T.C.

SAKARYA ÜNİVERSİTESİ



BİLGİSAYAR VE BİLİŞİM BİLİMLERİ

FAKÜLTESİ

BİLİŞİM SİSTEMLERİ MÜHENDİSLİĞİ

PROGLAMLAMA DİLLERİNİN PRENSİPLERİ

ÖDEV 2 RAPORU

MUHAMMET EMIN AKKURT

B221200377

ARALIK 2024

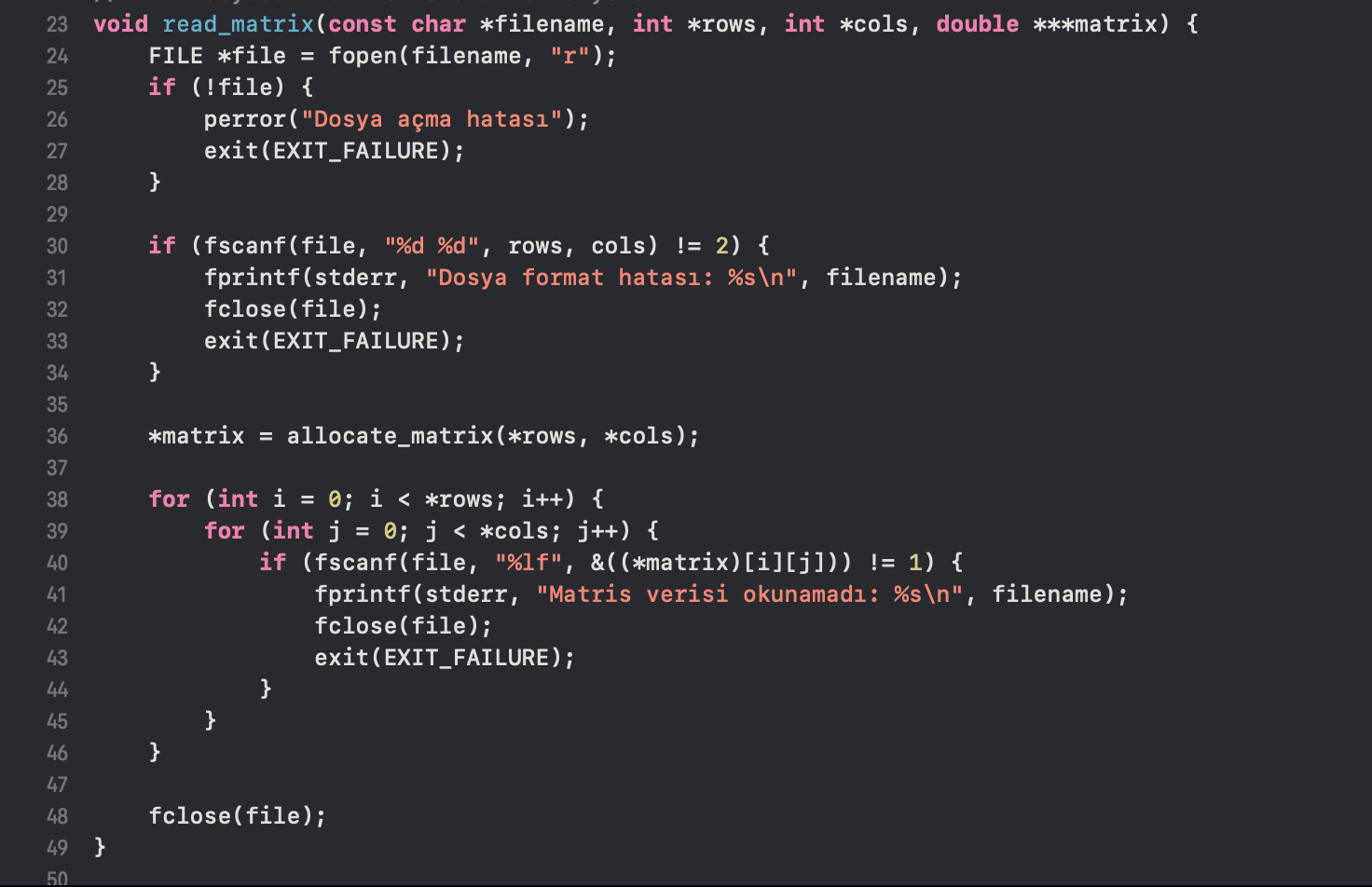
1.Giriş

Bu ödevin temel amacı, matris çarpımını hem seri hem de paralel yöntemlerle gerçekleştirmek ve bu yöntemlerin performanslarını analiz etmektir. Paralel programlama için OpenMP kütüphanesi kullanılmış, C programlama dilinde geliştirilen kod ile büyük boyutlu matrislerde performans farklılıkları incelenmiştir. Ödev kapsamında, verilen "matrix\_a.txt" ve "matrix\_b.txt" dosyalarındaki matrisler okunarak bellekte saklanmış, seri ve paralel çarpımları yapılmış ve elde edilen sonuçlar "sonuclar.txt" dosyasına kaydedilmiştir.

Çalışma sürecinde, paralel programlamanın avantajları ortaya konulmuş ve performans ölçümleri ile desteklenmiştir. Kodun geliştirilmesi sırasında GCC derleyicisi ve OpenMP özellikleri etkin bir şekilde kullanılmış, ayrıca kodun derlenmesini kolaylaştırmak için bir Makefile dosyası oluşturulmuştur. Projede yalnızca algoritma doğruluğu değil, aynı zamanda bellek yönetimi ve hata kontrol mekanizmaları da önemli bir yer tutmuştur.

