Fonksiyonel ve Imperative Programlama ile Sıralama

Elis Soylu[1], Muammer Akçay[2]

[1] Eskişehir Osmangazi Üniversitesi, Fen Edebiyat Fakültesi Matematik- Bilgisayar Bölümü, Eskişehir

[2] Dumlupınar Üniversitesi, Mühendislik Fakültesi, Bilgisayar Mühendisliği Bölümü,Kütahya

esoylu@ogu.edu.tr, muammer.akcay@dpu.edu.tr

Özet: Sıralama algoritması bilgisayar ortamında oldukça önemli bir yere sahiptir. Farklı donanım veya yazılım yapılarıyla bu algoritma önemli gelişmeler kaydetmektedir. Yazılım anlamında nesnel ve fonksiyonel diller bu algoritmanın paralelleştirilmesinde önemli bir yer tutar. Nesnel olarak MPI (Message Passing Interface) ve CUDA (Compute Unified Device Architecture) dilleri, fonksiyonel olarak Haskell ve Erlang dilleri yapısal açıdan bu algoritma doğrultusunda karşılaştırılacaktır.

Anahtar Sözcükler: Sıralama, Fonksiyonel diller, Nesnel diller, CUDA, Haskell, MPI, Erlang, Paralel Programlama.

Sorting with Functional and Imperative Programming

Abstract: Sorting algorithm is very important place in computing. This algorithm is developed with different software and hardware structures. Object-oriented and functional languages are important for parallelizing this algorithm in software. In this paper, MPI and CUDA for object-oriented, Haskell and Erlang for functional languages is compared with respect to structural view for this algorithm.

Keywords: Sorting, Functional Language, Object-oriented language, CUDA, Haskell, MPI, Erlang, Parallel Programming, Structural view.

1. Giriş

Bilgisayar programlama, insan tarafından yapılacak herhangi bir işin veya hesaplamanın kısa bir sürede, enerji harcamadan gerçekleştirilmesini sağlayan bir yöntemdir. Gerek sürenin kısaltılması gerek de yazılım maliyetleri açısından çeşitli programlama dilleri ortaya çıkmıştır.

Nesne tabanlı, yordamsal, bildirimsel, fonksiyonel programlama genel adları altında toplanan diller farklı yaklaşımlar ile en etkin programı yazmayı amaçlamışlardır. Hem çağın gelişen teknolojisi hem de en zor problemlerin çözüm yöntemleri açısından bilgisayarlar ortak bir çalışma ortamı olarak göz önüne alınırsa, büyük gelişmeler katetmiştir. Özellikle çok çekirdekli modern bilgisayarların üretilmesi ile farklı programlama mantıkları büyük önem kazanmıştır.

Çok çekirdekli bilgisayarlar ile eşzamanlı ortak iş yapılması hedeflenmiştir. Dolayısıyla ortaya çıkan birçok programlama dili bu açıdan bakılırsa, uyumlu veya uyumsuz birçok farklı yönteminin olduğu ortaya çıkar.

Çekirdek sayısının az olduğu durumlarda etkin bir şekilde çalışabilen bir programlama dili, çekirdek sayısının artmasıyla bu etkinliğini kaybedip bazı sorunlara yol açabilir. Dolayısıyla yeni çekirdek yapıları ve eşzamanlı çalışma prensibiyle bu sistemin altyapısını oluşturabilecek uygun bir programlama dili gereklidir.

2. Sıralama Algoritması

Sıralama algoritması, genel olarak matematikten güç alıp bilgisayar bilimlerine fayda sağlayan belli koşullar altında sıralama yaptıran önemli bir tekniktir. Sıralama yapılırken en çok kullanılan koşullar sayı büyüklüğü ve alfabetik kısıtlardan oluşmaktadır. Sıralama, genel olarak bir veri yığınının düzenlenebilme ve insanlar tarafından kolay algılanabilme özellikleri açısından vazgeçilmez bir yapı taşı halini alır.

Algoritma yapısı çok sade ve basit olarak çözülüp, anlaşılsa da sıralama işi karmaşık gerçekleştirildiği için üzerinde birçok yöntem geliştirilmiştir. Bu nedenle her adımda yeni bir algoritma mantığı bulunmuş olur [1].

