Uludağ Üniversitesi Mühendislik Fakültesi Dergisi, Cilt 27, Sayı 1, 2022 ARAŞTIRMA

DOI: 10.17482/uumfd.911736

GPU PROGRAMLAMA TEKNİĞİ KULLANILARAK HIZLANDIRILMIŞ XCLASS YAZILIMI İLE SPEKTRUM ANALİZİNİN GERÇEKLEŞTİRİLMESİ

Yasemin POYRAZ KOÇAK [*]
Selçuk SEVGEN [**]

Alınma: 08.04.2021 ; kabul: 13.02.2022

Öz: Astronomi, çok büyük boyutlu veriler ile çalışmak zorunda olan bir alandır. Bu büyük boyutlu verilerin işlenmesi ve bu verilerden bilgi çıkarımı işlemi çok uzun zaman aldığı için bu verileri modelleyen bilgisayar yazılımlarının performansı çok önemlidir. Bu çalışmada, Atacama Büyük Milimetre/Milimetre-altı Dizisi (Atacama Large Millimeter/Submillimeter Array- ALMA) isimli teleskop setinden gelen astronomik verileri modellemek için geliştirilen XCLASS (extended CASA Line Analysis Software Suite-genişletilmiş CASA Çizgi Analiz Yazılım Bölümü) adlı yazılımın çok yoğun hesaplamaların yapıldığı myXCLASS adlı bölümü, Grafik İşlemci Birimi (Graphics Processing Unit-GPU) programlama tekniği kullanılarak hızlandırılmıştır. GPU programlama ortamı olarak Birleşik Hesap Cihazı Mimarisi (Compute Unified Device Architecture- CUDA) kullanılmıştır. Uygulama, Tesla K20m isimli iki adet ekran kartı üzerinde test edilmiş, CPU- GPU çalışma süresi bakımından performans karşılaştırılması yapılmış ve ayrıntılı sonuçlar sunulmuştur. Elde edilen sonuçlar ile GPU programlama tekniği kullanılarak ALMA gibi gelişmiş gözlem araçlarından elde edilen çok büyük boyutlu astronomik verilerin modellenmesinde kullanılan yazılımlarda yüksek performans kazanımı sağlanacağı gösterilmiştir.

Anahtar Kelimeler: Grafik İşlemci Birimi, Birleşik Hesap Cihazı Mimarisi, Paralel Hesaplama, XCLASS, Spektral Analiz

Implementing Accelerated Spectrum Analysis on XCLASS Software by Using GPU Programming Technique

Abstract: Astronomy is a field that has to deal with massive amount of data. The performance of software modeling this data is very important because it takes a very long time to process and extract information from these massive amount of the data. In this study, myXCLASS which is very compute-intensive parts of extended CASA Line Analysis Software Suite (XCLASS) which is developed to model astronomical data taken from Atacama Large Millimeter/Submillimeter Array (ALMA) devices is accelerated using the Graphics Processing Unit (GPU) programming technique. Compute Unified Device Architecture (CUDA) is used as a GPU programming environment. The application is tested using two Tesla K20m GPU cards, CPU- GPU versions of the software are compared in terms of the time and obtained experimental results are presented in detail. The obtained experimental results show that high performance gain can be achieved in software used for modeling very large astronomical data obtained from advanced observation tools such as ALMA using GPU programming technique.

Keywords: Graphical Processing Unit, Compute Unified Device Architecture, Parallel Computing, XCLASS, Spectral Analysis

[*] İstanbul Üniversitesi-Cerrahpaşa, Teknik Bilimler Meslek Yüksekokulu, Bilgisayar Teknolojileri Bölümü, Alkent 2000 Mahallesi, Yiğittürk Cad. No:5/9/1, 34555, Büyükçekmece/İstanbul/Türkiye
[**] İstanbul Üniversitesi-Cerrahpaşa, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü, Avcılar Kampüsü, 34320, Avcılar/İstanbul/Türkiye

İletişim Yazarı: Selçuk Sevgen (sevgens@iuc.edu.tr)

1. GİRİŞ

Astronomi alanında, gökyüzünden alınan veri analizi önemli bir kavramdır. Günümüzde, sürekli olarak, Plateau de Bure (Plateau-de-bure, 2010), CARMA (Combined Array for Research in Millimeter-wave Astronomy) (CARMA, 2012), SMA (Submillimeter Array) (SMA, 2005), VLA (Very Large Array) (VLA, 2008) ve ALMA (ALMA, 2012) gibi gelişmiş gözlem araçlarından elde edilen verilerin boyutu gün geçtikçe artmaktadır. Gelecekte daha geniş bantlı gözlem araçları ile gözlem yapabilme imkânı olacağı ve elde edilen verilen daha da artacağı düşünülmektedir. Bu durum, astronomi alanının bilgisayar gücüne olan gereksinimini gün geçtikçe artırmaktadır. Astronomi aynı zamanda gelişmiş gözlem araçlarından elde edilen verilerin boyutunu azaltmaya ve bu verilerin sadece işlenebilir olanlarını almaya da ihtiyaç duymaktadır. Bu büyük boyutta verilerin modellenmesi için IRAF (Image Reduction and Analysis Facility) (IRAF, 2014), AIPS (Astronomical Image Processing System) (AIPS, 2011), CLASS (Continuum and Line Analysis Single-dish Software) (CLASS, 2010), GILDAS (Grenoble Image and Line Data Analysis Software) (GILDAS, 2010), XCLASS (XCLASS, 2016) gibi yazılımlar geliştirilmiştir. Bu yazılımlar hakkında genel bilgiler Tablo 1'de verilmiştir.

Tablo 1: Astronomik verileri modelleme amacı ile kullanılan yazılımlar.

Yazılım	Geliştiren kurum	Amaç	Çalıştığı platformlar
IRAF	National Optical Astronomy Observatory (NOAO)	Astronomik görüntüleri piksel dizisi formatında azaltma amacı ile geliştirilmiştir. Astronomik nesnelerin konumlarının kalibrasyonunun, detektör pikseller arasındaki hassasiyetinin ölçülmesi, görüntülerin kombinasyonunun gerçekleştirilmesi, spektrumdaki kırmızıya kaymalarının ölçülmesi işlemlerini gerçekleştirir.	Bütün işletim sistemleri için uygundur.
AIPS	National Radio Astronomy Observatory (NRAO)	Radyo teleskoplarla alınan verilerin azaltılmasını ve analizini yapmayı amaçlayan yazılım paketidir. Veri setinin görselleştirmesi, düzenlemesi ve kalibre etmesi işlemlerini ve uygun modeller oluşturması işlemini gerçekleştirir.	Windows X, Linux, Mac-OS platformlarında çalışır.
CLASS	Institute for radio astronomy Millimeter (IRAM)	Tek çanak bir teleskopla elde edilen spektroskopik verileri azaltmak için kullanılan bir yazılım paketidir. Ayrıca işaretleme veya odaklanma gibi süreklilik sürüklenmelerini azaltmak için temel işlevlere sahiptir.	Unix ve Mac platformlarında çalışır.
GILDAS	Institute for radio astronomy Millimeter (IRAM)	Veri azaltma paketi olarak kabul edilir. Daha önce var olan CLASS yazılım paketine ek işlevler yerleştirilerek oluşturulmuştur, 3 boyutlu verilerin spektral çizgi haritalanması gibi çok sayıda fonksiyon içerir.	Unix ve Windows platformlarında çalışır.

Astronomik verileri işlemek amacı ile kullanılan yazılımlar çok büyük boyutlu verileri işlemek için yüksek hesaplama gücüne ihtiyaç duyar. Günümüzde bu çok boyutlu verilerin işlenmesi için kullanılan CPU'lar tek başına gerekli performansı sağlamakta yetersiz kalmaktadır. Bu sorun, OpenMP (Open Multi-Processing) (Jang ve diğ., 2008), MPI (Message Passing Interface) (Diaz ve diğ., 2012), SMP (Symmetric Multiprocessing) (Arimilli ve Siegel, 2002) ve CUDA (NVIDIA, 2007) gibi paralel işlem yapabilme özelliğine sahip yöntemlerin kullanılmasıyla çözülebilir hale gelmiştir.

OpenMP, çoklu iş parçacığı yönetiminin gerçekleştirilmesi üzerine kurulu bir paralel programlama tekniğidir. Bu yöntemde, CPU'daki çekirdekler ana ve yardımcı çekirdekler olmak üzere iki bölüme ayrılır. Ana iş parçacığı belirli bir sayıda yardımcı iş parçacıklarını durdurur ve görevi onlar arasında paylaştırır. Yardımcı iş parçacıkları birbiri ardına paralel şekilde çalışır. (Jang ve diğ., 2008) Kullanılabilmesi için öncelikle sisteme yüklenmesi gereken

MPI kütüphanesi, OpenMP gibi çok çekirdekli CPU üzerinde uygulanır. CPU'lar arasında mesajlaşma imkânı sayesinde program çalıştığında çağırılacak fonksiyonun sırasının belirlenmesi sağlanır. Her bir CPU'ya ayrı bir fonksiyon da tayin edilebilir. (Diaz ve diğ., 2012) SMP, çok çekirdekli işlemci üzerine kurulu bir mimarîdir. Bu mimarîde çekirdeklerin her biri ayrı birer işlemci olarak görev alırlar ve ortak bir belleğe sahiptirler. Ayrıca her işlemci, veri girişini hızlandırmak ve sistem trafiğini azaltmak için bir ön belleğe sahiptir. Bu mimaride işlem hızını artırmak için herhangi bir fonksiyon bir işlemciden başka bir işlemciye gönderilebilir. (Arimilli ve Siegel, 2002)

CUDA, NVIDIA tarafından geliştirilmiş GPU üzerinde çalışan bir paralel hesaplama platformu ve programlama modelidir. GPU içerisinde iş parçacığı (thread), temel birimdir ve çok sayıda iş parçacığı bir araya gelerek iş parçacığı blokları, bloklar bir araya gelerek ızgaraları (gridleri) oluşturmaktadır. GPU programlamada fonksiyonlar çekirdek (kernel) olarak adlandırılır ve GPU üzerinde çekirdekler iş parçacıkları üzerinde paralel olarak çalışır. Programcı çekirdeklerin üzerinde çalışacağı iş parçacığı bloklarını ve ızgaralarını hiyerarşi içinde çalışacak şekilde programlamalıdır. Ayrıca CPU ve GPU'nun ortak bir bellek alanı olmaması sebebiyle GPU'nun verileri işleyebilmesi için öncelikle CPU'dan verileri alması gerekmektedir. Bu nedenle GPU'nun verimliliğinden en üst düzeyde yararlanabilmek için, CPU ve GPU arasındaki veri aktarımını mümkün oldukça en aza indirmek önemlidir. (NVIDIA, 2007, NVIDIA ve PGI, 2008, NVIDIA Corporation, 2017, Ruetsch ve Fatica, 2013). XCLASS adlı program Köln Üniversitesi, Astrofizik Anabilim Dalı'nda görev yapan Prof. Dr. Peter Schilke ve Dr. Thomas Möller tarafından Fortran programlama dili kullanılarak yaklaşık 10 yılda geliştirilen, çok kapsamlı, çalışması için yüksek kapasiteli cihazlara ihtiyaç duyulan bir programdır. Bu nedenle XCLASS'ın hızlandırılması işlemini gerçekleştirmek için CUDA Fortran programlama dili kullanılmıştır.

