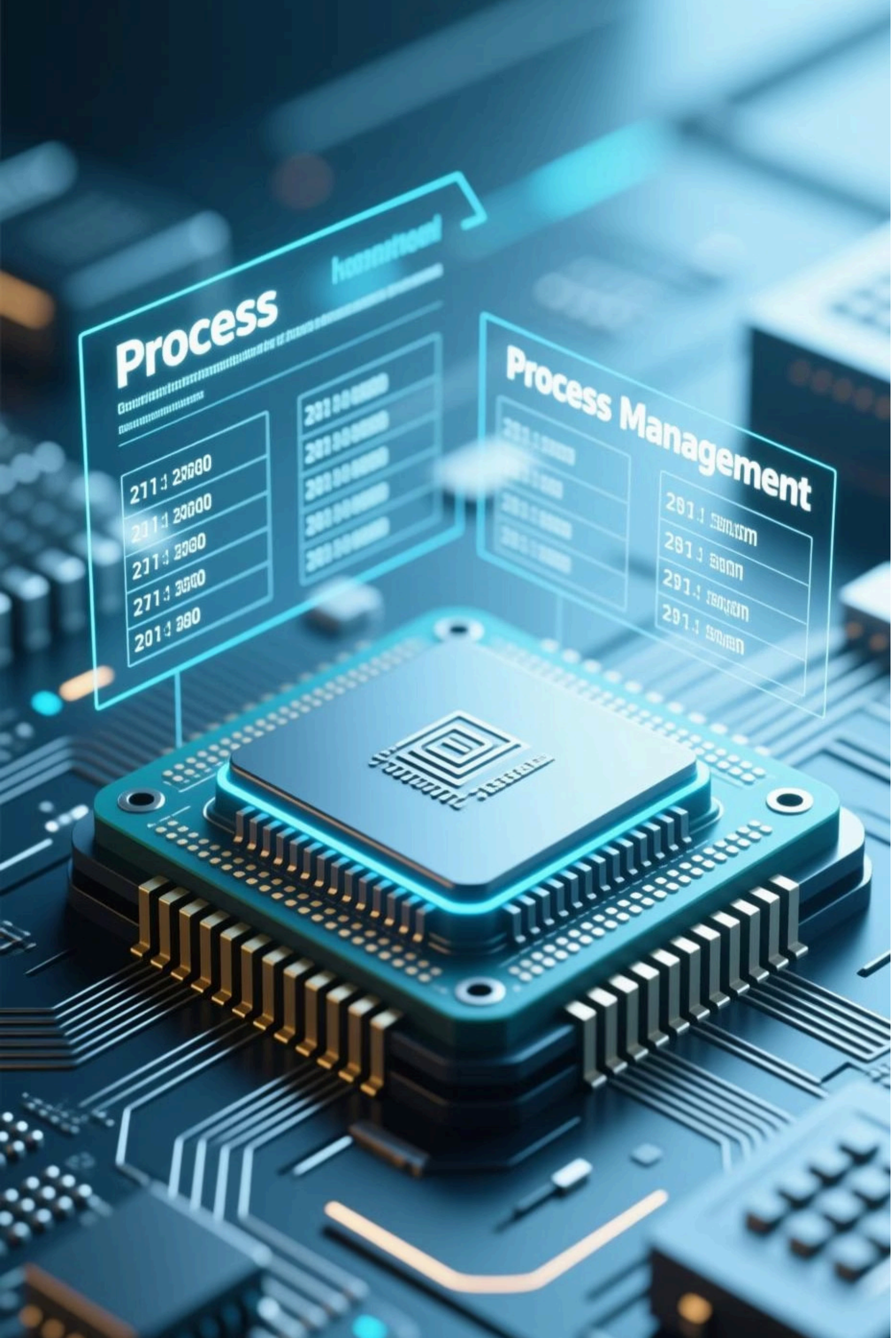


Scheduling Algorithms & Turnaround Time: Finding the Best Performer

This presentation explores how different CPU scheduling algorithms impact system performance, with a particular focus on turnaround time as our key metric for comparison.

Why Scheduling Algorithms Matter

The CPU is a system's most critical resource, and how it allocates time between competing processes directly impacts both system efficiency and user experience.

Turnaround time (TAT) serves as our primary metric for evaluating scheduler performance, measuring the total elapsed time from when a process arrives to when it completes.

