MARMARA ÜNİVERSİTESİ

**İşletim Sistemleri** **Özer Öztürk**
**2023-2024** **Gürkan Çelen**

# İşletim Sistemleri Proje Sonuç Raporu

## 1.Giriş

### 1.1.Amaç

Projenin amacı, yürütülmekte olan birden fazla process/thread arasındaki iletişimi sağlamanın en verimli yolunu bulmaktır.

### 1.2.Yaklaşım

Verimi ölçmek için üç farklı yöntem kullanılmıştır. Yerine getirilmesi gereken işlev, her bir thread/process'in n tane dosyadan değerleri alıp sıralaması ve parent processe k. en büyük değerleri ileterek parent process/thread'in bunları tekrar sıralayıp belirtilen outfileye yazdırılmasıdır. Bu işlevi yerine getirirken test dosyadaki veri miktarı ile k ve n parametreleri ayrı ayrı değişken olarak kullanılmıştır. Sonuçlar ile time plot grafikleri oluşturulmuştur.

### 1.2.Yöntem

Kullanılan yöntemler, dallanan birimlerin k. değeri ana birime gönderirken kullandığı metotta farklılık göstermektedir. Dosyalar ile iletişim, POSIX message queue ile iletişim yöntemleri processler bazında kullanılmıştır. Bunun yanında bir de threadler ve ortak bellek alanını kullanarak iletişim yöntemi kullanılmıştır.

## 2.Programın Yapısı

Dosya ile veri geçişi ve POSIX message queue ile veri geçişi yöntemlerinde fork() kullanılmıştır ve alt birim olarak child processlerden faydalanılmıştır. Diğer üçüncü yöntemde ise threadlerden ve ortak bellek alanından faydalanılmıştır.

## 2.1.Dosya ile Veri Aktarımı

Child process oluşturmak için kullanılan metod özyinelemeli olarak verilen N parametresi sıfır olana kadar devam eder. Child processler ise sürekli bir eksilen n değerini ID olarak kullanır ve kendilerine verilen dosya ismi dizisinden ID numaralarına göre bir dosya ismi seçeler. Bu sayede aralarında özel bir iletişime gerek kalmadan farklı dosyalar üzerinde işlem yaparlar. Her child process dosya okuma ve verileri sıralama işlemindensonra k. değeri alır. Bu değeri ID numarasını isim olarak kullanan bir geçici dosya oluşturarak saklar. Parent process sıra kendisine geldiğinde argüman olarak verilen n değerini kullanarak ID değerlerini bulur ve ara dosyaları okur. Değerler sıralandıktan sonra argüman olarak verilen çıkış dosyası ismi kullanılarak sonuçlar yazdırılır.

## 2.2.POSIX Message Queue ile Veri Aktarımı

2.1 kısmında bahsedilen aynı yöntem ile child processler oluşturulur. Bu metotta farklı olarak dosyalar yerine POSIX Message Queue yapısı kullanılır. Processler eşsiz bir ID'ye sahip posta kutularını kullanarak iletişim sağlarlar.

## 2.3.Thread ve Ortak Bellek Alanı ile Veri Aktarımı

Veri iletişimi için threadler ve ortak bellek alanları kullanılır. Threadlere argüman olarak bir struct yapısı adresi verilir. Threadler bu pointerden faydalanarak main thread ile aynı bellek alanına erişebilirler. Main thread, diğer threadlerin sonuçlarına ulaşmak için ekstra bir efor sarf etmez ve kendi bellek alanından erişir.

## 2.4.Programların Kullanımı

Sözü geçen üç farklı yöntemle hazırlanmış programlar aynı şekilde kullanılırlar. İlk argüman k parametresi içindir. Listelerdeki k. elemanın ana birime iletilmesini sağlarlar. İkinci argüman kaç dosya ve bu dosyalara bakacak kaç alt birimin olacağını beliren n parametresine gider. Ardınan n

tane dosya ismi girilmesi gerekir. Bunlar alt birimlerin değerleri alacağı dosya isimleridir. En sonda sonucun yazdırılması için bir dosya ismi verilir.