GPU programlama tekniği ilk olarak ortaya atıldığında amaç 3 boyutlu video oyun uygulamalarını hızlandırmak iken, daha sonra hesaplama kapasitelerinin artması ile birlikte özellikle astronomi gibi çok büyük boyutlu veriler ile işlem yapılmasını gerektiren bilimsel alanlarda kullanılmaya başlanmıştır. Yapılan çalışmalar aşağıda kısaca açıklanmıştır:

XCLASS'ın içerisinde myXCLASS adlı bölümünün fonksiyonlardan biri olan myXCLASSSfit ile benzer işlevi gerçekleştiren (Barsdell ve diğ., 2011)' deki çalışmada, GPU programlama tekniği kullanılarak hızlandırma işlemi gerçekleştirilmiştir. Tesla C1060 GPU kartının kullanıldığı bu çalışmada Levenberg-Marquardt Algoritması kullanılarak gerçekleştirilmiştir. CPU'ya göre süre açısından 10 kat daha hızlı sonuç elde edilmiştir ancak, galaksi çizgi analizi uygulamasının kalitesini ve sağlamlığını arttırmak için daha iyi performans gösteren algoritmalara ihtiyaç olduğu görülmüştür. (Cárcamo ve diğ., 2018)' de gerçek ve yapay veriler bir arada kullanılarak Maximum Entropy Method (MEM)'in yüksek performanslı GPU versiyonu gerçekleştirilmiştir. Çalışma hem tek GPU üzerinde hem de birden fazla GPU üzerinde test edilmiştir. Sonuçlar, veri ve görüntü boyutuna bağlı olarak, CPU versiyonuna göre 1000 ile 5000 kat daha hızlı sonuçlar elde edilebileceğini göstermiştir. (Mei ve diğ., 2017)' de, MingantU SpEctral RadioHeliograph (MUSER)'den elde edilen verileri işlemek için gerçekleştirilen görüntüleme uygulamasında, güneş diski ve gökyüzü parlaklığının algılanması, güneş diskinin otomatik merkezlenmesi işlemlerinin döngü sayısının tahmini gibi bir dizi kritik algoritma ve bunların sözde kodları ayrıntılı olarak sunulmuştur. GPU tekniği kullanılarak gerçekleştirilen bu algoritmaların performansının, XCLASS 'ın içerisine entegre edildiği Common Astronomy Software Applications (CASA) yazılım paketinden yaklaşık olarak 2 ile 8 kat daha hızlı olduğunu ifade eden sonuçlar, önerilen görüntüleme yaklaşımının MUSER'in işlem performansını önemli ölçüde arttırdığını ve MUSER veri işleminin gereksinimlerini karşılayabilecek yüksek kalitede görüntü oluşturduğunu göstermiştir. (Westerlund ve Harris, 2015) 'teki çalışmada, bizim çalışmamızda kullanılan verilerle aynı özelliklere sahip olan spektral-çizgi radyo astronomisinde, hesaplama bakımından yoğun bir süreç olan interferometrik verilerinin elektromanyetik emisyon kaynaklarının araştırılması konusu ele

alınmıştır ve HI emisyonundan üretilen spektral görüntü küplerini aramak için kaynak bulma teknikleri tasarlanmıştır. Çalışmada, bellek yönetimi teknikleri ile GPU programlama tekniği iyi bir bellek yönetim şeması kullanılarak CPU versiyonundan 2 kat daha hızlı sonuçlar elde edilmiştir. pModern yüksek çözünürlüklü Kütle Spektrometresi cihazları, tek bir sistem biyolojisi deneyinde milyonlarca spektrum üretebilir. Her spektrum binlerce tepeden oluşur. Gürültülü ve kullanışlı olmayan tepe noktalarını tespit etmek için verilerinin ön işlemesi aktif bir araştırma alanıdır. (Awan ve Saeed, 2017)'deki çalışmada, G-MSR adı verilen GPU tabanlı bir boyut azaltma algoritması sunulmuştur. G-MSR, CPU üzerinde çalışan versiyonuna göre 386 kat hızlanma elde eder ve sadece 32 saniyede bir milyonun üzerinde spektral verinin işlenmesini sağlamıştır. Grafik işleme birimleri (GPU'lar), geleneksel çok çekirdekli CPU'lara kıyasla watt başına performansta önemli enerji verimliliği sağlar. Sistemlerde yüksek performansı sürdürürken toplam enerji tüketimini azaltmak gerekir. (Abe ve Diğ, 2012)'deki çalışmada, GPU hızlandırmalı sistemlerin güç ve performans analizini sunulmuştur ve GPU'ların voltaj ve frekans seviyelerini kontrol ederek sistem enerjisinin %28 oranında azaltılabileceğini ortaya koymuştur.

ALMA, milimetre ve milimetre altı dalga boyunda görüntüleme ve spektroskopik ölçüm yapabilen radyo teleskoplardan oluşan, evrenin fiziksel ve kimyasal yapısını araştıran astronomi alanındaki en büyük zemin tabanlı projelerden biridir ve ilk yıldızları ve galaksileri, gezegen oluşumlarını inceleyerek astronomlara yeni pencere açmaktadır. (ALMA, 2012) ALMA cihazlarından yapılan gözlemlerde çok büyük boyutlu spektral veriler elde edilmektedir. Elde edilen spektral verilerin analizi işlemi ise çok zaman almaktadır. Bu çalışmada, ALMA teleskop setinden elde edilen çok büyük boyutlu astronomik verileri kullanarak spektral analizi işlemini gerçekleştiren XCLASS'ın hesap yoğun kısımları GPU programlama tekniği ile hızlandırılmıştır. XCLASS çok geniş kapsamlı ve kullanılması için güçlü cihazlara ihtiyaç duyulan bir yazılım paketidir. XCLASS'ın yeniden başka bir programlama dili kullanılarak geliştirilmesi çok uzun yıllar sürecektir. Bu nedenle hızlandırma işleminin yapıldığı bölüm Fortran programlama dili kullanılarak geliştirildiği için hızlandırma işlemi için CUDA Fortran programlama dili kullanılmıştır. Ayrıca gerçekleştirilen hızlandırma işlemleri sayesinde,

- Astronomi alanında önem arz eden spektral analiz işleminde GPU ve CPU'da aynı sonuçları verdiği,
- ALMA cihaz setinden elde edilen spektral verilerde herhangi bir bozulma olmadan hızlandırma işlemlerinin gerçekleştirildiği,
- Çok daha büyük boyutlu astronomik verilerin daha kısa sürelerde işlenmesi mümkün hale geldiği,
- Gelecekte ALMA gibi gelişmiş gözlem araçları ile farklı bölgelerden elde edilen verilerin daha hızlı bir şekilde analizinin yapılması ile karanlık evrenin fiziksel ve kimyasal yapısı hakkında daha detaylı bilgi elde edile bilineceği,

tespit edilmiştir.

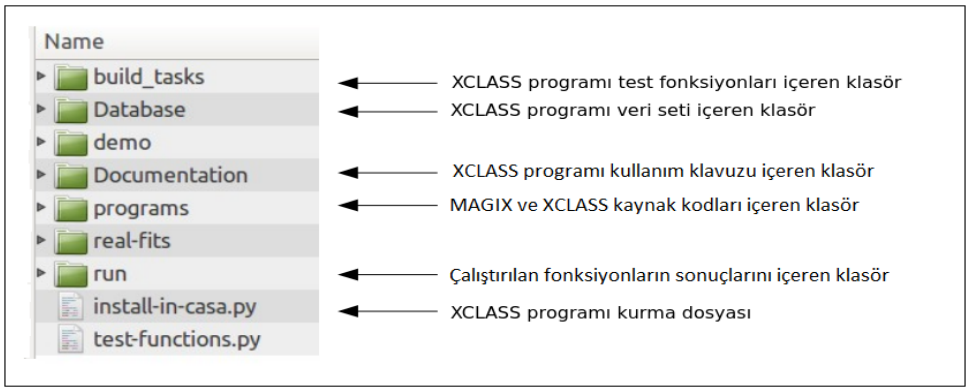
2. MATERYAL VE METOT

Bu bölümde, çizgi analizi işleminin nasıl gerçekleştiği, hızlandırma işleminin yapıldığı XCLASS yazılımından, fonksiyonlardan ve yapılan optimizasyon işleminden bahsedilmiştir.

2.1. XCLASS

CASA, ALMA ve VLA radyo teleskopların verilerini işleme amacı ile geliştirilen güçlü bir yazılımdır. Uluslararası bilim insanlarından oluşan bir konsorsiyum tarafından geliştirilen bu yazılım, astronomik verileri işlemek için temel fonksiyonlar içermektedir. (CASA, 2016) XCLASS ise, CASA için yazılmış tek boyutlu bir radyasyon transfer programıdır ve XCLASS, CASA'ya UpdateDatabase, DatabaseQuery, ListDatabase, GetTransitions, LoadASCIIFile,

myXCLASSPlot, myXCLASS, MAGIX, myXCLASSFit ve myXCLASSMapFit, myXCLASSMapRedoFit fonksiyonları eklenerek oluşturulmuştur. Şekil 1'de hem Linux hem de Mac platformları üzerinde çalışabilen XCLASS yazılımının yapısı gösterilmiştir.

