İSTİNYE ÜNİVERSİTESİ

MÜHENDİSLİK VE DOĞA BİLİMLERİ FAKÜLTESİ

YAZILIM MÜHENDİSLİĞİ BÖLÜMÜ

YAZ492: BİTİRME TEZİ 2

İLK RAPOR

Nisan 2022

PROJE BAŞLIĞI

Paralel Mimarilerde Seyrek Alt Üçgen Matris Çözümü için
Graf Parçalaması ile Yük Dengelemesi

PROJE YAZARI
Abdülkadir Furkan Yıldız - 190701145

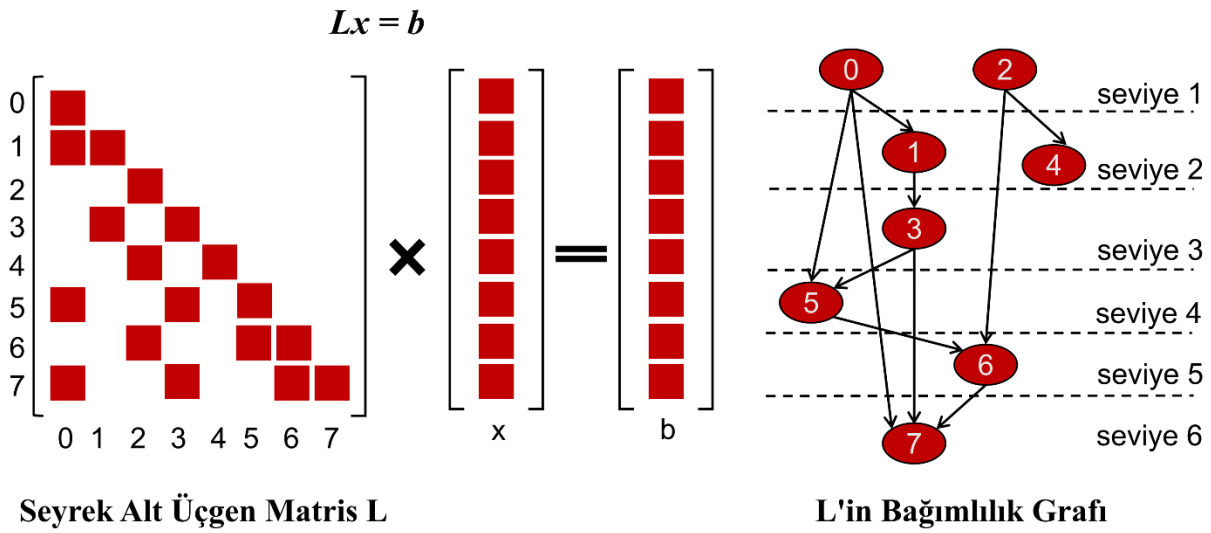
DANIŞMAN
Dr. Öğr. Üyesi Buse Yılmaz

ÖZET

Chainbreaker paralel mimarilerde Seyrek Alt Uçgen Çözümü'nü optimize eden bir çerçevedir. Çerçevenin önemli bir modül yapılacak bağımlılık grafı dönüşümleri için stratejiler bütünü ve stratejilerin uygulanacağı seviyeleri belirleyen strateji koleksiyonu ve strateji seçme modülüdür. Projenin amacı, grafı aşağıdaki hedeflere ulaşılacağı şekilde dönüştürerek, seviyeler arasında ve içinde yük dengelemesini sağlayacak stratejileri ve uygulanacakları seviyeleri belirlemektir. Bu çalışmada, bir seyrek alt üçgen matrisin sınırlı paralellik gösteren parçalarının paralellik derecesini artırmak için bağımlılık grafını dönüştüren Chainbreaker çerçevesi için bir strateji koleksiyonu geliştirilecektir.

- Düşük paralellik derecesine sahip seyrek alt üçgen matris parçalarının paralelliğini graf dönüşümü ile artırarak, seyreklik yapısını daha homojen hale getirmek.