Kullanım patterni: ./programİsmi <K> <N> <N adet veri dosyası> <output dosyası>

Örnek kullanım : ./findtopk 3 2 deneme1.txt deneme2.txt outputFile.txt

Bu kullanımda iki tane alt birim(thread veya child process ) oluşur ve verilen dosyaları alırlar. En büyük üçüncü değeri bulup ana birime(main thread veya parent process) iletirler. Ana birim ise bu sonuçları tekrar sıralayıp outputFile.txt adında bir dosyaya yazdırır.

## 3.Performans Metrikleri

Performans metrikleri 2.4 kısmında bahsedilen parametrelerdir. Performans ölçümü için alınacak örneklem sayısı

### 3.1.Metrik Olarak N Parametresi

N parametresi kaç tane veri dosyası olacağını ve dolayısı ile kaç tane alt birimin oluşturlacağına karar veren parametredir. Diğer iki paremetre sabit kalırken N parametresi değiştirilecek ve zaman tabanlı bir performans değerlendirmesi yapılacaktır.

## 4.Performans Ölçümü

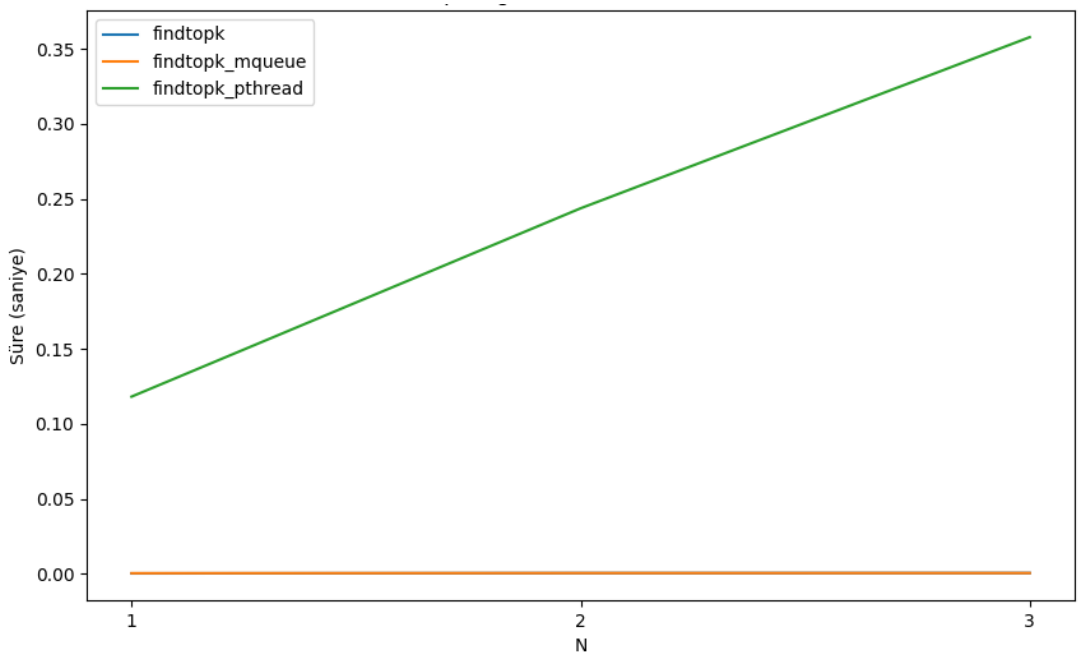
Performans ölçümü N parametresi üzerinden yapılacaktır. Diğer parametreler sabit kalacaktır. K parametresine argüman olarak 1000 değeri verilmesi ve dosyalardaki veri sayısının 10000 tane olması zamanların daha belirgin olması adına uygun görülmüştür. N parametresi ise 1, 2 ve 3 değerleri ile denenecek ve sonuçları analiz edilecektir.