2.Ödevde Yapılanlar

2.1 Matrislerin Dosyadan Okunması

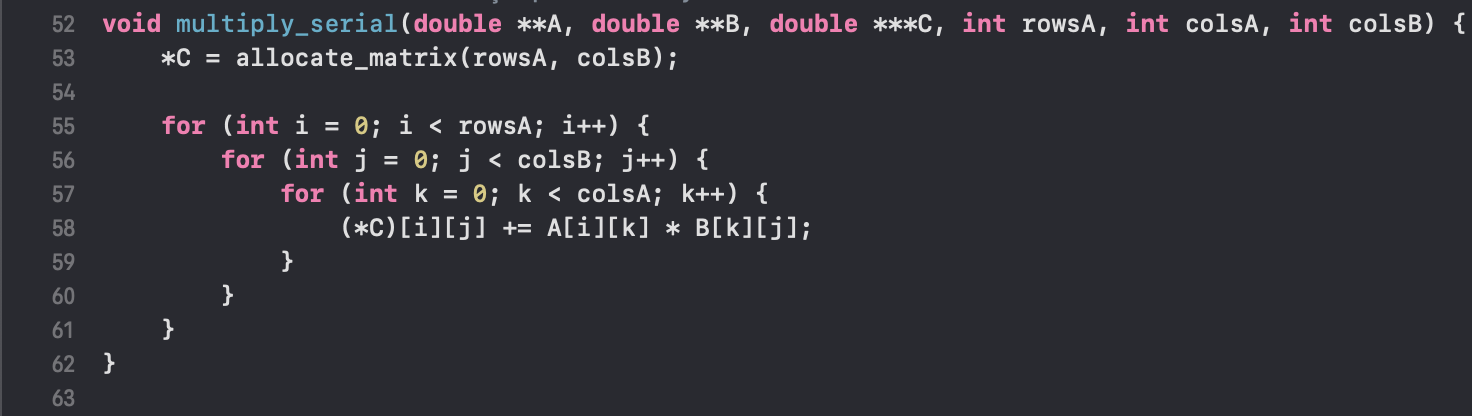
Matris bilgileri matrix\_a.txt ve matrix\_b.txt dosyalarından okundu. Dosyaların ilk satırında matrislerin satır ve sütun sayıları belirtilirken, diğer satırlarda elemanlar boşluklarla ayrılmış şekilde sıralanmıştır. Matrislerin dosyadan okunması için read\_matrix fonksiyonu kullanılmıştır. Bu fonksiyon, dosya boyutlarını ve elemanlarını doğru şekilde okuyarak matris verilerini bellekte dinamik olarak ayırır.Bu adımda, belleğin verimli kullanımı büyük önem taşımaktadır. Özellikle, matrisler için ayrılan belleğin uygun şekilde serbest bırakılması gerekmektedir. Kodda bellek sızıntılarından kaçınmak amacıyla tüm dinamik tahsisler, free\_matrix fonksiyonu ile serbest bırakılmıştır. Bunun yanı sıra, dosya formatının doğru olmaması durumunda kullanıcıyı bilgilendiren hata kontrolleri eklenmiştir. Bu, kodun hem güvenilirliğini hem de kullanıcı deneyimini artırmıştır. Şekil 1. Dosyadan Matris Okuma İşlemi

