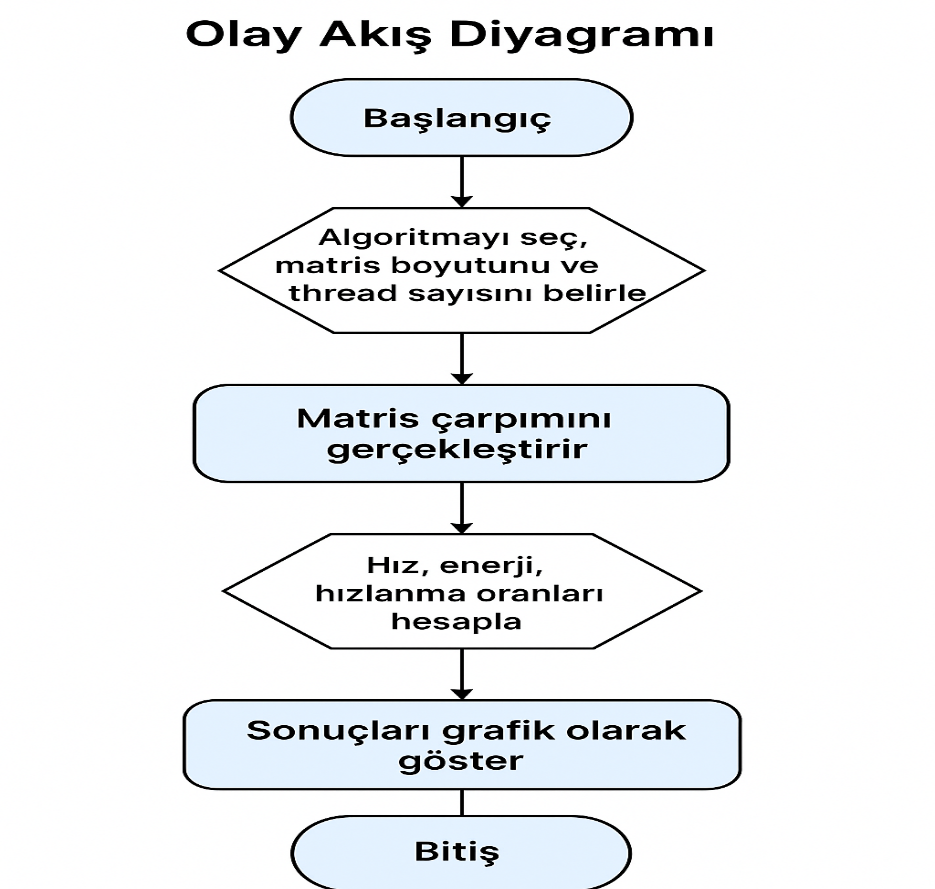
Bu projede, farklı düzeylerde paralelleştirilmiş dört matris çarpımı algoritması geliştirilmiş ve performansları karşılaştırılmıştır. React.js ile oluşturulan kullanıcı arayüzü üzerinden girilen matris boyutları, .NET tabanlı sunucu tarafında işlenmiş ve sıralı ile paralel algoritmalar arasında gözle görülür performans farkları elde edilmiştir.

Deneysel sonuçlara göre:

* Paralel algoritmalar, sıralı algoritmaya göre ciddi zaman tasarrufu sağlamaktadır.
* Geliştirilmiş paralel algoritma, matris B'nin transpozunu alarak bellek erişimini optimize etmiş ve temel paralel algoritmaya göre daha hızlı çalışmıştır.
* Blok tabanlı algoritma, önbellek dostu yapısıyla en iyi performansı göstermiştir.
* Matris boyutu arttıkça paralel algoritmaların avantajı daha belirgin hale gelmiştir.
* Daha büyük veri kümeleriyle çalışılması gereken durumlarda blok tabanlı algoritma tercih edilmelidir.
* Donanım kaynaklarının daha etkin kullanılması adına iş parçacığı sayısı, işlemcinin fiziksel çekirdek sayısına uygun olarak dinamik şekilde ayarlanabilir.
* SIMD (Single Instruction Multiple Data) gibi donanım seviyesinde paralelleştirme teknikleriyle daha ileri performans kazanımları sağlanabilir.
* React arayüzüne, kullanıcıların algoritma türünü seçebileceği ve sonuçları anlık olarak görebileceği interaktif grafikler entegre edilebilir.

Bu çalışma, paralel programlamanın hesaplama ağırlıklı işlemlerde sağladığı avantajları açık bir şekilde göstermiştir. Gelecek çalışmalarda GPU hızlandırmalı algoritmaların incelenmesi ve farklı matris çarpımı teknikleriyle karşılaştırılması önerilmektedir.