- Senkronizasyon noktalarına olan ihtiyacı azaltmak



Şekil 1. Seyrek Alt Üçgen Matrislerin Lx = b

İÇİNDEKİLER

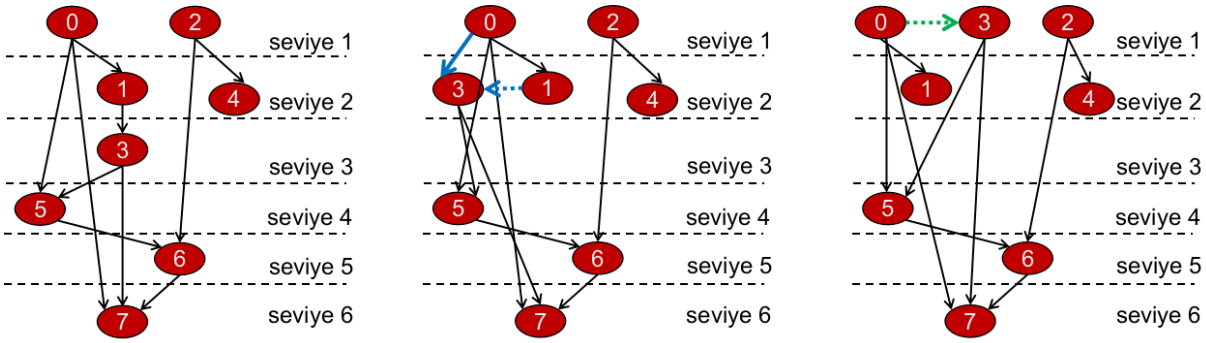
TABLO LİSTESİ

FİGÜR LİSTESİ

GİRİŞ

Bu çalışmanın amacı Chainbreaker çerçevesinin efektif olarak Seyrek Alt Üçgen Çözümü için optimizasyon yapabilmesine olanak sağlayacaktır. Bunun için matrisin bağımlılık grafının mimarinin çekirdeklerine dengeli biçimde dağıtılması gerekmektedir.

Bu çalışma bağımlılık grafı üzerinde ki dönüşümleri modelleyerek bu dengelemeyi yapacaktır. Graf dönüşümü göz önüne alındığında, Chainbreaker'ın önerdiği yeniden yazma yöntemi ile matrisin paralellik derecesi az olan parçalarını dönüştürerek, bu parçaların paralellik derecesini artırmaktadır. Satır bağımlılıklarını esneterek ya da tamamen ortadan kaldırarak, daha çok çekirdek kullanımına olanak sağlar. Buna ek olarak, ince seviyeler bu yöntemler ile tamamen ortadan kaldırılarak senkronizasyon bariyeri ihtiyacını azaltmış olur.



a) Seyrek Alt Üçgen L'nin Bağımlılık Grafı

b) Satır 3'ün satır 1'e olan bağımlılığı kırıldı, satır 3 seviye 3'ten seviye 2'ye çıkarıldı

c) Satır 3'ün satır 0'e olan bağımlılığı kırıldı, satır 3 seviye 2'ten seviye 1'ye çıkarıldı

Şekil 2. Yeniden Yazma Yöntemi

LİTERATÜR TARAMASI

[1]'de Gauss Eliminasyonunda Markov Karar Sürecinin öğrenilmesi adına yapılan bir yaklaşımdan bahsedilmektedir. Burada özellikle Reinforcement Learning'in bir öğrenim şekli olan Q-Learning tekniği ele alınmıştır. Burada birden fazla Q-learning tekniği okuyucuya önerilmiştir. Bunlar minimum degree ordering, task scheduling and adaptive pivoting. Öğrenmeye dayalı bir algoritma olan bu Teknik ile seyrek çözücüde performansı artırmaya dayalı bir şekilde kullanılmıştır.

Bizim projemizde Seyrek alt üçgensel bir matrisler üzerinde işlem yaptığımızdan dolayı genel olarak burada Gauss Eliminasyonu ile yeniden yazma işlemlerini gerçekleştiriyoruz. [2] burada ki kitabın Chapter 3'ünde Sonlu Markov Karar Süreçlerini anlatarak bizlere bu konuda daha güzel örnekler üzerinden RL ile Markov'u beraber nasıl kullanabileceğimiz konusunda eğitmektedir.