Sıralama mantığına göre, sıraya dizilecek elemanlar bellekte yer alıyorsa içsel (internal), verilerin bazıları ikincil bellekte ise dışsal (external) sıralama olarak bilinir.

En çok bilinen sıralama türleri kabarcık sıralaması (bubble sort), hızlı sıralama (quicksort), seçmeli sıralama (selection sort) ve birleştirmeli sıralama (merge sort) şeklindedir.

Kısaca bu algoritmalara değinilecek olursa;

- Kabarcık sıralama, ilk elemandan başlayarak ve her geçişte yan yana bulunan elemanları inceleyen ve sıraya sokan bir sıralama çeşididir.
- Hızlı sıralama, veriler için rastgele bir x elemanını belirleyip sol tarafına kendinden küçükleri, sağ tarafına ise büyükleri yerleştirerek kurulur.
- Seçmeli sıralama, verilerin 1.,2., 3. ,…,n sıradaki elemanları ile en küçük, en küçük 2. eleman,…,n şeklindeki elemanları yer değiştirilerek kurulur.
- Birleştirmeli sıralamada ise, veri yapısı ikişerli parçalar haline inene kadar bölünerek, bu parçaların kendi içlerinde sıralanmasıyla birleştirme yapılarak elde edilir [2].

3. Paralel Fonksiyonel Programlama

Paralellikteki mantık, yapılması gereken işi paylaştıran, çeşitli birimlere veren ve bu işi eşuyumlu olarak gerçekleştiren etkin bir programlama çeşididir. Paralel programlama, görevler arasında uygun olarak tanımlanan ve etkinliği arttırmak için görevlerin paralel hale gelmesini sağlayan özel bir yapıdır.

Paralel programlama mantığının önündeki en büyük engel, eldeki bir verinin değerini diğer işlemcilere dağıtabilmek için çoğaltma ve değiştirme yetkisine sahip olmasıdır. Özellikle bir verinin değerinin kaybına sebep olur. Bu anlamda anlık veri değerlerinin de tutulabildiği yani sabit veri yapısını kullanabilme yeteneği bu engeli ortadan kaldıracaktır. Böylece bir paralel program bu noktada fonksiyonel bir dil ile kesişir. Fonksiyonel bir dilin temelindeki değişmez veri türleri yapısal bakımdan paralel programlamada ihtiyacı karşılayarak açığı kapatabilir.

Fonksiyonel bir dilin, özellikleri bakımından paralelliğe olan katkıları şu şekildedir:

- Ortak sınıf yapısı olarak fonksiyonlar kullanıldığından parametreler aynı olduğu sürece aynı sonuçlar elde edilir.
- Fonksiyonlar yapısal olarak kendi verisi dışındaki verilerde değişiklik yapamaz. Dolayısıyla yan-etki durumuyla karşılaşılmaz.
- Fonksiyonel dilde bir değişkene atanan değer program sonuçlanıncaya kadar bir değişikliğe uğramaz. Değişikliğin yapılmasının zorunlu olduğu durumlarda özyinelemeli (recursive) yapılar kullanılır.
- Fonksiyonel dildeki bir fonksiyon oluşturduğu değer ile birebir eşlenmiştir. Dolayısıyla tersine bir durum kolaylıkla elde edilebilir.
- Bu dil sayesinde farklı bir bakış açısı sunmasından dolayı matematiksel bir yorum katar.
- Daha kısa ve anlaşılabilir kod yazımı sayesinde de büyük bir avantaj sağlar.

Fonksiyonel dil olarak sayılabilecek birçok dil içinde Haskell ve Erlang önemli bir yere sahiptir. Paralelleştirmenin kolaylıkla görüldüğü bu dillere kısaca göz atalım [3].

3.1 Haskell

Fonksiyonel dillerden biri olan Haskell, herhangi bir yazılımı modellerken yapının temelinin tamamen fonksiyonlardan oluşmasını kullanır. Veri yapılarının içeriğini değiştirmek yerine onlardan yeni veri yapıları türeten fonksiyonlar oluşturur. Yani herşey bir fonksiyondur.