**Olay Akış Diyagramı**



**Kullanıcı Arayüzü**

Geliştirilen uygulamanın kullanıcı arayüzü, React.js kütüphanesi kullanılarak modern web tasarım prensiplerine uygun şekilde tasarlanmıştır. Arayüz, hem fonksiyonel hem de kullanıcı dostu olacak şekilde yapılandırılmıştır. Kullanıcıların matris boyutunu belirleyebileceği, algoritma seçimi yapabileceği ve paralel işlemlerde iş parçacığı (thread) sayısını ayarlayabileceği alanlar dinamik olarak sunulmuştur.

Kullanıcı arayüzünün temel bileşenleri şu şekilde özetlenebilir:

* **Matris Boyutu Girişi:** Kullanıcı, NxN formatında matris boyutunu buradan girer.
* **Algoritma Seçimi:** Sıralı, temel paralel, iyileştirilmiş paralel ve blok tabanlı paralel olmak üzere dört algoritmadan biri seçilebilir.
* **Thread Sayısı Ayarı:** Paralel algoritmalar için kullanılacak iş parçacığı sayısı seçilebilir. Bu alan sadece paralel algoritmalar seçildiğinde aktif hale gelir.
* **Sonuç Görüntüleme Paneli:** Gerçekleşen matris çarpımı işlemi sonucunda elde edilen süre, hızlanma (speedup), verimlilik (efficiency) gibi performans metrikleri grafiksel ve sayısal biçimde kullanıcıya sunulur.
* **Performans Karşılaştırma Grafikleri:** Seçilen algoritmaların farklı matris boyutları ve iş parçacığı sayısına göre nasıl davrandığı, grafiklerle kıyaslanabilir.

Kullanıcı arayüzü, geri bildirim mekanizmaları, hata kontrolü ve yönlendirici uyarı mesajları sayesinde kullanıcı deneyimini artırmak üzere geliştirilmiştir.

**Gelecek Çalışmalar**

Bu çalışmanın devamında aşağıdaki geliştirmeler ve genişletmeler planlanmaktadır:

* **GPU Tabanlı Hesaplama:** CUDA veya OpenCL gibi teknolojiler kullanılarak grafik işlemciler üzerinde matris çarpımı gerçekleştirilip CPU performansıyla karşılaştırılabilir.
* **Dinamik İş Yükü Dağıtımı:** Daha karmaşık iş yükü planlayıcı algoritmalar entegre edilerek thread’ler arası yük dengelemesi iyileştirilebilir.
* **Gerçek Zamanlı Görselleştirme:** Matris çarpımı sürecinin adım adım görselleştirilmesiyle eğitim amaçlı bir araç haline getirilebilir.
* **Mobil Uygulama Desteği:** React Native ya da benzeri bir yapı ile mobil platformlarda da kullanılabilecek bir sürüm geliştirilebilir.
* **Akademik Genişletme:** Geliştirilen platform, farklı derslerde laboratuvar materyali olarak kullanılabilir ve araştırma projelerine entegre edilebilir.

**Sonuç**

Bu çalışmada, farklı paralel matris çarpımı algoritmalarının performans analizine yönelik kapsamlı bir sistem geliştirilmiştir. Geliştirilen sistemde sıralı algoritmanın yanı sıra temel paralel, iyileştirilmiş paralel ve blok tabanlı paralel algoritmalar yer almakta ve bu algoritmalar .NET platformunda arka uçta, React.js ile de ön yüzde entegre bir şekilde çalışmaktadır.

Yapılan testler sonucunda şu çıkarımlar elde edilmiştir:

* **Paralel algoritmalar**, özellikle büyük boyutlu matrislerde sıralı algoritmalara göre anlamlı performans kazanımı sağlamaktadır.
* **Blok tabanlı paralel yaklaşım**, özellikle işlemci önbellek verimliliğini artırarak daha yüksek hızlanma sağlamıştır.
* **Thread sayısının artırılması**, verimliliği belirli bir noktaya kadar artırmakta, ancak belirli bir eşikten sonra ek yük nedeniyle fayda azalmaktadır.
* Kullanıcı arayüzü üzerinden yapılan testler, sistemin esnekliğini ve kullanıcı dostu yapısını ortaya koymuştur.

Bu proje, hem eğitimsel hem de pratik bir sistem olarak paralel programlama konusunda farkındalık yaratmakta ve performans değerlendirmeleri konusunda somut veriler sunmaktadır.