[2]'nin Chapter 3'ünde ilk önce RL'in çalışma prensiplerinde Agent-Environment ilişkisinden bahsedilmektedir. Daha sonrasında ise Ödüller ve Cezalar üzerinden durulmaktadır. RL'de eğer istenilen işlemler sağlanıyorsa ödül sağlanmıyor ise ceza alarak ajanımız bir oyun oynar gibi eğitilmektedir. Burada verilen bir örnekte Çöp Toplama robotunun hareketleri üzerinden bir anlatım yapılmıştır. Robotumuzun şarjının 2 adet durumu vardır bunlar "high,low" ve robotumuz 3 tane işi yapabiliyor "search, wait, recharge". Eğer robotumuzun pili "high" ise "search,wait" işlemlerini "low" ise 3 işlemide yapabiliyor, eğer "search " işlemini yaparken şarjı biterse ceza alıyor, her topladığı çöp için ise bir adet ödül puanı alıyor. Robotun çalışma şekli Reinforcement Learning ile bu şekilde kuruluyor ve eğitiliyor. Burada Markov Karar Süreçlerini ise bu durumların koşullu olarak oluşma olasılıkları üzerinden hesaplanarak yapılıyor.

SpTRSV optimizasyonuna yönelik en iyi bilinen yaklaşımlar seviye ayarlı yöntemler[6-7] ve senkronizasyonsuz yöntemlerdir[8-10]. Düzey kümeli yöntemlerde, birbirine bağımlılığı olmayan satırlar düzeyler halinde gruplandırılır. Bu nedenle, bir düzey içindeki satırlar, bir dizi iş parçacığı kullanılarak paralel olarak hesaplanabilir. Farklı seviyelerdeki satırlar birbirine bağımlı olabileceğinden, her seviye sonunda bir senkronizasyon bariyerine ihtiyaç duyulur ve seviyeler seri bir şekilde hesaplanır.

Düzey kümesi yöntemleri, bağımlılık grafiğini satırların doğal sıralamasına göre yatay olarak bölmekle birlikte, düzeyler, senkronizasyon engelleri nedeniyle ek yüke maruz kalır ve bunların seri olarak hesaplanması gerekir, aksi takdirde birbirleri için boşta beklemek zorunda kalacaklardır. Bu senaryo, etkisiz bir yük dengeleme yaklaşımına işaret ediyor olabilir, ancak daha da önemlisi, hesaplaması biten satırlar, engeller nedeniyle, kendilerine bağlı olan satırlar için hesaplamaları hemen "açamaz". Matrisin seyreklik modeli, yük dengelemeyi daha da karmaşık hale getirebilir.

Bu dezavantajları ortadan kaldırmak için seviye setli yöntemlere bir alternatif olarak senkronizasyonsuz yöntemler ortaya çıkmıştır. Senkronizasyondan bağımsız yöntemler, yatay yerine satırları bağımlılıklarına göre gruplandırır, dolayısıyla bölümleme dikeydir. Bu yaklaşımın ilk örnekleri [8]'da CPU'lar üzerindeydi. Daha sonra, GPU'lar [9-10] üzerinde birkaç uygulama önerilmiştir. [11]'de yazarlar karşı tabanlı bir çizgeleme mekanizması sunarlar ve seviyeleri ayarlamak için parelel bir topolojik sıralama algoritması kullanırlar ve her eleman sadece kendi bağımlılıklarını bekler. Senkronizasyon içermeyen yöntemlerle, ince taneli görevler oluşturulabilir ve senkronizasyon engelleri ortadan kaldırılabilir ancak bu yöntem, görevlere atanması ve önceliklerde meşgul beklemesi için yüzlerce veya binlerce iş parçacığı gerektirir. Bu nedenle senkronizasyonsuz yöntemler GPU'lar için daha uygun görülmektedir.