Lower turnaround times mean faster process completion and better system responsiveness, making TAT a crucial factor in scheduler selection.

Key Concepts to Understand

Arrival Time

When a process enters the ready queue, waiting for CPU allocation

Burst Time

Total CPU time required by the process to complete execution

Completion Time

The moment when a process finishes its execution

Calculating Performance Metrics

Turnaround Time = Completion Time – Arrival Time

Measures total time in system (including waiting)

Waiting Time = Turnaround Time – Burst Time

Measures time spent waiting for CPU

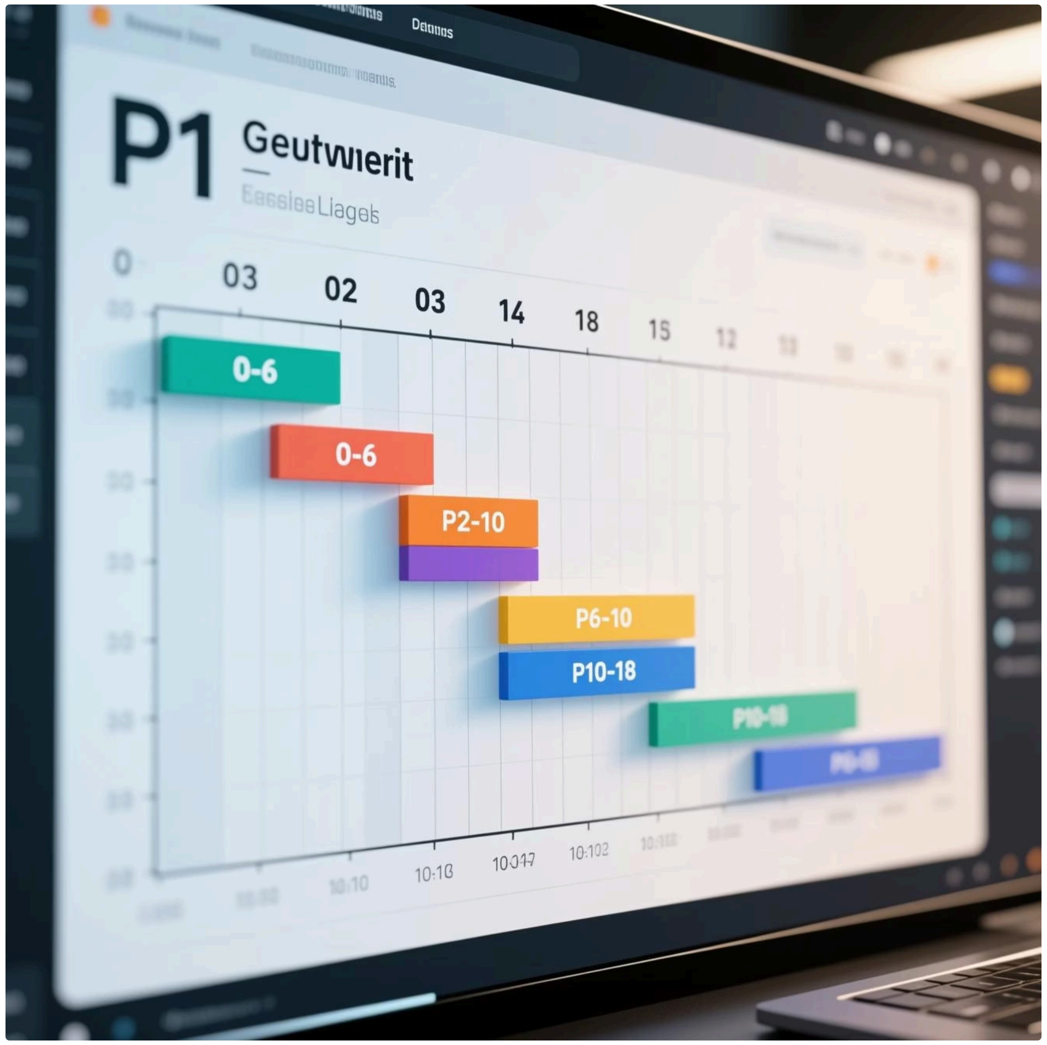
First Come First Serve (FCFS) Example

Process Details

Process	Arrival Time	Burst Time
P1	0	6
P2	2	4
P3	4	8

FCFS executes processes in order of arrival without interruption.

