3-Way Merge Sort: Algoritma, Kodlama ve Performans Karşılaştırması

Kerem ÜLLENOĞLU kerem.ullen@proton.me

Giriş

3-way merge sort, klasik birleştirme sıralamasının optimize edilmiş bir varyasyonudur. Bu algoritma, diziyi iki yerine üç parçaya bölerek işlem süresini iyileştirmeyi hedefler. Büyük veri kümelerinin sıralanmasında özellikle yararlı olan 3-way merge sort, paralel programlama ile birleştirildiğinde daha da etkili hale gelir.

Bu çalışmada, algoritma Haskell, Go ve Python dillerinde uygulanmış, paralellik (goroutines) dahil çeşitli yaklaşımlar ile performansları değerlendirilmiştir. Performans ölçümleri; işlem süresi ve bellek kullanımını kapsar.

3-Way Merge Sort Algoritması

Klasik merge sort, diziyi iki parçaya ayırırken 3-way merge sort algoritması ise eşit 3 parçaya böler. Bölümler ayrı ayrı sıralanır ve en sonda birleştirilir. Bölümler sıralandıktan sonra birleşim işlemi daha karmaşık hale gelir ancak genellikle daha verimlidir.

Algoritmanın adımları genel olarak şu şekildedir: Diziyi üç alt diziye böl. Alt dizileri ayrı ayrı sırala (recursive). Sıralanmış üç alt diziyi birleştir.

Algoritmanın psuedo koduna bakacak olursak şu şekilde yazılabilir:

```
function three_way_merge_sort(arr):
    if len(arr) <= 1:
        return arr
    split arr into three parts: left, middle, right
    left_sorted = three_way_merge_sort(left)
    middle_sorted = three_way_merge_sort(middle)
    right_sorted = three_way_merge_sort(right)
    return merge(left_sorted, middle_sorted, right_sorted)
```

Algoritmanın zaman karmaşıklığı $O(nlogn)$ ve alan karmaşıklığı $O(n)$ olarak belirlenmiştir.

Algoritmanın Uygulamaları

Bu çalışmada algoritmanın uygulamalarını incelemek için üç farklı dil kullanılacak. Bu diller, farklı paradigmaları temsil ederek algoritmanın çeşitli bağlamlardaki performansını değerlendirmek için seçilmiştir. **Haskell**, fonksiyonel bir yaklaşım sunarak rekürsiyon tabanlı algoritmaların matematiksel netliğini gösterir. **Go**, paralel işlemeyi destekleyen modern bir dil

olarak, büyük veri setlerinde hız ve verimlilik avantajlarını test etme imkânı sunar. **Python**, yaygın kullanımı ve öğrenme kolaylığı nedeniyle pratik uygulamaları hızlıca geliştirmek için uygundur.

Haskell

Haskell, saf fonksiyonel bir programlama dilidir. Tüm işlemler saf fonksiyonlar ve immutable veri yapıları ile gerçekleştirilir. Bu yapı, 3-way merge sort gibi rekürsif algoritmaların uygulanmasında matematiksel bir açıklık ve güvenlik sağlar. Ayrıca, Haskell'in yüksek seviyeli fonksiyonel ifadeleri ve tür sisteminin sağlamlığı, sıralama algoritmalarını daha az hata ile tanımlamaya olanak tanır. Ancak, Haskell'in performansı, imperative diller kadar optimize edilemeyebilir ve öğrenme eğrisi biraz daha dik olabilir.

Go

Go, hafif iş parçacıkları (goroutines) ve yerleşik paralel işlem desteğiyle ön plana çıkar. Paralel programlama, Go'nun tasarımının merkezinde yer alır ve 3-way merge sort gibi böl ve birleştir temelli algoritmalarda doğal bir uyum sağlar. Go'nun minimal sözdizimi, paralel işlemlerin uygulanmasını kolaylaştırır.

Python

Python, yalın ve okunabilir sözdizimiyle hızlı prototipleme ve basit algoritmaların uygulanmasında tercih edilen bir dildir. 3-way merge sort, Python'un dinamik veri tipleri ve list manipülasyonu yetenekleriyle kolayca uygulanabilir.

Metodoloji

Test ortamı ve donanım

- **İşlemci:** AMD Ryzen 7 5800H
- **RAM:** 13.9GB 3200MHz
- **İşletim Sistemi:** Windows 10 Home (19041.1.amd64fre.vb_release.191206-1406) (*bazı testler için WSL2 Ubuntu yardımı alınmıştır)
- **Python:** Python 3.10.6
- **Go:** go version go1.22.3 linux/amd64
- **Haskell:** The Glorious Glasgow Haskell Compilation System, version 9.4.8

Test edilen veri setleri

Performans testi, üç farklı dizi boyutuyla yapılmıştır: 10.000, 100.000, 1.000.000. Bu diziler, rastgele tamsayılardan oluşturulmuş ve sıralama işlemine tabi tutulmuştur.