METOT VE YAPILACAK TESTLER

Bu proje için geliştirdiğimiz en son algoritma 3 adet kritere sahip ve bu kriterler üzerinden gerekli işlemler yapılarak matrisimizi optimal hale getirmeyi amaçlamaktadır. Bu algoritma her bir düğümü yeniden yazma işlemini yaparken bakması gereken 3 adet kriter vardır bunlar:

1-avgCostPerLevel – Seviyelerde ki ortalama maliyet değerinden büyük veya küçük olma durumlarına göre gerekli işlemler yapılır.

2-avgInPerRow – Ortalama maliyet değerimizden küçük seviyelerde ki ortalama düğüm sayısından küçük veya büyük olma durumuna göre gerekli işlemler yapılır.

3-avgIndegrePerLevel – Ortalama maliyet değerimizden küçük seviyelerde giren düğüm sayılarının ortalaması. Bu değerden küçük veya büyük olma durumlarına göre gerekli işlemler yapılır.

Eğer bu değerlerden küçük ise bir üstte ki seviyeye gider ve orada bu kriterleri arar. Eğer bu kriterlerden herhangi birisi sağlanmaz ise o düğüm en son sağlanan seviyeye yazılır. Bu şekilde daha optimal bir matris elde etmek istenmektedir. Bu algoritma için lung_2 isimli matris kullanılmaktadır.

PROJE İLE İLGİLİ YAZ491'IN SON RAPORUNDAN YAZ492 İLK RAPORUNA KADAR YAPILAN İŞLER

En son ki raporda yazılı olan algoritma biraz geliştirilerek bu algoritma BAŞARIM 2022 konferansı için bildiri olarak gönderildi ve yakın zamanda bu bildiri kabul gördü. Buraya gönderdiğimiz algoritma en son raporda ki algoritmanın biraz daha geliştirilmiş halidir. Bu algoritma genel olarak şu şekilde çalışmaktadır :

Algoritma, ince seviyeler arasında yeniden yazma işlemini uygular. İnce Seviyeler, toplam maliyeti avgLevelCost'dan daha küçük olan seviyelerdir. İnce seviyelerin ikinci seviyesinden başlayarak (seviye 1 hedef seviye), bu seviyeler (kaynak seviye) yeniden yazdımız seviyenin maliyeti avgLevelCost'a ulaşana kadar üst seviyelere yeniden yazılır. Bu nedenle, seviye sayısını azaltmak için kaynak seviyeler silinir ve hedef seviyelerin maliyetleri avgLevelCost'a yükseltilir. AvgLevelCost süreç boyunca sabit tutulur.

Level	Rows		Cost Map (row #, cost)		
n	0, 1	1, 1	2, 6	3, 6	4, 7
n+1	~~2, 3~~	~~3, 3~~			
n+2	~~4, 3~~	5, 5	6, 5	7, 7	
n+3	~~6, 3~~	~~7, 3~~			