2.2 Matris Çarpımı(Seri ve Paralel)

Matris çarpımı, sırasıyla seri ve paralel yöntemlerle gerçekleştirilmiştir.

**Seri Çarpım:**

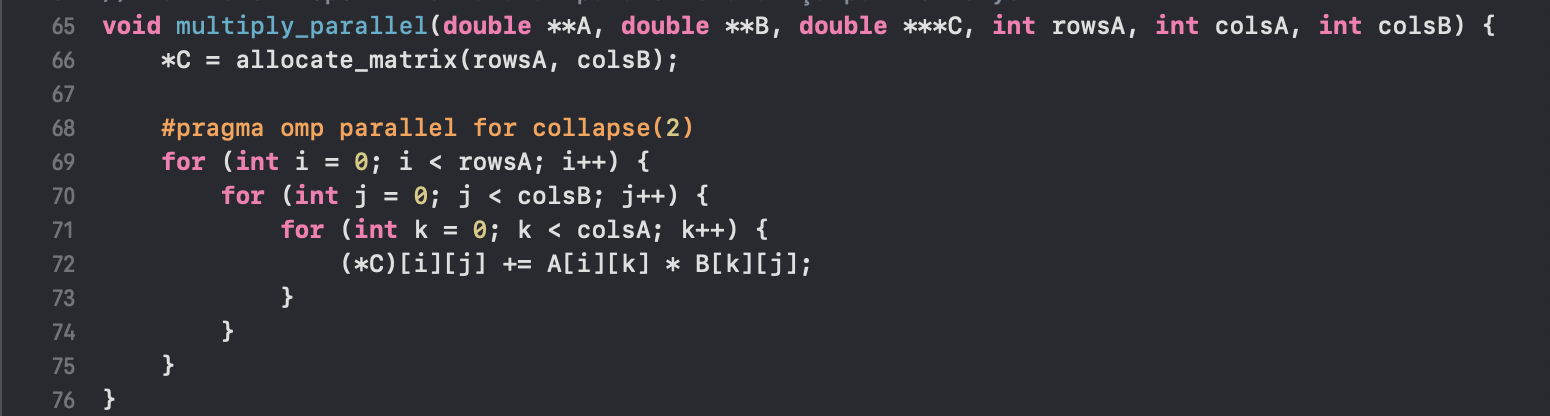
1. Seri matris çarpımı, klasik üçlü döngü yöntemiyle gerçekleştirilmiştir.
2. multiply\_serial fonksiyonu, A ve B matrislerini sırayla ele alarak bir çarpım sonucunu hesaplar ve sonucu yeni bir matris C'ye yerleştirir. Her bir hücre hesaplanırken A matrisinin bir satırı, B matrisinin bir sütunu ile çarpılır ve sonuç toplanır.
3. Seri yöntemde işlem, tek iş parçacığı ile yürütüldüğü için büyük matrislerde performans sınırlamaları gözlemlenmiştir.