Test edilen değerler ve performans ölçüm yöntemi

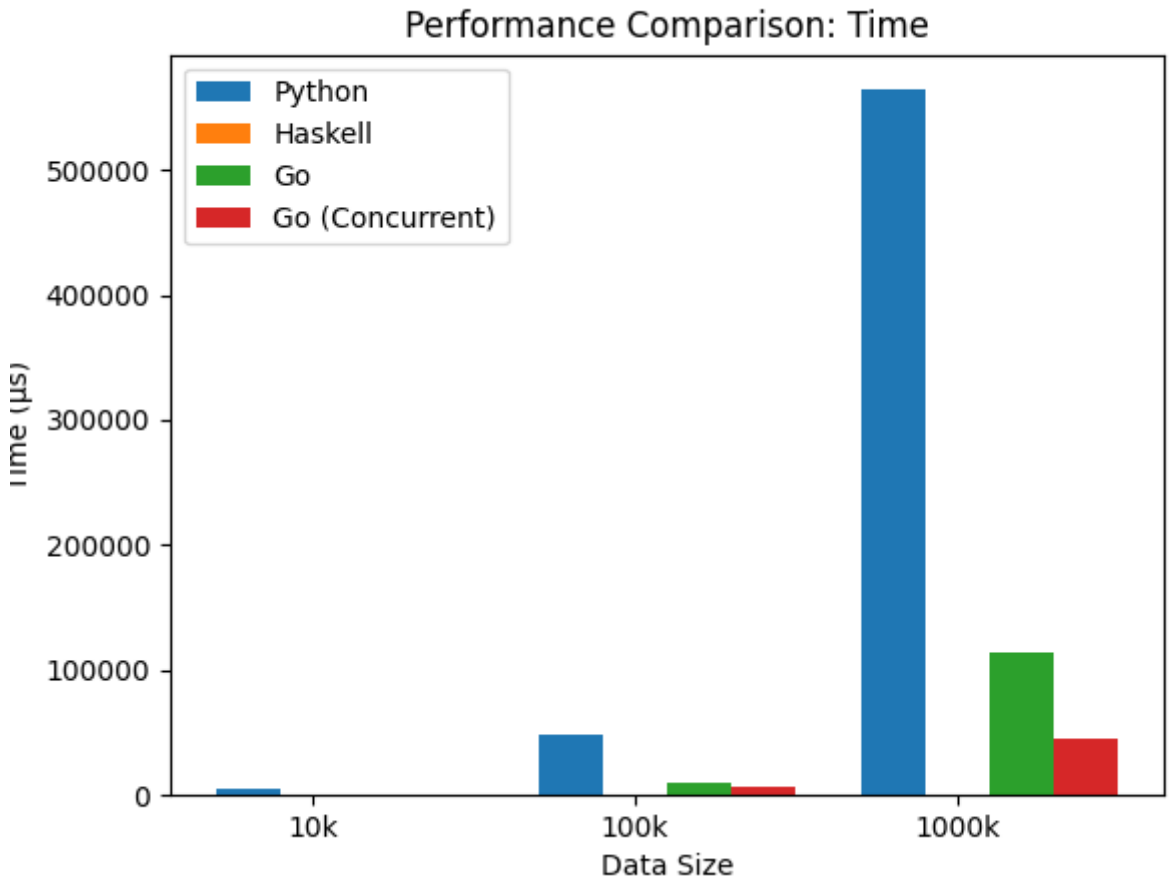
Her dilde, paralel olmayan 3-way merge sort ve Go dilinde ayrıca paralel versiyonları test edilmiştir. Performans karşılaştırmalarında, yalnızca algoritmanın sıralama süresi ölçülmüştür. Bu testlerde, her dilin verimli sıralama performansını kıyaslamak için **işlem süresi** (sıralama işlemi için tamamlanması için geçen süre, microseconds cinsinden) ve **bellek kullanımı** (test sırasında kullanılan toplam bellek miktarı, bytes cinsinden) göz önünde bulundurulmuştur. Her test, her veri seti boyutu için 5 kez çalıştırılmış ve sonuçlar ortalanmıştır.

Algoritma ve testler en optimize halinde olmayabilir. Kullanılan kodlar https://github.com/battos/3-way-merge adresinde bulunabilir.