### 4.1.Ölçüm Değerleri

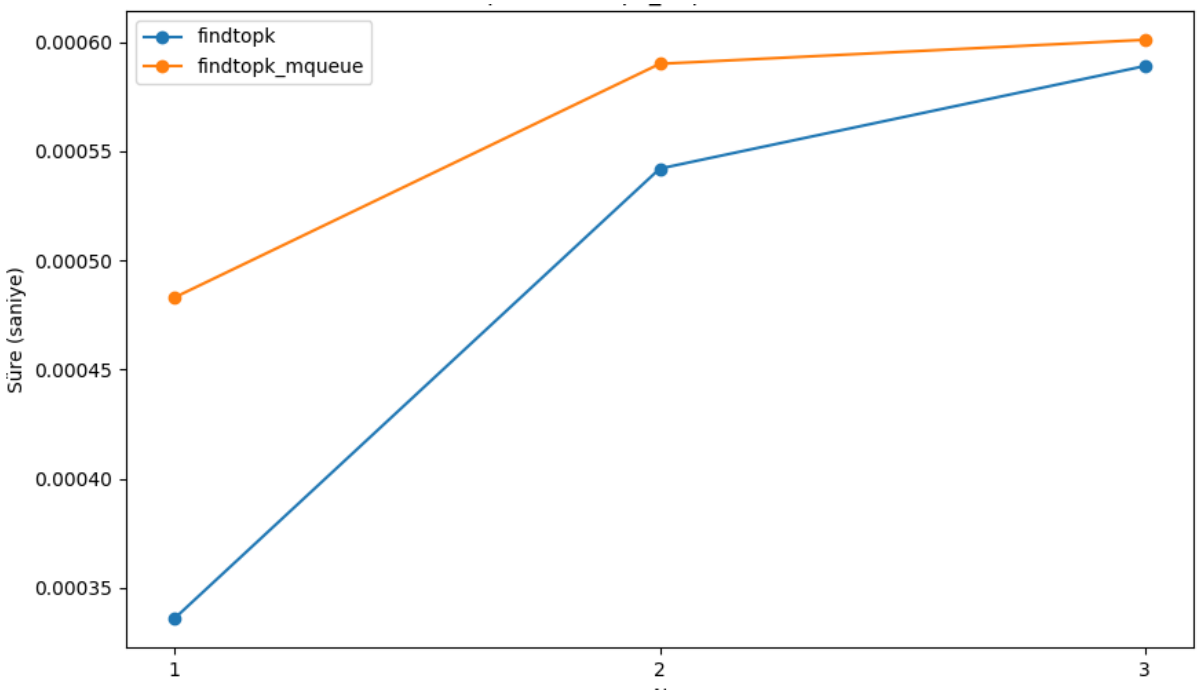
Tablo 1: Programların farklı N değerleri ile test edilmesinin ölçüm sonuçları.

| | findtopk | findtopk_mqueue | findtopk_pthread |
|---|---|---|---|
| 1 | 0.000336 | 0.000483 | 0.118135 |
| 2 | 0.000542 | 0.00059 | 0.243659 |
| 3 | 0.000589 | 0.000601 | 0.357686 |

Grafik 1: findtopk, findtopk_pthread, findtopk_mqueue programlarının ölçüm sonuçlarının çizgi grafiği.




Grafik 2 : findtopk ve findtopk_mqueue programlarının ölçüm sonuçlarının çizgi grafiği.




## 5. Sonuç

Grafiklere bakıldığında en verimliden en verimsize doğru programlar findtopk, findtopk_mqueue ve findtopk_pthread olmaktadır. findtopk_pthread diğer programlara göre büyük bir zaman farkına sahiptir. Bunun sebebi her bir processe belli bir CPU zamanı ayrılmasıdır. Thread sayısı arttıkça bu CPU zamanı threadler arasında bölüştürülür. Yapılması gerken iş artmasına rağmen birim zamanda yapılabilecek iş değişmez. Child process oluşturan programlar yeni processler oluşturduğu için ek iş gücüne sahip olur ve yapılması gereken işin artması programı daha az etkiler.

findtopk programının findtopk_mqueue programından hızlı olma sebebi ise dosya okuma yazma gibi temel sistem çağrıları kullanan bir işlemi yapmasıdır. POSIX queue, daha kompleks sistem çağrıları kullanır ve altında bir senkronizasyon mekanizması vardır.