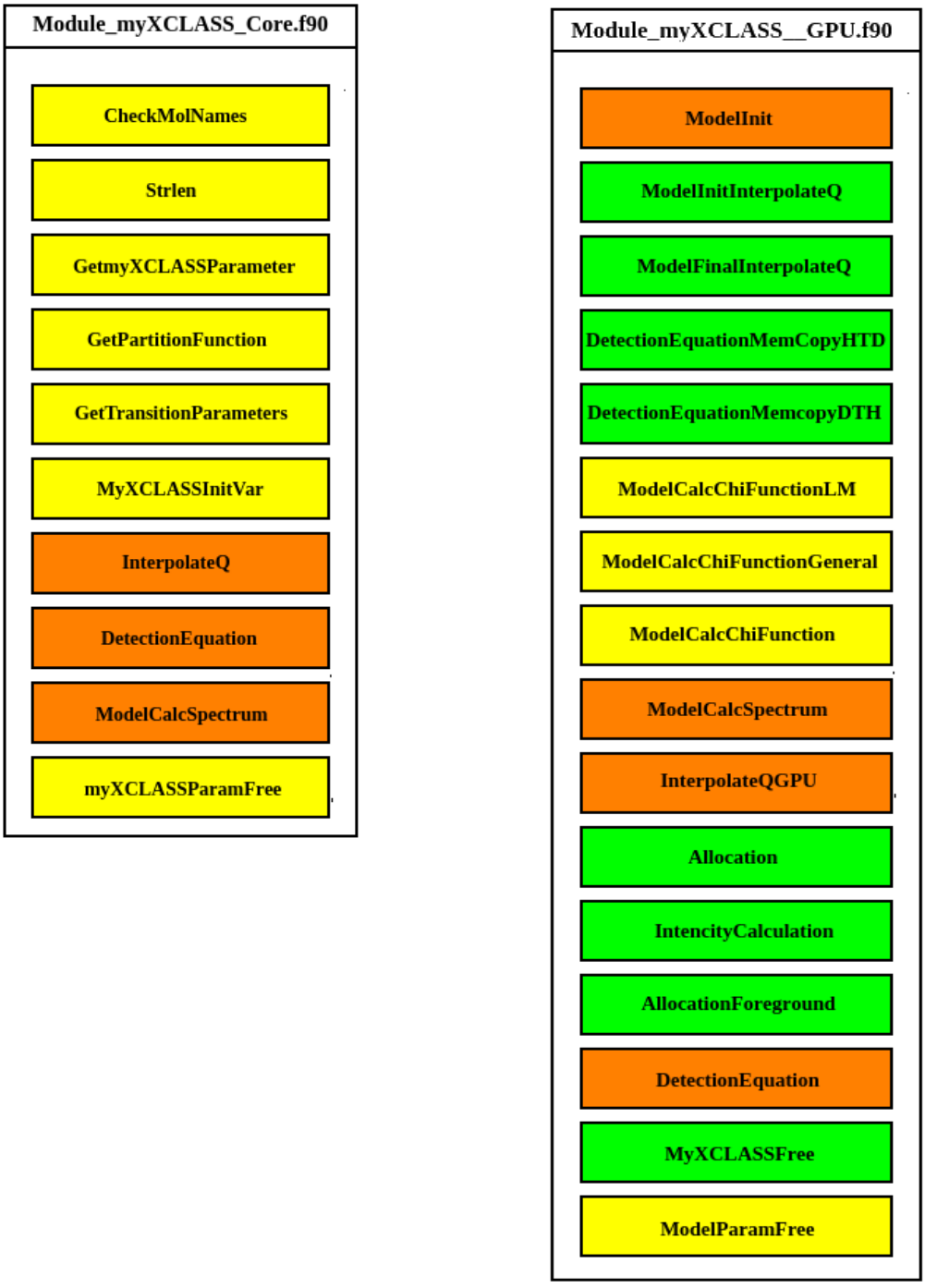


Şekil 1:
XCLASS yazılımının yapısı. (Sánchez, 2018)

CASA, XCLASS programını kullanarak ALMA'dan alınan verilerin bulunduğu Cologne Database for Molecular Spectroscopy (CDMS) / Jet Propulsion Laboratory (JPL)'den alınan moleküler verilerin modellemesini sağlamaktadır. Bu veri modelleme işlemini içerisinde bulundurduğu myXCLASS adlı program ile gerçekleştirmektedir. myXCLASS programı, bir boyutta izotermal bir nesne için sonlu kaynak boyutunu ve toz zayıflamasını göz önünde bulundurup ışınımsal transfer denklemini çözerek verileri modellemektedir. Aynı zamanda verilen spektrumda çizgi-yoğun yerleri tanımlamak için tasarlanmıştır. myXCLASS programı, bir spektrumu keyfi sayıda molekül ile modelleyebilir ve böylece her bir molekülün nerede katkısı olduğunu tanımlayabilir. myXCLASS, aşağıda gösterilen 4 adet Fortran dosyasını içermektedir:

- Module_myXCLASS__Core.f90
- Module_myXCLASS__GPU.f90
- Module_myXCLASS__MPI.f90
- Module_myXCLASS__SMP.f90

Şekil 2(a)'daki Module_myXCLASS_Core.f90 dosya içerisinde bulunan turuncu renkli InterpolateQ, DetectionEquation ve ModelCalcSpectrum adlı fonksiyonlar Module_myXCLASS_GPU.f90 adlı dosyaya taşınmıştır. Şekil 2(b)'deki sarı renkli fonksiyonlarda herhangi bir değişiklik yapılmamıştır. Yeşil renkli fonksiyonlar yeni oluşturulmuştur. Turuncu renkli fonksiyonlar ise Module_myXCLASS_Core.f90 adlı dosyadan alınmış ve GPU programlama tekniği kullanılarak "çekirdek"e çevrilerek hızlandırılmıştır. ModelInitInterpolateQ, ModelFinalizeInterpolateQ adlı fonksiyonlar InterpolateQGPU adlı çekirdekler için yazılmıştır ve InterpolateQGPU'de kullanılacak olan cihaz (device) değişkenler için CPU'dan GPU'ya ve GPU'dan CPU'ya bellek kopyalama işlemlerinin yapıldığı fonksiyonlardır. DetectionEquationMemCopyHTD, DetectionEquationMemCopyDTH adlı fonksiyonlar ise DetectionEquation adlı fonksiyon tarafından çağrılan Allocation, IntencityCalculation ve AllocationForeground adlı çekirdeklerde kullanılacak olan cihaz değişkenleri için CPU'dan GPU'ya ve GPU'dan CPU'ya bellek kopyalama işlemlerinin yapıldığı fonksiyonlardır.



Şekil 2:
a. *Module_myXCLASS_Core.f90 dosyası.* ***b.*** *Module_myXCLASS_GPU.f90 dosyası*

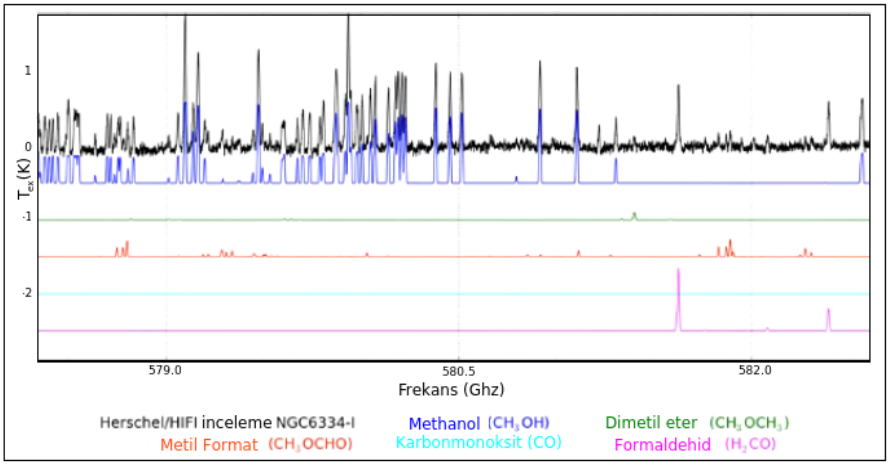
ModelCalcSpectrum adlı fonksiyon, verilen parametre vektörü için spektrum hesaplamaktadır. ModelCalcSpectrum içerisinde çağrılan InterpolateQ ve DetectionEquation fonksiyonlar spektrum hesabını hızlandırmak amacı ile CUDA Fortran programlama dili kullanılarak çekirdeğe dönüştürülmüştür. InterpolateQ fonksiyonunun ismi InterpolateQGPU olarak değiştilmiştir. DetectionEquation adlı fonksiyon içerisinde birbirine bağlı çok fazla değişken bulunmaktadır. Bu nedenle fonksiyonun tamamını çekirdeğe dönüştürmek yerine

hesap yoğun kısımlarını alınıp Allocation, IntencityCalculation, AllocationForeground adında yeni çekirdekler oluşturulmuştur.

Son olarak MyXCLASSFree adlı fonksiyon ise bütün Module_myXCLASS_GPU.f90 adlı dosya içerisinde bulunan cihaz dizilerinin program sonlandıktan sonra bellekte kapladığı alanı boşaltmak için oluşturulmuştur. ModelCalcSpectrum fonksiyon tarafından çağrılmaktadır. XCLASS içerisinde yapılan tüm bu değişikliklerin getirdiği performans kazancına ait veriler ve değerlendirmeler 3. Bölümde ayrıntılı olarak sunulmuştur.