Haskell dili tanımlı temel tip ve fonksiyonlar ile kullanıcı tarafından tanımlanan tip ve fonksiyonlardan ibarettir. Haskell dilinin elemanları,

- Temel veri tipleri
- Her bir veri tipinin sabitleri
- Veri tipleri arasındaki fonksiyonlar
- Veri tiplerine ve fonksiyonlara uygulanan yapıcılar

olarak listelenebilir. Bu dilde, yeni tip ve fonksiyonlar yapıcılar (constructors) yardımıyla üretilir. Yapıcılar temel olarak tip yapıcısı ve fonksiyon yapıcısı olmak üzere ikiye ayrılır. Genel olarak bu yapıcılar sayesinde bir veri tipindeki değer değişikliğine ulaşılır [4].

3.2 Erlang

Erlang, yüksek geçerliliğe sahip oldukça fazla büyüklükteki işlemlerin gerçekleştirilmesini sağlayan bir programlama dilidir. Günlük yaşamda, haberleşme, bankacılık, e-ticaret, anlık mesajlaşma gibi alanlarda sıklıkla karşılaşılır. OTP (Open Telecom Platform); sistem gelişimi için tasarlanmış özel bir Erlang kütüphanesidir. Bu yapı, kendi dağıtık veritabanını, diğer dillerin arayüzü uygulamalarını, hata ayıklama araçlarını barındırır.

Erlang dili, Ericsson firmasının dili olarak bilinir. Daha çok haberleşme alanında bu dil kullanılır ve daha çok tanınır [5].

4. Nesnel Programlama

Nesnel programlama, bir programlama dilinin yaklaşım tarzıdır. Bu programlama çeşidi, çözülecek problemi parçalayarak nesne üzerinden çözüm yöntemi geliştirir. Nesne mantığı, daha sonra da kullanılma olasılığına karşın genel bir yapıda toplanmasına dayanır. Parça parça elde edilen çözüm adımları çözüme bütün bir şekilde ulaşabilmek için bir araya getirilir. Günlük yaşamdan verilen en güncel örnek; arabanın oluşumudur. Arabayı oluşturabilecek her bir temel bileşen bir araya getirilerek amaca uygun hizmet eder. Buradaki her bir temel bileşen de nesnel programlamanın nesneleriyle birebir eşlenebilir [6].

4.1 MPI

MPI, genel olarak program çalışma süresini kısaltacak paralelleştirme mantığında kullanıcıya kolaylık sağlayan mesaj gönderim ara yüzüdür. C, Fortran dilleri yardımıyla MPI ile haberleşme ortamı kurulur. Haberleşme olayı MPI ara yüzü yardımıyla mesaj gönderme-alma şeklindedir. MPI programının mesajı alan, gönderen, ortamdaki bilgisayar sayısı, bilgisayar sırasını bulan fonksiyonları temel olarak her kod bloğunda yer alır.

Kaynak '0' olarak numaralandırılarak yani verinin yer aldığı ve dağıtılacağı bilgisayar numarası verilerek numaralandırma yapılır. Koşul ifadeleri yardımıyla da bileşen bilgisayarlara veri dağıtımı mesaj yoluyla gerçekleşir. Benzer yolla da verilerin toplanması esnasında mesajlar toplanarak işlenmiş veri elde edilir [7].

4.2 CUDA

CUDA, NVIDIA firması tarafından bilgisayarın ekran kartı bileşenini etkin şekilde kullanmaya yardımcı olan bir mimaridir. Ekran kartı üzerinde bulunan çok sayıdaki küçük işlemciler bilgisayarda gerçekleştirilen temel aritmetik hesaplamalar haricinde paralel hesaplamaya da yardımcı olur. Bu aşamada ekran kartındaki işlemci yapısında yer alan 'thread'

ler yardımıyla görev dağılımı yapılarak paralel hesaplama elde edilir. Ekran kartı işlemcileri, paralel hesaplamaya uygun basit yapılarına zıt olarak kontrolü güç algoritma yapılarıyla sıkıntılı durumlar oluşsa da hız, etkinlik, süre bakımından oldukça etkilidir. Büyük derecedeki hesaplamalar CUDA mimarisi sayesinde kolayca elde edilir [8].