Sonuç ve Değerlendirme

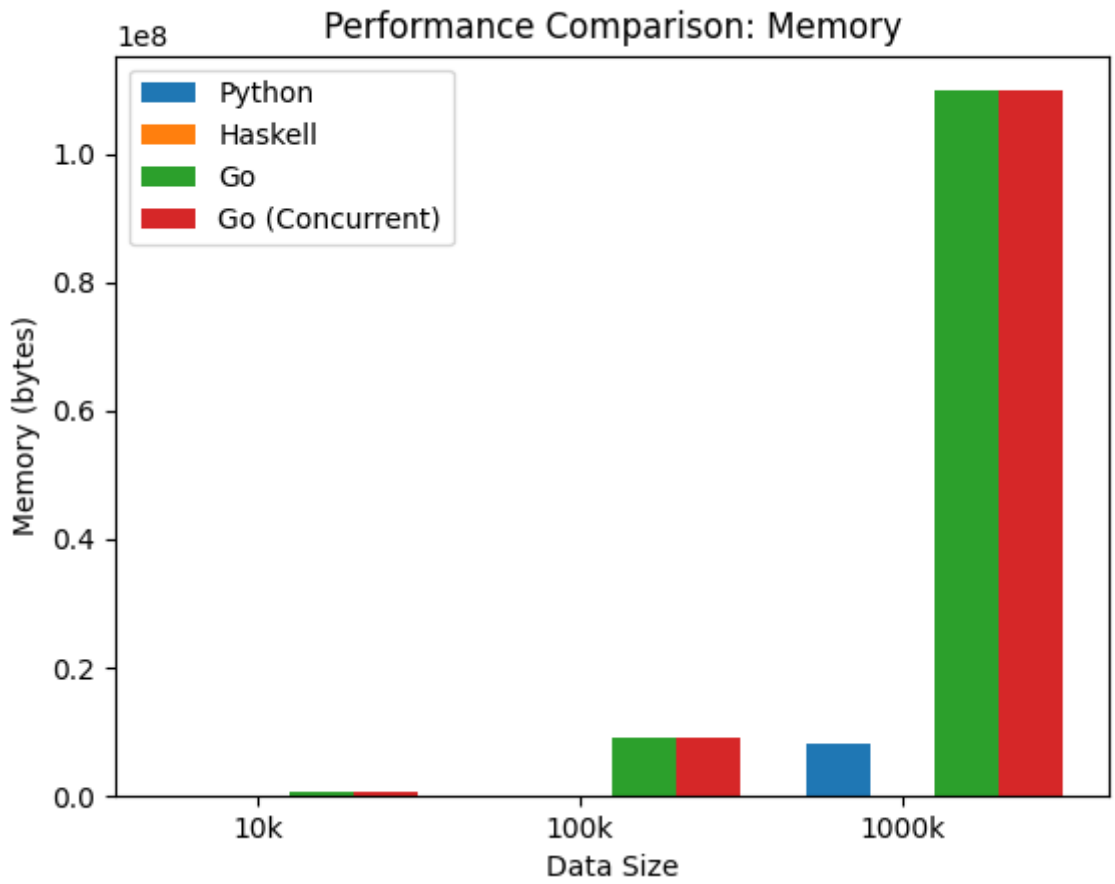
Zaman Karşılaştırmaları

	10.000	**100.000**	**1.000.000**
Python	4482.86µs	48282.16µs	564100.80µs
Haskell	0.072µs	0.0638µs	0.06598µs
Go	964 µs	10615 µs	114214 µs
Go (Concurrent)	459 µs	6328 µs	45187 µs



Bellek Karşılaştırmaları

	10.000	100.000	1.000.000
Python	50790.4 bytes	174489.6 bytes	8032256.0 bytes
Haskell	?	?	?
Go	737088 bytes	9060784 bytes	109763520 bytes
Go (Concurrent)	738624 bytes	9070256 bytes	109822816 bytes



Sonuç

Bu çalışmada, 3-way merge sort algoritmasının farklı programlama dillerindeki (Haskell, Go, Python) performansları karşılaştırılmış ve paralel programlama desteğinin etkileri değerlendirilmiştir. Test sonuçlarına göre:

- Haskell, tüm veri setlerinde açık ara en hızlı sonuçları vermiştir. Bunun sebebi, Haskell'in saf fonksiyonel yapısı ve rekürsiyon tabanlı algoritmalar için ideal olan optimize edilmiş derleyicisidir. Go, paralel sürümüyle Haskell'e yaklaşabilse de Python en yavaş performansı göstermiştir.

- Go'nun paralel ve paralel olmayan sürümleri, bellek tüketimi bakımından neredeyse aynı sonuçları vermiştir. Bu durum, Go'nun paralel programlama özelliklerinin hafifliği ve etkin kaynak kullanımıyla açıklanabilir. Python ise Go'ya göre daha az bellek kullanmış, ancak bu düşük bellek kullanımı düşük hızla dengelenmiştir.

- Go'nun goroutines desteği, büyük veri setlerinde paralel sıralamada belirgin avantajlar sunmuş ve sıralama süresini önemli ölçüde düşürmüştür.

Genel olarak, 3-way merge sort algoritması, paralel programlama ile daha da optimize edilebilse de, Haskell en hızlı dil olarak öne çıkmıştır. Bu durum, Haskell'in akademik ve teorik uygulamalardaki gücünü bir kez daha kanıtlamaktadır. Go, özellikle paralel programlama desteğiyle büyük veri projelerinde pratik ve etkili bir çözüm sunarken, Python ise kolay uygulanabilirliğiyle eğitim ve prototipleme amaçlı tercih edilebilir.