2.1.1. Çizgi Analizi İşlemi

Her atomun, o atomun kimyasal yapısı hakkında bilgi veren ve bir nevi parmak izi özelliği taşıyan bir spektrum değeri bulunmaktadır. Örneğin Şekil 4' te siyah çizgi ile gösterilen çizgiler NGC6334-I adlı bulutsunun 579-582 Ghz frekans aralığındaki spektrumudur. Gözlem ile elde edilen gerçek spektrum, atmosferin etkisi ve fiziksel değişmelerden dolayı nesnenin kimyasal yapısı hakkında tam bir bilgi veremez ve çizgi analizi adı verilen bir işlemden geçmesi gerekmektedir. Matematiksel olarak bir dizi veri noktasına en iyi şekilde uyan bir çizgi veya matematiksel fonksiyon oluşturma işlemi olan çizgi analizi işlemi, ya verilere tam olarak uyan çizgi belirler (interpolation), ya da verilere yaklaşık bir çizgi tespit eder. (smooting).



Şekil 3:
NGC6334-I adlı bulutsunun 579-582 Ghz frekans aralığındaki spektrumuna uygulanan çizgi analizi işlemi. (Sánchez, 2018)

Şekil 3'te NGC6334-I adlı bulutsunun içerisinde bulunan bazı moleküller için myXCLASSFit fonksiyonun uygulanmasından elde edilen sonuç gösterilmiştir. Böylece hangi molekülün hangi frekans aralığında katkısı olduğu görülmektedir.

2.1.2. ModelCalcSpectrum Fonksiyonu

Bu fonksiyon belirli bir parametre vektörü için spektrum hesaplamaktadır. ModelCalcChiFunction adlı fonksiyon içerisinde bütün parametre vektörleri için ayrı ayrı çağrılarak bir model spektrum ve gerçek spektrum ile model spektrum arasındaki hata değerini gösteren ch^2 vektörünü oluşturmaktadır. ch^2 vektörü aşağıda bulunan denklem ile hesaplanmaktadır.

$$ch^2 = \sum_{i=0}^{N} \left[\left(y_i^{obs} - y_i^{fit} \right)^2 \frac{1}{\left(\sigma_i^{error} \right)^2} \right] \quad (1)$$

Bu denklemde, y_i^{obs} değeri i. noktadaki deneysel veriyi, y_i^{fit} ise i. deneysel veriye karşılık gelen değeri, σ_i^{error} de i. veri noktasının hatasını temsil etmektedir [29]. Spektral analiz işleminin başladığı ModelCalcSpectrum adlı ana fonksiyonun giriş argümanları Şekil 4'te gösterilmiştir.

```
!< parameter vector
ParameterVector
!< flag for indicating if model function values are stored or not
ModelFunctionFlag
!< total number of componentsDetect
NumComp
!< total number of frequency ranges
TotalNumFreqRan
!< total number of transition frequencies
TotalNumTransFreq
!< chi2 value for parameter vector
chi2Value
!< total number of molecules
TotalNumMol
!< total number of data points
NumDataPoints
!< output variable: model function values
ModelFuncList
!< number of free parameters
NumParamVector
```

Şekil 4:
ModelCalSpectrum fonksiyonunun giriş değerleri.

Çizgi analizi işlemi için seçilen molekül hakkındaki bilgilerin tutulduğu Molfit isimli bir dosya bulunmaktadır. Bu dosyada hangi molekül işlenecekse o molekül için bileşen sayısı tanımlanır. Kullanıcı bu dosya içerisinde molekülün her bir bileşeni için kaynak boyutu ($\theta^{(m,c)}$- *arcsec*), uyarım sıcaklık değeri (T_{ex}- *Kelvin*), sütun yoğunluğu (N_{tot}- $cm^{(-2)}$), hız genişlik değeri (FWHM – Maksimum genlik değerinin yarısının genişliği) ($\Delta v - kms^{(-1)}$), hız denge değeri ($V_{LRS} - kms^{(-1)}$) ve bayrak değişken (CFFlag: bileşenin ön plan mı - f çekirdek mi - c olduğunu belirtir) değerlerini tanımlamak zorundadır. Bu dosya içerisinde bulunan değerler myXCLASSParameter adlı global dizide tutulmaktadır.

Iso ratio dosyasında ise moleküller ve izotopları arasındaki izotop oranları tanımlanmıştır. ASCII formatındaki dosyanın ilk iki sütunu sırasıyla molekülleri gösterir. Üçüncü sütun, her iki molekül arasındaki izotop oranını gösterir. Bu dosya içerisinde bulunan değerler IsoRatioConversionTable adlı dizide tutulmaktadır. ModelCalcSpectrum fonksiyonunun işlem adımları şu şekildedir:

1- Molfit dosyasının parametreleri güncellenir.

2- Bütün rotasyon sıcaklıkları ve doppler-kayması geçiş frekansları için bölüşüm fonksiyon değeri belirlenir.

3- Uyarım sıcaklık değeri kontrol edilir.

4- InterpolateQGPU çekirdeği çağrılır.

5- Çıkış değişkenleri için bellekte alan tahsis edilir.

6- Emisyon bileşenlerinin sayısı belirlenir

7- Işın boyutu belirlenir:

Tek çanaklı veri için (InterFlag(l) = .false.) ışın boyutu aşağıdaki denklem ile belirlenir.

$$\theta_t = 1{,}22 * c/\nu D * \zeta = 1{,}22 * c/\nu D * (180.d0 * 3600.d\,0/pi) \quad (2)$$

Teleskop ışınının FWHM boyutu,

D, teleskopun çapını,

c, ışık hızını belirtir.

İnterferometrik veri için (InterFlag(l) = .true.) ışın boyutu aşağıdaki denklem ile belirlenir.

$$\theta_t = D \quad (3)$$

D, kullanıcı tarafından belirlenen interferometrik ışının FWHM boyutudur ve tüm frekans aralığında sabittir.

8- Maksimum ışın dolgu faktörünü ve tüm emisyon bileşenlerinin ilgili bileşen indeksini belirlenir.

9- Ön plan ve arka plandaki bileşenlerin yoğunluk değeri hesaplanır.

10- DetectionEquation fonksiyonu çağrılır.

11- Bütün bileşenlerin toplam yoğunluğu hesaplanır.

12- Modellenmiş fonksiyon çıkış dosyasına yazılır.

13- Hata'yı belirten chi^2 değeri hesaplanır.

2.1.3. InterpolateQGPU Fonksiyonu

İnterpolasyon, bir dizi veri noktası arasına yeni veri noktaları oluşturma yöntemidir. Astronomide gözlem yoluyla elde edilen çok sayıda veri noktası olduğundan bu bağımsız veri noktaları arasında bağlantı kurmak için interpolasyon işleminden geçirilmesi gerekmektedir [30]. InterpolateQGPU, doğrusal interpolasyon işleminin gerçekleştirildiği çekirdektir. Verilen sıcaklık aralığına göre bölüşüm fonksiyonu interpole edilmektedir. İlk önce uyarım sıcaklık değerinin logaritması aşağıdaki denklem ile hesaplanmaktadır.

$$x = log_{10}(temperature) \quad (4)$$

Elde edilen değer verilen en düşük sıcaklık değerinden daha az ise en düşük sıcaklık değeri bölüşüm fonksiyon değerine atanmaktadır. Verilen en yüksek sıcaklık değerinden daha yüksek ise en yüksek sıcaklık değeri bölüşüm fonksiyon değerine atanmaktadır. Daha sonra doğrusal interpolasyon işlemi başlamaktadır. Doğrusal interpolasyon hesaplamak için Denklem 5'te gösterilen formül kullanılmıştır. Denklemde *f(x),* bölüşüm fonksiyon değerini temsil etmektedir ve 2 boyutlu cihaz değişkeni olarak tanımlanmıştır.

$$f(x) = f_0 + \frac{(f_1 - f_0)}{(x_1 - x_0)} * (x - x_0) \quad (5)$$

Bu denklemde *f(x)* = [toplam bileşen sayısı, toplam molekül sayısı]'dır. Ayrıca, bölüşüm fonksiyon değişkeni için sıcaklık, *lgQ* ve bölüşüm fonksiyon değerlerini gösteren diziler cihaz

olarak sırasıyla Şekil 5'deki gibi tanımlanmıştır. Diğer değişkenler ise çekirdek içerisinde tanımlanmış yerel değişkenlerdir.

```
!< temperatures for partition function
TempPartFunc_d = [NumberOfTemperatures]
!< lgQ entries of table PartitionFunction
lgQ_d = [NumberOfTemperatures, NumberMoleculePartFunc]
!< partition function at the rotation temperatures
Olocal = [NumComp, TotalNumMol]
```

Şekil 5:
InterpolateQGPU fonksiyonunun cihaz dizileri.

Interpolate adlı fonksiyon tamamen CUDA Fortran programlama dili kullanılarak çekirdeğe dönüştürülmüştür. GPU için yapılan ayarlamalar aşağıdaki gibidir:

tBlock = dim3(MaxNumThreads, 1, 1)

grid = dim3(ceiling(real(NumberOfTemperatures)/tBlock%x), 1, 1)

MaxNumThreads sayısı, 3. Bölümde belirtildiği gibi Tesla K20m GPU kartında 1024 tür. Yapılan ayarlamalar cihaz olarak tanımlanan dizilerin boyutları ve Tesla K20m GPU kartının özellikleri göz önünde bulundurularak yapılmıştır. Çekirdeğin tamamı GPU üzerinde çalışmaktadır. Elde edilen performans kazanımı, 3. Bölümde ayrıntılı olarak sunulmuştur.