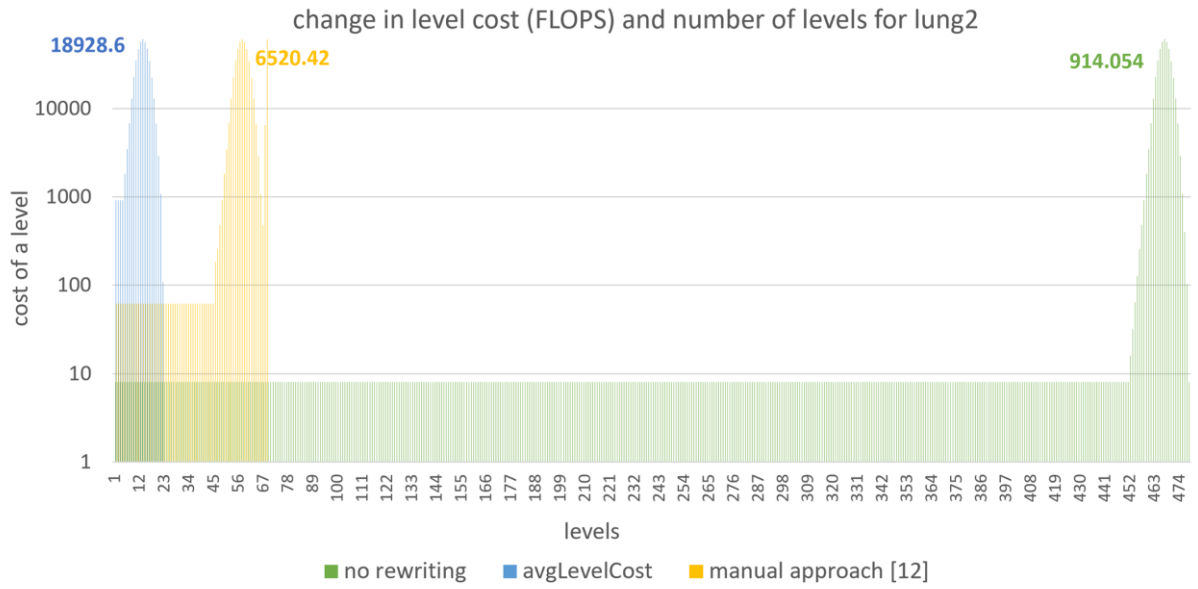
Tablo 1. Satırların Yeniden Yazılma İşlemi

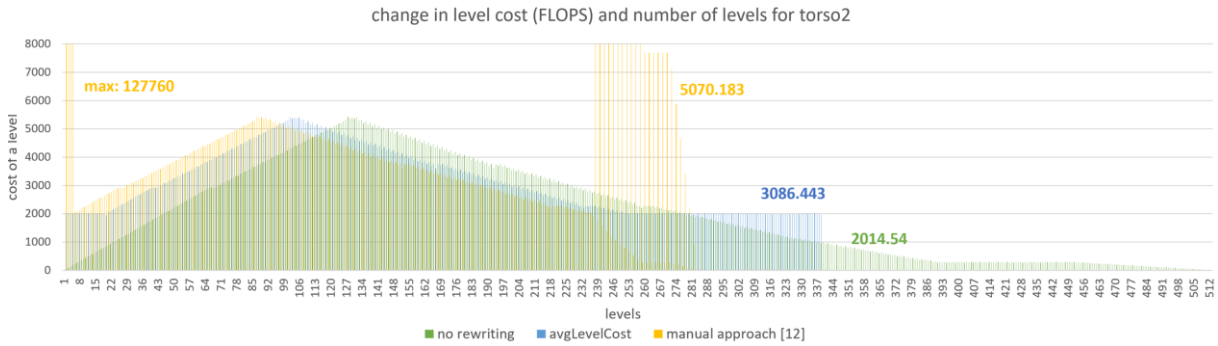
Yeniden yazma işleminden sonra seviye sayısı düşürülür ve seviye sayıları şu şekilde görülür:

Level	Rows					Level Cost
n	0, 0	1, 0	2, 7	3, 7	4, 7	21
n+1~~n+2~~	5, 5	6, 5	7, 7			17

Tablo 2. Matrisin En Son Durumu

Aşşağıda lung2 ve torso2'ye Seviye Maliyeti ve Seviye Sayısında ki Değişiklik Grafikleri gösterilmektedir:





Tablo 3. lung2 ve torso2 için seviye maliyeti, yeniden yazma işlemi yapılmadan, manuel yaklaşım ve avgLevelCost yaklaşımının karşılaştırmalı olarak grafikleri verilmiştir.

Bu algoritmadan sonra daha efektif olması açısından yeni bir algoritma daha oluşturduk ve şuan bu algoritmanın koda dökülmesi aşamasındayız. Bu algoritmanın çalışma prensibi şu şekildedir :

Öncelikli olarak Ortalama Seviye Maliyeti (avg. level cost), Ortalama Giren Düğüm Derecesi (avg. indegree per row), (Yeniden Yazılacak Seviyelerde) Seviye Başına Düşen Ortalama Düğüm Sayısı (avg. rows per level in levels to be rewritten) bu kriterler hesaplanır.