4.3 Hızlı Sıralama Algoritmasının Seçilen Dillerdeki Performansları

Hızlı sıralama algoritması genelde n elemanlı bir diziyi O(nlog n) karmaşıklığıyla sıraya koyar. Sıralamanın bozuk bir şekilde başladığı durumlarda ise en kötü O(n2) süreye kadar işlem yapar [9].

Hızlı sıralama algoritması:

- Fonksiyon qsort(dizi)
- kucuk, eşit, buyuk listelerini oluştur
- Boyut(dizi)<1 ise diziyi döndür.
- Diziden bir pivot değer belirle
- Dizideki her x elemanı için
  X < pivot ise kucuk listesine al,
  X = pivot ise esit listesine al,
  X > pivot ise buyuk listesine al,
- qsort(qsort(kucuk),esit,qsort(buyuk)) için döngüyü tekrarla.

n=1.000.000 verili rastgele tamsayılardan oluşan bir liste için:

Haskell dilinde sıralama algoritmalarından hızlı sıralama algoritmasına bakılırsa; Intel(R) Core(TM) 2 Quad CPU, Ram belleği 3.0 Ghz olan Windows işletim sistemli bilgisayarda 100 çalıştırma sonucunda 2.954869 saniyede gerçekleşir [10].

Erlang dilinde; benzer şekilde Intel 2 Quad Core 4 çekirdekli işlemci ile 4.14 saniyede sıralanmış veri elde edilir. Bütün veri setini en başından karşılaştırmaya başladığı için derleme süresi uzar [11].

7 çekirdekli Intel Xeon E5420 işlemcili 16GB bellekli Linux cluster'ı içindeki bilgisayarlar üzerinde veri MPI ile 1.25 saniyede sıralanmaktadır [12].

CUDA dili için yine 4 işlemcili Intel 2 Quad Core ekran kartı için aynı veri seti 1.6 saniyede sıralanmaktadır [13]. Bu analizler Şekil 1 tablosunda aşağıdaki gibi verilmiştir.

Programlama Dilleri	Donanım	Süre (saniye)	Veri seti
Haskell	Intel Core 2 Quad	2.9548	10^6
Erlang	Intel Core 2 Quad	4.14	10^6
MPI	Intel Xeon E5420 7 Core 16 GB	1.25	10^6
CUDA	Intel Core 2 Quad	1.6	10^6

Şekil 1: Seçilen Programlama Dillerinin Hızlı Sıralama Algoritma Analizleri

4.4 Yapısal Karşılaştırma

Fonksiyonel ve imperative diller yapısal olarak farklı yöntemler uygulamaktadır. Fonksiyon ve nesnel taban mantığı bu iki dil yapısını birbirinden ayırır. Bazı yapısal özellikler ile fonksiyonel dillerin avantajı çoktur.

5. Sonuç ve Öneriler

Nesnel ve fonksiyonel dillerde paralelliğin karşılaştırılmasıyla bir algoritma üzerinde sonuçlar alınmıştır. Benzer şekilde farklı algoritma yapılarının aynı dil üzerindeki etkileri ve farklılıkları da gözlenecektir.

5. Kaynaklar

[1] www.vikipedi.org.tr

[2] enformatik.kku.edu.tr/uygulamalar

[3] http://en.wikipedia.org/wiki/Functional_programming

[4] D.Hünniger, Haskell, 2012

[5] http://www.erlang.org/

[6] Mesut, A. , Programlama Dilleri, 2011

[7] Kaleci,D. ,Şahin A., Kaya O.A., MPI ile Paralel Programlamanın Temelleri, Akademik Bilişim'09, 2009

[8] http://www.nvidia.com/object/cuda_home_new.html

[9] Capannini, G.,Silvestri, F., Baraglia, G. , Sorting on GPUsforlargescaledatasets: A throughcomparison, Information Processing and Management, 2011

[10] Jones, D. , Marlow, S., Singh, S., Parallel-PerformanceTunningforHaskell , ACM SIGPLAN symposium on Haskell, 2009

[11] Patterson, M.,Sagonas, K., Johanson, E., TheHiPE/x86 Erlang Compiler: SystemDescriptionandPerformance Evaluation, 2012

[12] http://www.codeproject.com/Articles/42311/Parallel-Quicksort-using-MPI Performance-Analysis

[13] http://on-demand.gputechconf.com