2.1.4. Detection Equation Fonksiyonu

Nesnenin fiziki yapısını öğrenmek için nesneden gelen ışığın spektrumunu incelemek en önemli yöntemlerden biridir. Nesnenin sıcaklık, kimyasal kompozisyonu, hareketi, kökeni ve evrimi ve diğer ipuçları gibi pek çok parametre spektrum analizi ile belirlenir. myXCLASS adlı program DetectionEquation fonksiyonu ile bir boyutlu izotermal bir nesne için ışınımsal transfer denklemini çözerek spektrum modellemektedir. Fonksiyon, belirli bir frekans aralığında ve bir boyutta izotermal nesne için ışınımsal transfer denklemini çözerek bir spektrum modeli oluşturur. Eğer transfer işlemi verilen frekans aralığında bulunmuyor ise fonksiyon bir şey yapmadan sonlandırılır. DetectionEquation adlı fonksiyonda, nesneden gelen ışığın spektrumdaki tek bir Gauss eğrisi aşağıdaki denklem ile hesaplanmaktadır. m değeri molekülü, c değeri ise molekül içinde bulunan bileşenleri, $\eta(\vartheta)^{m,c}$ molekül ve bileşenin ışın doldurma faktörünü, $(\vartheta)^{m,c}$,ϑ_t değerleri ise sırası ile kaynak ve teleskop ışın FWHM boyutunu temsil etmektedir.

$$(\eta(\vartheta)^{m,c}) = \frac{(\vartheta^{m,c})^2}{\vartheta_t(\nu)^2 + (\vartheta^{m,c})^2} \tag{6}$$

İkinci adım, Rayleight-Jeans sıcaklık değerinin belirlenmesidir. Fizikte Rayleight-Jeans, klasik argümanlar aracılığı ile belirli bir sıcaklıkta siyah bir gövdeden dalga boyunun bir fonksiyonu olarak elektromanyetik radyasyonun spektral radyasyonuna bir yaklaşımdır [31]. Üçüncü adımda, toz katkı değeri aşağıdaki denklem ile hesaplanır:

$$\tau_d^{m,c} = N_H^{m,c} * \kappa_{\nu_{ref}}^{m,c} * m_{H_2} * \left(\frac{1}{\zeta_{gas-dust}}\right) * \left(\frac{\nu}{\nu_{ref}}\right)^{(\beta^{m,c})} \tag{7}$$

Bu denklemde, N_H hidrojen yoğunluğunu, $\kappa_{\nu_{ref}}^{m,c}$ toz kütle bulanıklık değerini, β spektral indeksi, ν_{ref}= 230~GHz bir dalga boyunun frekansını, m_{H_2}hidrojen molekülünün kütlesini gösterir. $\frac{1}{\zeta_{gas-dust}}$ değeri toz, gaz kütle oranını belirtir ve 1/100 değeri atanır. Dördüncü adımda her molekül m ve bileşen c için optik derinliği belirleme işlemi aşağıdaki denklem ile yapılmaktadır:

$$\tau(\nu)^{m,c} = \sum \left[\frac{c_{light}^2}{8\pi\nu^2} * A_{ul}^t * N_{tot}^{m,c} * \left(\frac{g_u^t e^{-E_l^t/k_B * T_{ex}^{m,c}}}{Q(m, T_{ex}^{m,c})} \right) * \left(1 - e^{-h\nu^t/k_B T_{ex}^{m,c}}\right) \phi(\nu)^{m,c,t} \right] \quad (8)$$

Burada, A_{ul}, Einstein katsayısını, u ve l indeksleri sırasıyla üst ve alt geçiş durumunu belirtir. E_l, en düşük enerji durum; g_u, üst sınır dejenerasyon; $Q(m, T_{ex}^{m,c})$, bölüşüm fonksiyon değerleridir. Bu değerler InterpolateQGPU adlı çekirdekten elde edilen değerlerdir. $T_{ex}^{m,c}$ ve N_{tot} değerleri kullanıcı tarafından tanımlanan ve molfit dosyasından alınan uyarım sıcaklık ve yoğunluk değeridir. Bir sonraki adımda aşağıdaki denklem ile kaynak fonksiyon hesaplanır, $S^{m,c}(\nu)$ ifadesi, kaynak fonksiyonunu tanımlar ve Kirchhoff'un termal radyasyon yasasına göre yapılır.

$$S^{m,c}(\nu) = \frac{\tau_l^{m,c}(\nu), J(T_{ex}^{m,c}, \nu) + \gamma, \tau_d^{m,c}(\nu), J\left(T_d^{m,c}, \nu\right)}{\tau_l^{m,c}(\nu) + \tau_d^{m,c}(\nu)} \quad (9)$$

Şekil 6'da belirtilen diziler cihaz olarak tanımlanmıştır. GPU için yapılan ayarlamalar aşağıdaki gibidir.

```
!< list containing all observational data
ObservationalDataList_d = [TotalNumberDataPoints, 3]
!< working array for integrated line intensity
IntensityPerPeak_d = [TotalNumberOfFrequencyRanges, j, MaxNumIso, k]
!< working array for integrated line intensity
PrevIntPerPeak_d = [MaxNumIso, k]
!< working array for intensity per component calculation
LocalIntArray_d = [MaxNumIso]
!< working array for integrated line intensity
IntegHelperArray1_d = [MaxNumIso, k]
!< working array for integrated line intensity
IntegHelperArray2_d = [MaxNumIso, k]
!< working array for intensity per component calculation
TauHelperArray1_d = [MaxNumIso]
!< working array for intensity per component calculation
TauHelperArray2_d = [MaxNumIso]
```

Şekil 6:
DetectionEquation fonksiyonunun cihaz dizileri.

tBlock = dim3(MaxNumThreads, 1, 1)

grid = dim3(ceiling(real(IsoCounter)/tBlock%x), 1, 1)

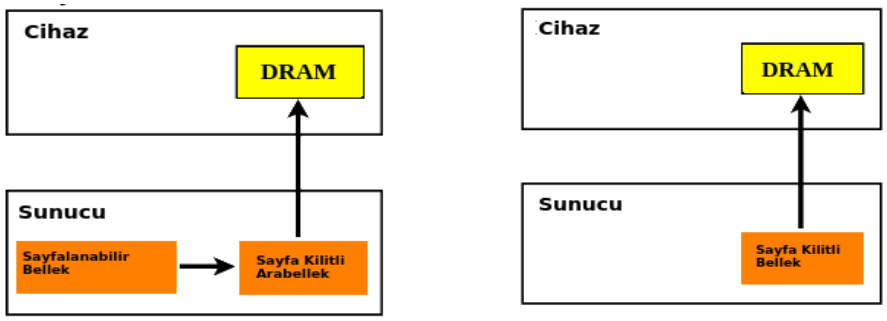
MaxNumThreads sayısı, her blokta bulunan maksimum iş parçacığı sayısını göstermektedir. Tablo 3'te özellikleri belirtilen Tesla K20m GPU kartında 1024 adet iş parçacığı bulunmaktadır. Yapılan ayarlamalar cihaz olarak tanımlanan dizilerin boyutları göz önünde bulundurularak yapılmıştır. DetectionEquation adlı fonksiyonunun içinde birbirine bağlı çok sayıda değişken olması ve bu değişkenlerin sunucu cihaz arasında bellek kopyalama işlemi sırasında çok fazla zaman alması nedeni ile tamamı "çekirdek"e dönüştürülmemiştir. Bu nedenle, hesap-yoğun

kısımları Şekil 2(b)'de yeşil renk ile belirtilen Allocation, IntensityCalculation, AllocationForeground adlı çekirdeklere dönüştürülmüştür. Elde edilen performans kazanımı, 3. bölümde ayrıntılı olarak açıklanmıştır.

2.1.5. Optimizasyon İşlemi

GPU kartlar birçok bellek türüne sahiptir. Etkili bir programlama için bu bellek türlerinin verimli bir şekilde kullanılması gerekir. Sunucu ve cihaz belleği arasında veri aktarımı ve cihaz içinde farklı bellek bölümleri arasındaki veri aktarımı olmak üzere veri transfer işlemi iki bölümden oluşur. Çalışma süresi açısından herhangi bir hızlanma göstermeyen bazı işlemler olabilir. Eğer sunucuda işlemin yürütülmesi, sunucu ve cihaz arasında çok fazla transfer gerektiriyorsa, işlemlerin sunucu üzerinde gerçekleşmesi daha avantajlı olur. Bu nedenle, en iyi uygulama performansı için, mümkün olduğunca sunucu cihaz arasındaki veri transferlerini en aza indirmek ve aynı zamanda bu tür transferler gerektiğinde optimize edildiğinden emin olmak önemlidir. Bu optimize işlemi sayfa kilitli (pinned) bellek kullanımını gerektirir, küçük aktarımları toplu olarak gerçekleştirmeyi ve veri aktarımlarını asenkron olarak gerçekleştirmeyi içermektedir. Normalde sunucuda değişkenler sayfalanabilir (pageable) bellekte bulunur. Sayfalanabilir bellek, programın sunucuda bulunan RAM'de mevcut olandan daha fazla alan kullanmasına izin vermek için programı diske götürülebilen bellektir. Veriler, sunucu ve cihaz arasında transfer edildiğinde, GPU üzerindeki Direct Memory Access (DMA) motoru, sayfa kilitli sunucu belleğini hedeflemelidir. Sayfalanabilir sunucu belleğinden GPU'ya veri aktarım işlemi sırasında, Şekil 7(a)'da gösterildiği gibi sunucu işletim sistemi önce geçici bir sayfa kilitli arabellek tahsis eder, verileri sayfa kilitli arabelleğe kopyalar ve daha sonra diziler cihaza aktarır. (Ruetsch ve Fatica, 2013) Sayfa kilitli arabellekler, cihazdan sunucuya veri transfer işlemini benzer şekilde gerçekleştirir. Eğer sunucudan cihaza transfer edilecek diziler sayfa kilitli olarak tanımlanırsa, Şekil 7(b)'de gösterildiği gibi sayfalanabilir bellek ve sayfa kilitli arabellek arasındaki transferin maliyeti önlenmiş olur ve direkt olarak bellek transfer işlemi gerçekleşir.