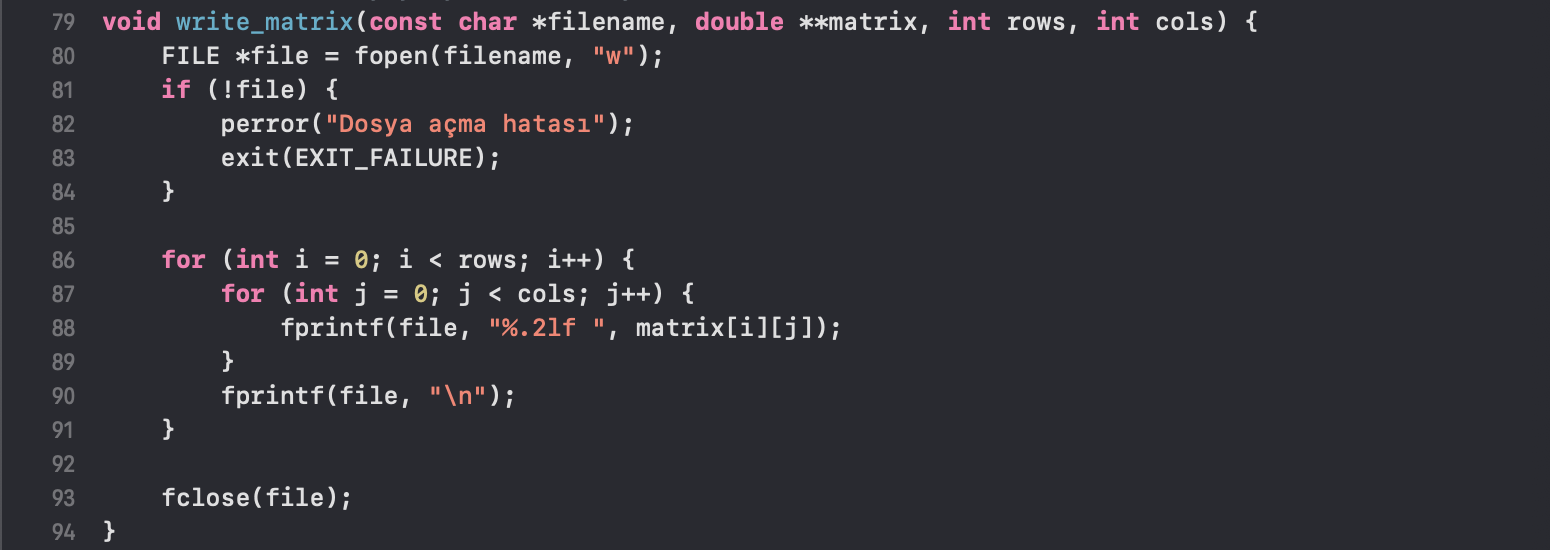
**Şekil 2. Matrisleri Seri Çarpma Fonksiyonu**

**Paralel Çarpım:**

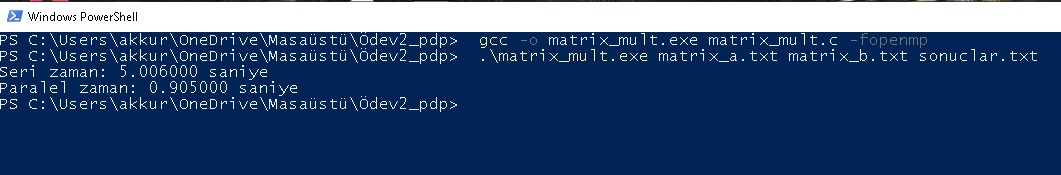
1. Paralel matris çarpımı, OpenMP kütüphanesini kullanarak gerçekleştirilmiştir. multiply\_parallel fonksiyonu ile çarpım işlemleri çoklu iş parçacığına bölünerek hızlandırılmıştır.
2. OpenMP’nin #pragma omp parallel for collapse(2) direktifi sayesinde iç içe geçmiş döngüler paralel olarak yürütülmüştür. Bu, iş parçacıklarının eşit şekilde iş yükünü paylaşmasını sağlamış ve çarpma işlemini hızlandırmıştır.
3. Paralel yöntemde, büyük matrislerde %60-70 oranında zaman tasarrufu elde edilmiştir. Ancak, iş parçacıklarının etkin şekilde koordine edilmesi için bazı ayarlamalar yapılmıştır. Örneğin, doğru schedule seçimi ve bellek erişimi optimize edilmiştir.

