

Bekleme Statistics

| | fcfs | pjf_preemptive | rr | pjf | sjf_preemptive | sjf |
|----------|-------|----------------|---------|--------|----------------|--------|
| ortalama | 813.5 | 833.63 | 1,115.3 | 824.77 | 537 | 537.42 |
| maksimum | 1,683 | 1,689 | 1,905 | 1,689 | 1,863 | 1,863 |

Bekleme zamanı Analizi

- **Ortalama:** En yüksek **rr** (1,115.3), en düşük **sjf_preemptive** (537).
- **Maksimum:** En yüksek **rr** (1,905), en düşük **fcfs** (1,683).

Açıklama

Bu durum, **SJF (Shortest Job First)** algoritmalarının kısa işlere öncelik vererek bekleme ve tamamlama sürelerini en verimli şekilde optimize etmesiyle açıklanabilir. **RR (Round Robin)** algoritmasında ise her işin zaman dilimlerine bölünmesi, özellikle yoğun kuyruklarda toplam bekleme süresini ve işin sistemde kalma süresini ciddi oranda artırarak en yüksek değerlerin oluşmasına neden olur.

Tamamlama Statistics ⇄

| | fcfs | pjf_preemptive | rr | pjf | sjf_preemptive | sjf |
|----------|-------|----------------|---------|--------|----------------|--------|
| ortalama | 824 | 844.13 | 1,125.8 | 835.27 | 547.5 | 547.92 |
| maksimum | 1,703 | 1,707 | 1,925 | 1,707 | 1,883 | 1,883 |

- **Ortalama:** En yüksek **rr** (tamamlama: 1,125.8), en düşük **sjf_preemptive** (547.5).
- **Maksimum:** En yüksek **rr** (tamamlama: 1,925) en düşük **fcfs** (1,703)

Açıklama

Bu durum, **sjf_preemptive** algoritmasının en kısa işleri sürekli önceliklendirerek ortalama süreleri yaklaşık **%50** oranında düşürmesiyle, **rr** algoritmasının ise sabit zaman dilimleri nedeniyle her süreci kuyrukta uzun süre bekleterek verimliliği düşürmesiyle açıklanabilir. **fcfs** algoritmasının maksimum değerlerde en düşük kalması, uzun süreçlerin bölünmeden tamamlanması sayesinde uç değerlerdeki gecikmenin sınırlı kaldığını gösterir.

Troughput Statistics

| | fcfs | pjf_preemptive | rr | pjf | sjf_preemptive | sjf |
|-------|------|----------------|----|-----|----------------|-----|
| T=50 | 9 | 6 | 6 | 8 | 11 | 11 |
| T=100 | 13 | 11 | 9 | 13 | 22 | 22 |
| T=150 | 16 | 14 | 14 | 16 | 32 | 32 |
| T=200 | 19 | 20 | 14 | 21 | 42 | 42 |

En Hızlı: SJF / SJF_P (42 iş) : En kısa işleri önce bitirerek birim zamanda en çok işi tamamlar. Matematiksel olarak en yüksek verimi sağlar.

Orta: PJF (21 iş) ve FCFS (19 iş) : Geliş sırasına veya önceliğe bakarlar. Uzun bir iş öne gelirse arkadaki kısa işleri bekleterek toplam hızı düşürür.

En Yavaş: RR (14 iş) : İşlemler arasında sürekli geçiş (context switch) yaptığı için zaman kaybeder. Verimlilikten ziyade adil paylaşımı hedefler.

CPU verimi

| | fcfs | pjf_preemptive | rr | pjf | sjf_preemptive | sjf |
|-----------------|---------|----------------|---------|---------|----------------|---------|
| num_switches | 200 | 202 | 1,100 | 199 | 213 | 199 |
| cpu_utilization | 99.9049 | 99.9039 | 99.4318 | 99.9053 | 99.8987 | 99.9053 |

En Yüksek Verim: PJF ve SJF algoritmaları, en düşük anahtar değişimi (**199**) ve yüksek CPU kullanımı (**99.9053**) ile en iyi performansı göstermiştir.

En Düşük Verim: Round Robin (RR), çok yüksek anahtar değişimi (**1,100**) nedeniyle CPU kullanımında en düşük seviyede (**99.4318**) kalmıştır.

Genel Gözlem: Bir önceki tabloya göre iş yükü artmış (anahtar değişimleri yaklaşık iki katına çıkmış), ancak verimlilik sıralaması değişmemiştir; **FCFS (99.9049)** hala istikrarlı bir performans sergilemektedir.