Bellek tahsis işleminin başarısız olduğu durumda sayfalanabilir bellekte tahsis işlemi gerçekleşir. Bu çalışmada kullanılacak olan Tesla K20 GPU kartın veri transfer hızı Tablo 2'de gösterilmiştir.



a **b**

Şekil 7:
a. *Sayfalanabilir bellekten cihaza veri transferi.* ***b.*** *Sayfa kilitli bellekten cihaza veri transferi.*

Tablo 2: Tesla K20 veri transfer hızı. (Ruetsch ve Fatica, 2013)

Sayfalanabilir Transfer		Sayfa Kilitli Transfer	
Sunucudan Cihaza bant genişliği (GB/s) :	1,6595	Sunucudan Cihaza bant genişliği (GB/s) :	5,7450
Cihazdan sunucuya bant genişliği (GB/s) :	1,5933	Cihazdan sunucuya bant genişliği (GB/s) :	6,5663
Transfer Boyutu (MB):		16.00000	

3. BULGULAR VE TARTIŞMA

Çalışma süresini ölçmek için XCLASS içerisinde bulunan fonksiyonlardan biri olan myXCLASSFit kullanılmıştır. myXCLASSFit fonksiyonun kullanılmasının sebebi çalıştığında hızlandırma işleminin yapıldığı bütün fonksiyonları çağırmasıdır. Fonksiyon, myXCLASS programı ile deneysel verileri çizgi analizi işleminden geçirmek için Levenberg-Marquardt algoritmasını kullanır. Fonksiyon, maksimum döngü sayısına ulaşıldığında veya ch^2 değeri 10^{-7}' nin altına düştüğünde durmaktadır. Bu çalışmada, gözlemsel veriyi uydurmak için Molfit dosyasında bulunan "ArH+" isimli molekülün seçilmesinin sebebi daha önce Molfit dosyasında tanımlanmış olmasıdır. Gözlemsel veri olarak Sagittarius B2 (Sgr B2) adlı büyük moleküler gaz ve toz bulutundan 617200.0- 618000.0 MHz frekans aralığındaki spektroskopik veriler kullanılmıştır. myXCLASSFit fonksiyonu sonlandığında, optimize edilmiş model spektrumunu gözlemsel verilerle birlikte gösteren bir spektrum oluşur. En sonunda modellenen spektrum ile gözlem verisi karşılaştırılır.

Çalışmada kullanılan 2 adet Tesla K20m GPU kartının özellikleri Tablo 3'te verilmiştir. Model spektrumun oluşturulduğu, DetectionEquation, InterpolateQGPU ve diğer fonksiyonların çağrıldığı ModelCalSpectrum adlı fonksiyonların CPU ve GPU üzerinde çalışma süreleri ve her bir fonksiyon için performans kazancı da yüzde olarak Tablo 4'te gösterilmiştir. Bu fonksiyonların CPU ve GPU üzerinde çalışma sürelerine bakıldığında sırası ile %87, %14 ve %62 performans kazancı olduğu görülmektedir. Yapılan hızlandırma işleminin doğruluğunu test etmek için XCLASS içerisinde bulunan test_demo-from-manual, test_demo-from-manual_Genetic+LM+Error, test_dust-continuum-fit adlı test fonksiyonları kullanılmıştır. Bu test fonksiyonlarının her bir döngüsünün hem CPU hem de GPU platformu üzerinde çalışma süresi çalışma süresi hesaplanarak Tablo 5'te gösterilmiştir. Bu tablo sonuçlarına bakıldığında da sırasıyla %98, %98 ve %93'lük bir performans kazancı elde edildiği tespit edilmiştir. Deneysel sonuçlara bakıldığında, GPU programlama tekniği ile elde edilen bu performans kazancının, astronomi gibi çok büyük boyutlu veriler ile uğraşmak zorunda olan bilim dalı için önemli bir seviyede olduğu görülmektedir.

Tablo 3: Tesla K20m GPU ekran kartının özellikleri.

Özellik	Değer
Cihaz İsmi	Tesla K20m
Bellek Hızı (KHz)	2600000
Bellek Bant Genişliği (bits)	320
En yüksek Bellek Bant Genişliği (GB/s)	208
Maksimum Grid Boyutu	2147483647x65535x65535
Maksimum Blok Boyutu	1024x1024x64
Her Bloktaki Maksimum İş Parçacığı Sayısı	1024

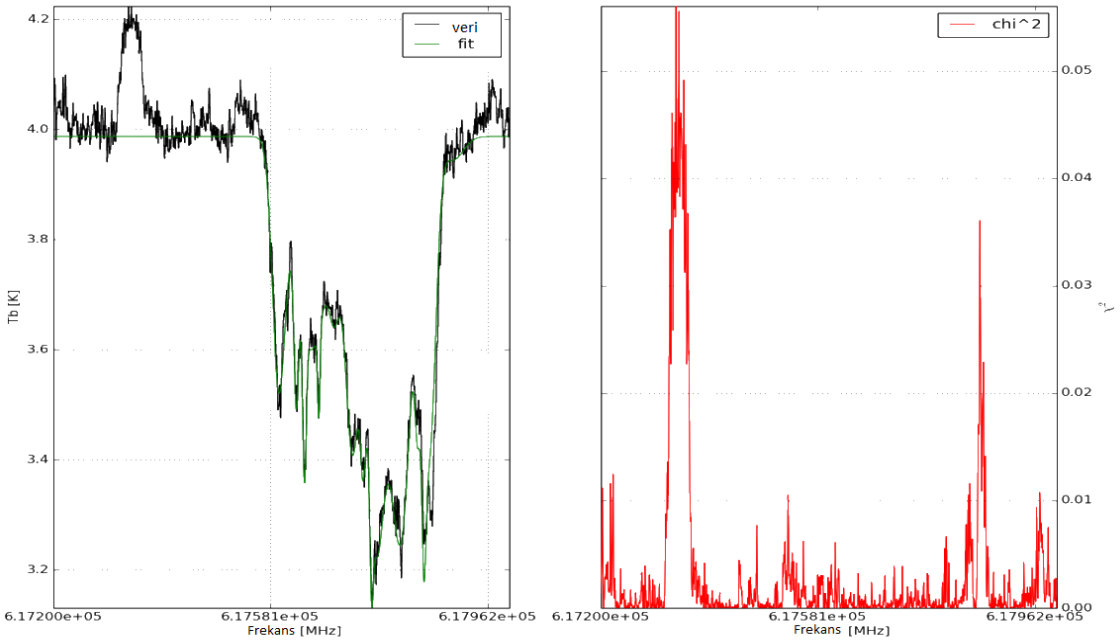
Tablo 4: Deneysel sonuçlar ve performans kazancı.

Fonksiyon İsmi	CPU'da çalışma süresi	GPU'da çalışma süresi	Performans Kazanımı
DetectionEquation	7,6293E-06 s	9,5367E-07 s	%87
InterpolateQGPU	9,1062E-05 s	7,7962E-05 s	%14
ModelCalcSpectrum	2,5320 s	0,9645 s	%62

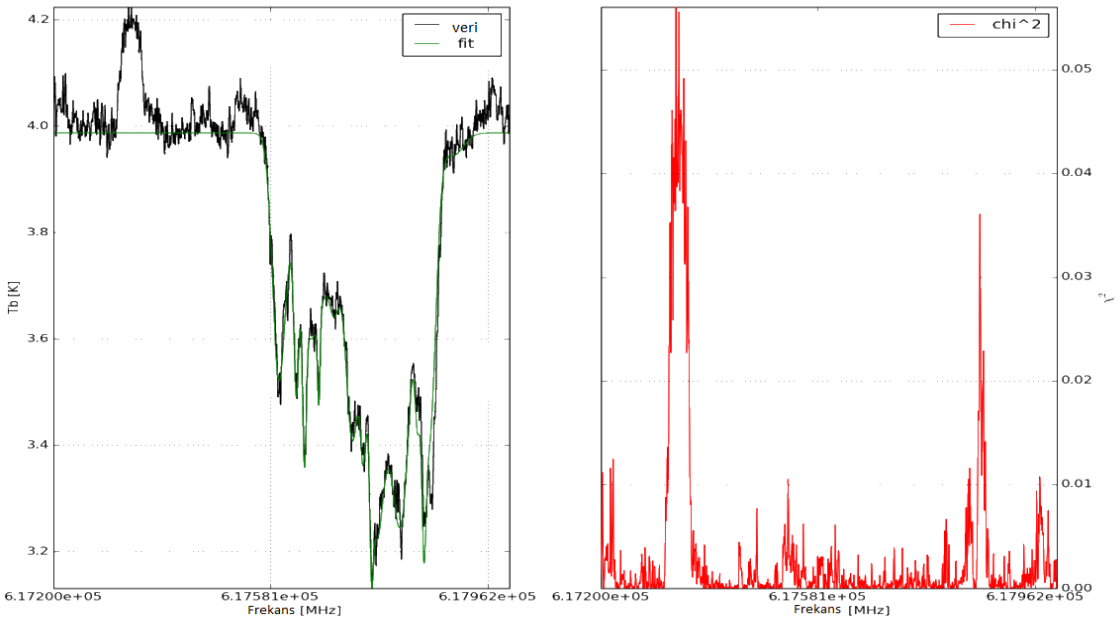
Tablo 5: Diğer test fonksiyonlarının CPU ve GPU platformu üzerinde çalışma süreleri ve performans kazancı.

Test Fonksiyon ismi	CPU'da çalışma süresi	GPU'da çalışma süresi	Performans Kazanımı
test___demo-from-manual	5,7932 s	0,0943 s	%98
test___demo-from-manual___Genetic+LM+Error	5,8041 s	0,0892 s	%98
test___dust-continuum-fit	70,4631 s	4,2856 s	%93

Şekil 8'de XCLASS'ın CPU üzerinde çalışan versiyonundan elde edilen gözlemsel veriler (siyah çizgi) ve modellenmiş spektrum (yeşil çizgi) ve çizgi analizi işlemine ait hata değerini gösteren ch^2 gösterilmektedir. Şekil 9'da GPU üzerinde çalışan versiyonundan elde edilen gözlemsel veriler, modellenmiş spektrum ve ch^2 değerleri gösterilmiştir. Şekil 8 ve Şekil 9 karşılaştırıldığında; CPU üzerinde yapılan spektrum analizinin, GPU tekniği ile gerçekleştirildiğinde de bozulmadığı; ayrıca GPU programlama tekniği kullanılarak uygulamanın daha hızlı bir şekilde gerçekleştirildiği görülmüştür.