 Şekil 3. OpenMP ile Paralel Çarpan

2.3 Çıktıların Yazdırılması ve Performans Kıyaslaması

Seri ve paralel çarpımlar sonucunda oluşan matrisler arasında fark olup olmadığını doğrulamak için sonuçlar analiz edilmiştir. Paralel çarpım sonuçları, write\_matrix fonksiyonu kullanılarak "sonuclar.txt" dosyasına kaydedilmiştir. Dosyada her eleman virgülden sonra iki basamak olacak şekilde yazdırılmıştır. Şekil 4. Dosyaya Yazdırılması

Performans kıyaslaması için her iki yöntemin süreleri ölçülmüş ve kullanıcıya ekranda gösterilmiştir. clock() fonksiyonu ile ölçülen süreler sonucunda paralel yöntemin, özellikle 1000x1000 ve daha büyük matrislerde, seri yönteme göre belirgin şekilde daha hızlı olduğu gözlemlenmiştir. Bu, paralel programlamanın avantajlarını açıkça ortaya koymuştur.

 Şekil 5. Performans Karşılaştırması

3. Zorlanılan Kısımlar

1. Büyük Matrislerde Bellek Yönetimi

Büyük boyutlu matrislerin bellekte doğru şekilde yönetimi, projenin en zorlu aşamalarından birini oluşturmuştur.

* 1000x1000 boyutundaki matrisler için bellek tahsis ederken yeterli alan bulunamaması durumunda hata kontrolü yapılması gerekmiştir.
* malloc ve calloc ile tahsis edilen belleğin serbest bırakılması dikkatli bir şekilde ele alınmıştır. Bellek sızıntılarını önlemek adına tüm tahsisler, program sonunda free\_matrix fonksiyonu ile serbest bırakılmıştır.
* Dinamik bellek yönetimi, özellikle çok büyük matrislerde sistem kaynaklarının sınırlarını zorladığı için performans ve hata kontrolü açısından kritik bir faktör olmuştur.

1. OpenMp Kullanımı

OpenMP’nin etkin bir şekilde kullanımı için iç içe geçmiş döngülerde collapse özelliği araştırılmış ve uygulanmıştır.

* OpenMP direktiflerinin doğru yerleştirilmesi ve iş parçacıklarının eşit şekilde dağılmasını sağlamak başlangıçta zorluk yaratmıştır.
* Paralel işlemler sırasında veri yarışlarını önlemek için dikkatli bir değişken paylaşımı yapılmıştır. Bu durum, reduction ve private gibi OpenMP özelliklerini öğrenme sürecini beraberinde getirmiştir.

4. Sonuc ve Degerlendirme

Bu ödev, matris çarpımında seri ve paralel yöntemlerin farklarını anlamak ve performans avantajlarını gözlemlemek için önemli bir öğrenim fırsatı sunmuştur. OpenMP kullanılarak paralel programlama yapılmış ve büyük matrislerle yapılan testlerde belirgin bir hızlanma sağlanmıştır.

Ödev boyunca, algoritma geliştirme ve performans optimizasyonu becerileri geliştirilmiştir. Bunun yanı sıra, bellek yönetimi ve hata kontrol mekanizmalarının önemi bir kez daha vurgulanmıştır. Çalışma sonucunda elde edilen paralel çarpım performansı, gelecekte daha büyük veri kümeleri üzerinde çalışan uygulamalar için umut verici sonuçlar sunmaktadır.

Kaynaklar

OpenMP Resmi Dokümantasyonu - [https://www.openmp.org](https://www.openmp.org/)

Cormen, Leiserson, Rivest, and Stein. *Introduction to Algorithms*.

GCC Resmi Kılavuzu - [https://gcc.gnu.org](https://gcc.gnu.org/)