Daha sonrasında yeniden yazılacak seviyeler Ortalama Seviye Maliyeti'ne göre belirlenir. En alt seviyelerden başlayarak ;

a. Satırın düğüm derecesi, Ortalama Giren düğüm derecesinden küçük ise,
b. Yeniden yazılan seviyenin maliyeti, ortalama seviye maliyetinden küçük,
c. Yeniden yazılan seviyenin satır sayısı, Seviye başına düşen ortalama düğüm sayısından küçük ise,

Bu kriterlerin hepsi sağlandığı sürece düğümü bir üst seviyeye taşırız. Kriterlerden herhangi biri sağlanmaz ise kriterlerin sağlandığı en son ki satıra yazılır.

PROJE İLE İLGİLİ SONRAKİ RAPORA KADAR YAPILACAKLAR VE BEKLENEN SONUÇLAR

Final Raporuna kadar en son yazdığımız algoritmayı tamamlamayı planlıyoruz daha sonrasında farklı veri setleri ile bu algoritmanın test edilmesini sağlayacağız.

Bu problemin sonuçlarının en optimal şekilde olması istenmektedir. 1.Algoritma da aldığımız sonuçlar istenilen düzeyde olmadığı için 2.Algoritmayı yazma gereği duyduk. Aldığımız sonuçlar sonrasında 2.Algoritmayı yazma sebeplerimiz şunlardır:

- Sadece Ortalama Seviye Maliyeti üzerinden işlem yaptığımız için yeniden yazdığımız seviyelerde ki düğümlerin sayısının ortalamadan yüksek olması.

- 1.Algoritma da seviye sayımızda azalma oldu ama seviyelerde hem düğüm hemde ortalama maliyetleri arasında ciddi farklılıklar meydana geldi.

Bu sebepler istenilen optimallik düzeyine ulaşmadığı için 2.Algoritmayı yazıp bu sebeplerinde ortadan kaldırılmasını amaçlıyoruz.

KAYNAKÇA

Kaynakça Chicago formatında eklenmelidir.

[1]Yingshi Chen. "LearningThe Markov Decision Process In The Sparse Gaussian Elimination." arXiv: 2109.14929v1 30 Sep 2021

[2]Richard S. Sutton and Andrew G. Barto,. Reinforcement Learning. 2016

[3]Christian Schulz. "High Quality Graph Partitioning" Doktora Tezi, Karlsruher Instituts für Technologie, 2013.

[4]John E. Savage and Markos G. Wloka. "Parallelism in Graph-Partitioning" Journal of Parallel and Distributed Computing 13(1991): 257-272.

[5]Buse Yılmaz, Graph Transformation and Specialized code Generation For Sparse Triangular Solve(SpTRSV), 2103.11445v1, 21 Mart 2021

[6]E. Anderson, and Y. Saad, "Solving sparse triangular linear systems on parallel computers". International Journal of High Speed Computing 1, 1, 1989, 73–95.

[7]R. Li, and Y. Saad, "GPU-accelerated preconditioned iterative linear solvers". 2013, The Journal of Supercomputing 63, 443–466.

[8]S. W. Hammond, and R. Schreiber, "Efficient iccg on a shared memory multiprocessor". 1992, International Journal of High Speed Computing 04, 01 (1992), 1–21

[9]J. I. Aliaga, E. Dufrechou, P. Ezzatti, and E. S. Quintana-ort´i, "Accelerating the task/data-parallel version of ilupack's bicg in multi- cpu/gpu configurations". 2019, Parallel Computing 85 (2019), 79 – 87.

[10]W. Liu, A. Li, J. Hogg, I. S. Duff, and B. Vinter. "Fast synchronization - free algorithms for parallel sparse triangular solves with multiple right- hand sides". 2017, Concurrency and Computation: Practice and Experience 29, 21 (2017).

[11]R. Li, "On Parallel Solution of Sparse Triangular Linear Systems in Cuda". 2017, Technical Report.