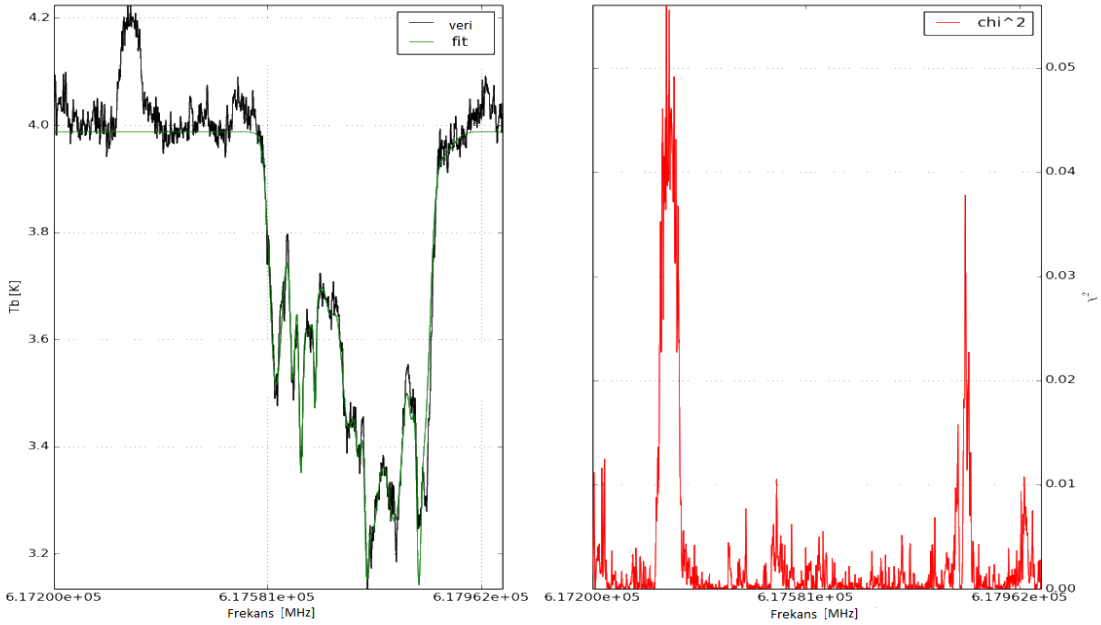


Şekil 8:
CPU platformu üzerinde ArH + molekülünün çizgi analizi işlemi (solda) ve chi² hata vektörü (sağda).

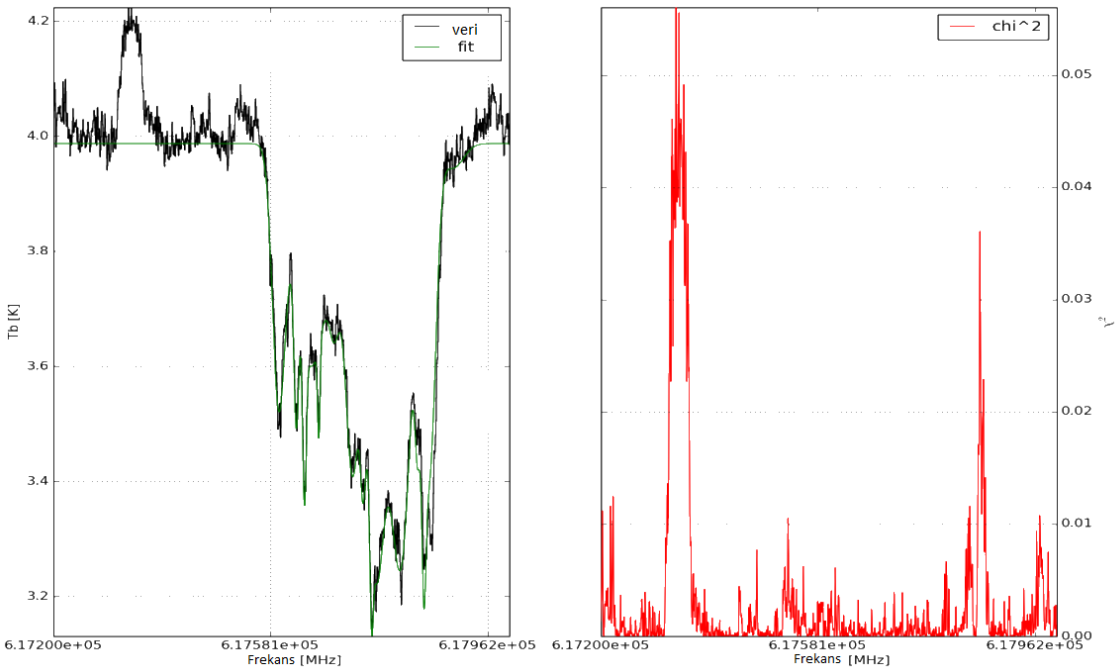


Şekil 9:
GPU platformu üzerinde ArH + molekülünün çizgi analizi işlemi (solda) ve chi^2 hata vektörü (sağda).

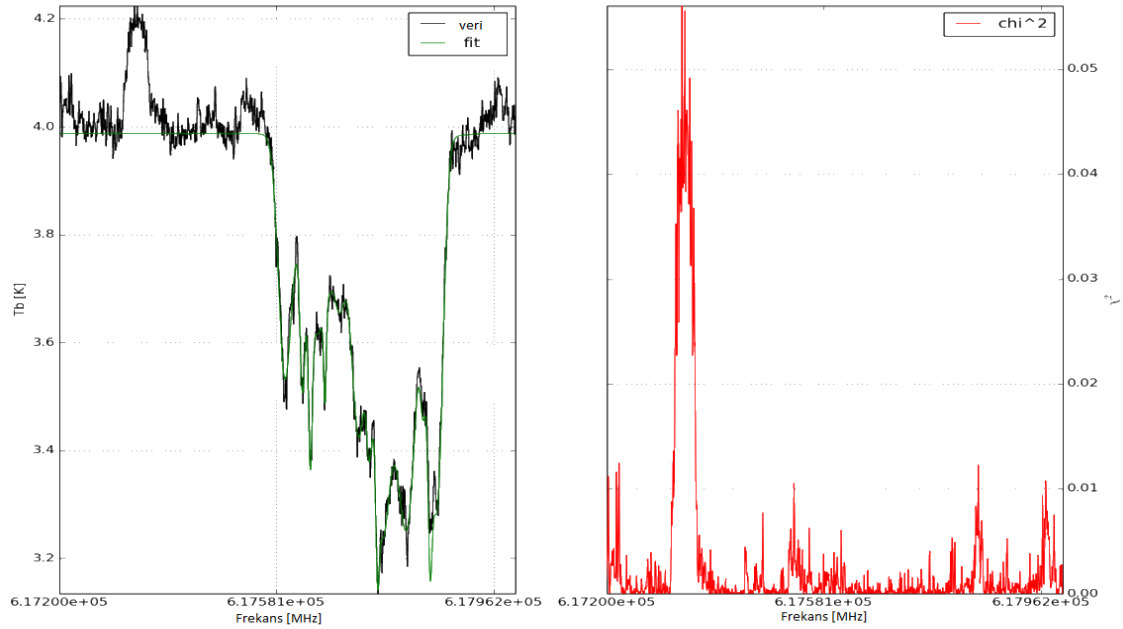
myXCLASSFit fonksiyonunu 672000 MB boyutundaki ArH+ molekülü üzerinde test etmek ve döngü sayısının çizgi analizi işlemi üzerindeki başarı oranına ve çalışma süresine olan etkisini tespit etmek için döngü sayısı sırası ile 10, 50 ve 100 olarak alınmıştır. Şekil 10, 11 ve 12'de sırası ile 10, 50 ve 100 döngü ile gerçekleştirilen myXCLASSFit fonksiyonunun sonuçları sunulmuştur. Elde edilen sonuçlar, döngü sayısı arttıkça çizgi analizi işleminin başarı oranının arttığını ve chi^2 hata değerinin azaldığını göstermektedir.



Şekil 10:
10 döngü ile yapılan modelleme işlemi.

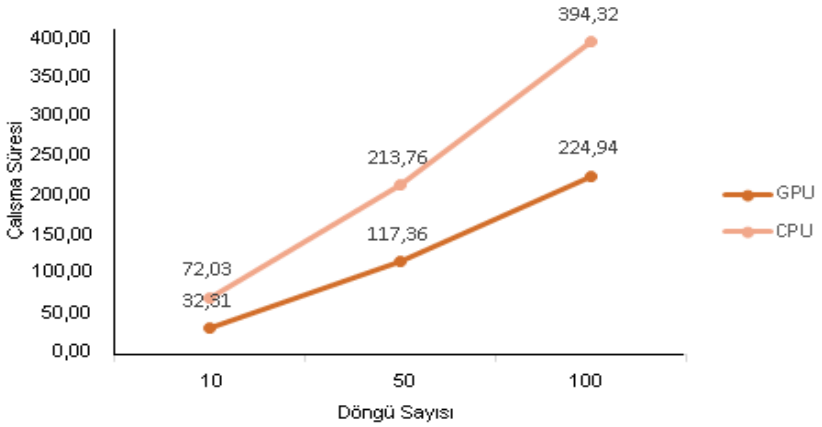


Şekil 11:
50 döngü ile yapılan modelleme işlemi.



Şekil 12:
100 döngü ile yapılan modelleme işlemi.

Tablo 6'da ise döngü sayısına bağlı olarak myXCLASSFit fonksiyonunun her iki platform üzerinde toplam çalışma süresinin değişimi ve performans kazanımı gösterilmiştir. Tablo 6' dan elde edilen verilerle Şekil 13' te oluşturulan grafik, spektrum analizinin doğruluk oranını artırmak amacı ile döngü sayısı artırılması her iki platform üzerinde de çalışma süresinde artışa neden olduğunu göstermiştir. Şekil 13' teki grafik incelendiğinde döngü sayısının artması ile CPU platformu üzerinde çalışma süresinin daha hızla arttığı gözlemlenmiştir.



Şekil 13:
GPU ve CPU platformu üzerinde döngü sayısının toplam çalışma süresine etkisinin grafiksel gösterimi.

Tablo 6: GPU ve CPU platformu üzerinde döngü sayısının toplam çalışma süresine etkisi ve döngü sayısına bağlı olarak toplam çalışma süresinin artış oranı.

Döngü Sayısı	GPU'da çalışma süresi	CPU'da çalışma süresi	Performans Kazanımı
10 döngü	32,31 s	72,03 s	%55
50 döngü	117,36 s	213,76 s	%45
100 döngü	224,94 s	394,32 s	%42

Tablo 6'da görüldüğü gibi en yüksek performans kazancı 10 döngü ile elde edilmiştir. Şekil 10, 11 ve 12'de belirtilen elde edilen sonuçlar, döngü sayısı arttıkça çizgi analizi işleminin başarı oranının arttığını ve chi^2 hata değerinin azaldığını göstermektedir. Gelecekte, XCLASS ile gerçekleştirilecek olan çok daha büyük boyutlu astronomik verilerin spektral analizi işleminde optimum performans ve güvenilir sonuçlar için XCLASS'ı geliştirenlerin de ortak görüşü ile 50 döngünün yeterli olacağı uygun görülmüştür. Daha önce muazzam boyutlardaki astronomik verileri modelleyen XCLASS gibi geniş kapsamları yazılımların performansını artırmak amacı ile yapılan herhangi bir çalışmaya rastlanmadığı için elde edilen bu performans kazanımı çok büyük önem arz etmektedir.

4. SONUÇLAR

Bu çalışmada, ALMA teleskop setinden elde edilen çok büyük boyutlu astronomik verileri modellemek amacı ile geliştirilen XCLASS'ın çok yoğun hesaplamaların yapıldığı myXCLASS adlı bölümü, GPU programlama tekniği kullanılarak hızlandırılmıştır. Veri boyutunun büyük olması nedeni ile ve yüksek performans elde etmek amacı ile, XCLASS'ın hızlandırılması işleminde iki adet Tesla K20m ekran kartı kullanılmıştır. XCLASS'ın yoğun hesaplarının gerçekleştirildiği tüm fonksiyonlarının hem GPU hem de CPU platformu üzerinde çalışma süreleri ölçülmüştür. Elde edilen çalışma süreleri XCLASS'ın hızlandırma işleminin gerçekleştirildiği tüm fonksiyonlarında GPU programlama tekniği kullanılarak önemli bir performans kazancı sağladığını göstermiştir. Ayrıca spektral analiz işleminin doğruluğunu ve güvenirliliğini artırmak amacı ile yapılan döngü sayısındaki artışın hem hata oranının değişimine ve her bir fonksiyonun çalışma süresine olan etkisi de incelenmiştir. Döngü sayısı arttıkça hata değerinin azaldığı ancak fonksiyonların çalışma süresinin arttığı gözlemlenmiştir. GPU programlama tekniği kullanılarak elde edilen sonuçlar çalışma süresi bakımından önemli performans kazancı sağlandığını göstermiştir. Son olarak, astronomik çalışmalarda önem arz

eden spektral analiz işleminde hem GPU hem de CPU platformu üzerinde aynı sonuçları elde edilmiştir, diğer bir deyişle verilerde herhangi bir bozulma olmadan GPU programlama tekniği kullanılarak hızlandırma işlemi gerçekleştirilmiştir. Böylece astronomik verilerin modellenmesi amacı ile kullanılan geniş çaplı bir yazılım paketinin ilk defa GPU programlama tekniği kullanılarak hızlandırılması gerçekleştirilmiştir.

Gerçekleştirilen hızlandırma işlemleri sayesinde, gelecekte çok daha büyük boyutlu astronomik verilerin daha kısa sürelerde işlenmesi ve daha fazla gök cisminin başarılı bir şekilde incelenmesi mümkün hale gelebilecektir. Elde edilen sonuçları daha da iyileştirmenin bir başka yöntemi ise, daha çok sayıda ya da daha üstün özelliklere sahip grafik kartları kullanılması spektral analiz işleminde performans artışında önemli bir etkiye sahip olacaktır. Ayrıca spektral analiz işleminde, GPU grafik kartlarına uygulanacak yeni algoritmalar, yeni yöntemler geliştirilebilir.

ÇIKAR ÇATIŞMASI

Yazarlar, bilinen herhangi bir çıkar çatışması veya herhangi bir kurum/kuruluş ya da kişi ile ortak çıkar bulunmadığını onaylamaktadırlar.

YAZAR KATKISI

Selçuk Sevgen – Çalışmanın Kavramsal ve/veya Tasarım Süreçlerinin Belirlenmesi, Selçuk Sevgen – Çalışmanın Kavramsal ve/veya Tasarım Süreçlerinin Yönetimi, Yasemin Poyraz Koçak – Veri Toplama, Yasemin Poyraz Koçak – Veri Analizi ve Yorumlama, Yasemin Poyraz Koçak ve Selçuk Sevgen – Makale Taslağının Oluşturulması, Selçuk Sevgen – Fikirsel İçeriğin Eleştirel İncelemesi, Yasemin Poyraz Koçak ve Selçuk Sevgen – Son Onay ve Tam Sorumluluk.

KAYNAKLAR

1. Abe Y., Sasak H., Peres M., Inoue K., Murakami K., Kato S., (2012), Power and Performance Analysis of GPU-Accelerated Systems, HotPower.
2. Arimilli R. K., Siegel D. W. (2002) Symetric Multiprocessing (SMP) System with Fully-Interconnected Heterogenous Microprocessors, International Business Machines Corpotation.
3. Astronomical Image Processing System (2011), Erişim Adresi: http://www.aips.nrao.edu/index.shtml. (Erişim tarihi: 22.12.2019).
4. Atamaca Large Millimeter/Submillimeter Array (2012), Erişim Adresi: http://www.almaobservatory.org/en/home/. (Erişim Tarihi: 25.12.2018).
5. Awan M. G. ve Saeed F., (2017), n Out-of-Core GPU based dimensionality reduction algorithm for Big Mass Spectrometry Data and its application in bottom-up Proteomics, 550-555, doi:10.1145/3107411.3107466.
6. Barsdell B. R., Barnes D. G. ve Fluke C. J. (2011) Fitting Galaxies on GPUs, Astronomical Data Analysis Software and Systems Conference, Massachusetts-USA, 451- 454.
7. Cárcamo M., Román P.E., Casassus S., Moral V. ve Rannou F.R. (2018) Multi-GPU Maximum Entropy Image Synthesis for Radio Astronomy, Astronomy and Computing, 22, 16–27, doi:10.1016/j.ascom.2017.11.003.
8. Combined Array for Research in Millimeter-wave Astronomy (2012), Erişim Adresi: https://www.mmarray.org/. (Erişim Tarihi: 22.12.2017).
9. Common Astronomy Software Applications (2016), Erişim Adresi: https://casa.nrao.edu/. (Erişim Tarihi: 24.03.2018).

10. Continuum and Line Analysis Single-dish Software (2010), Erişim Adresi: https://www.iram.fr/IRAMFR/GILDAS/doc/html/class-html/class.html. (Erişim Tarihi: 21.06.2019).

11. Diaz, J., Muñoz-Caro, C. ve Niño, A. (2012) A Survey of Parallel Programming Models and Tools in the Multi and Many-Core Era, *IEEE Transactions on Parallel and Distributed Systems*, 8-23, 1369-1386, doi: 10.1109/TPDS.2011.308.

12. eXtended CASA Line Analysis Software Suite (2016), Erişim Adresi: https://xclass.astro.uni-koeln.de/. (Erişim Tarihi: 14.03.2018).

13. George B. R., Lightman A. P. (1979) Radiative Processes in Astrophysics, Haruard-Smithsonian Center for Astrophysics, Germany.

14. Grenoble Image and Line Data Analysis Software. (2010), Erişim Adresi: https://www.iram.fr/IRAMFR/GILDAS/doc/pdf/gildas-intro.pdf. (Erişim Tarihi: 22.06.2019).

15. Hall C. A., Meyer W. W. (2004) Optimal Error Bounds for Cubic Spline Interpolation, Journal of Approximation Theory, 2-**16,** 105–122, doi*:* 10.1016/0021-9045(76)90040-X.

16. Image Reduction and Analysis Facility (2014), Erişim Adresi: http://ast.noao.edu/data/software. (Erişim Tarihi: 21.12.2019).

17. Jang H., Park A. ve Jung K., (2008) Neural Network Implementation Using CUDA and OpenMP, *Digital Image Computing: Techniques and Applications*, Canberra, ACT, 155-161, doi: 10.1109/DICTA.2008.82.

18. Mei Y., Wang F., Wang W., Chen L., Liu Y., Deng H., Dai W., Liu C. ve Yan Y. (2017) GPU-Based High-performance Imaging for Mingantu Spectral RadioHeliograph, Instrumentation and Methods for Astrophysics, 130, 983, doi: 10.1088/1538-3873/aa9608.

19. NVIDIA and PGI Compiler. CUDA Fortran (2007), Erişim Adresi: https://developer.nvidia.com/cuda-fortran. (Erişim Tarihi: 08.04.2018).

20. NVIDIA and PGI Compiler. Introduction to PGI CUDA Fortran (2008), Erişim Adresi: http://www.pgroup.com/lit/articles/insider/v1n3a2.htm. (Erişim Tarihi: 10.04.2018).

21. NVIDIA Corporation (2017) PGI Compilers and Tools. CUDA Fortran Programming Guide and Reference.

22. NVIDIA CUDA C Programming Guide (2010), Erişim Adresi: https://developer.download.nvidia.com/compute/DevZone/docs/html/C/doc/CUDA_C_Programming_Guide.pdf. (Erişim Tarihi: 20.03.2018).

23. Plateau-de-bure (2010), Erişim Adresi: http://www.iram-institute.org/EN/plateau-de-bure.php/. (Erişim Tarihi: 22.12.2018).

24. Ruetsch G., Fatica M. (2013) CUDA Fortran for Scientists and Engineers, NVIDIA Corporation, Santa Clara, USA.

25. Sánchez-Monge Á. (2018) XCLASS: Automatic Line Fitting of ALMA Data Introduction and Tutorial, I. Physikalisches Institution Universität zu Köln.

26. Submillimeter Array (2005), Erişim Adresi: http://sma1.sma.hawaii.edu/. (Erişim tarihi: 24.12.2018).

27. Very Large Array, (2008), Erişim Adresi: http://www.vla.nrao.edu/. (Erişim Tarihi: 24.12.2018).

28. Westerlund S. ve Harris C. (2015) Performance analysis of GPU-accelerated filter-based source finding for HI spectral line image data, Experimental Astronomy, 1-39, 95–117, doi: 10.1007/s10686-015-9445-2.