Gantt Chart



Turnaround Time Calculation

- P1: 6 - 0 = **6**
- P2: 10 - 2 = **8**
- P3: 18 - 4 = **14**

Average TAT = (6 + 8 + 14) / 3 = 9.3

Shortest Job First (SJF) Example

SJF prioritizes the process with the shortest burst time available at the moment of scheduling.

Non-preemptive Execution

Using the same processes, SJF schedules based on shortest burst time:

- At t=0, only P1 has arrived (burst=6) → Execute P1
- At t=6, P2 (burst=4) and P3 (burst=8) are waiting → P2 has shorter burst → Execute P2
- At t=10, only P3 remains → Execute P3

Gantt Chart

P1 (0-6) → P2 (6-10) → P3 (10-18)

In this particular example, the execution order matches FCFS due to arrival times.

Turnaround Times

- P1: 6 - 0 = **6**
- P2: 10 - 2 = **8**
- P3: 18 - 4 = **14**

Average TAT = 9.3

Same as FCFS in this specific example, but SJF generally performs better with varied burst times.

Round Robin (RR) Example

Using Time Quantum = 4

Round Robin allocates the CPU to each process for a fixed time slice (quantum), cycling through all processes until completion.

Execution Sequence:

1. P1 runs for 4 units (0-4), 2 units remain
2. P2 runs for 4 units (4-8), completes
3. P1 returns to run for 2 units (8-10), completes
4. P3 runs for 4 units (10-14), 4 units remain
5. P3 runs for final 4 units (14-18), completes

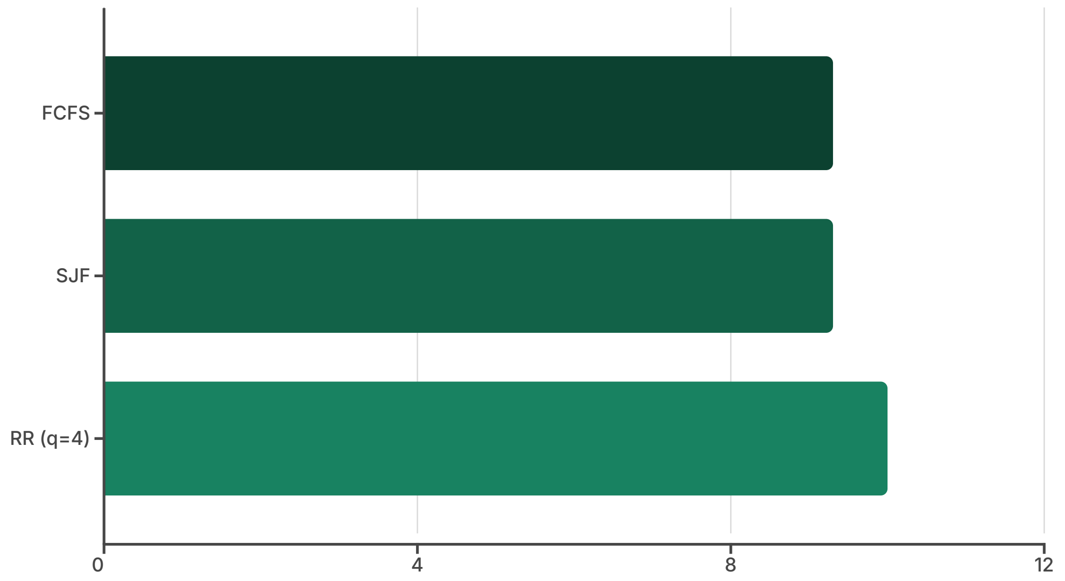
Turnaround Times:

- P1: 10 - 0 = **10**
- P2: 8 - 2 = **6**
- P3: 18 - 4 = **14**

Average TAT = (10 + 6 + 14) / 3 = 10

RR increases responsiveness but can increase average turnaround time compared to other algorithms.

Comparing Average Turnaround Times



FCFS

Simple implementation but can cause long waits if lengthy processes arrive first

SJF

Theoretically optimal for average TAT if job lengths are known in advance

Round Robin

Fair, better response time, but typically higher average TAT

In our example scenario, FCFS and SJF tie due to the specific arrival pattern, but SJF generally outperforms FCFS with varied burst times.

Why Turnaround Time Matters for Scheduler Choice

System Performance

Minimising TAT improves throughput and overall system efficiency

User Satisfaction

Lower TAT leads to better user experience and perceived responsiveness

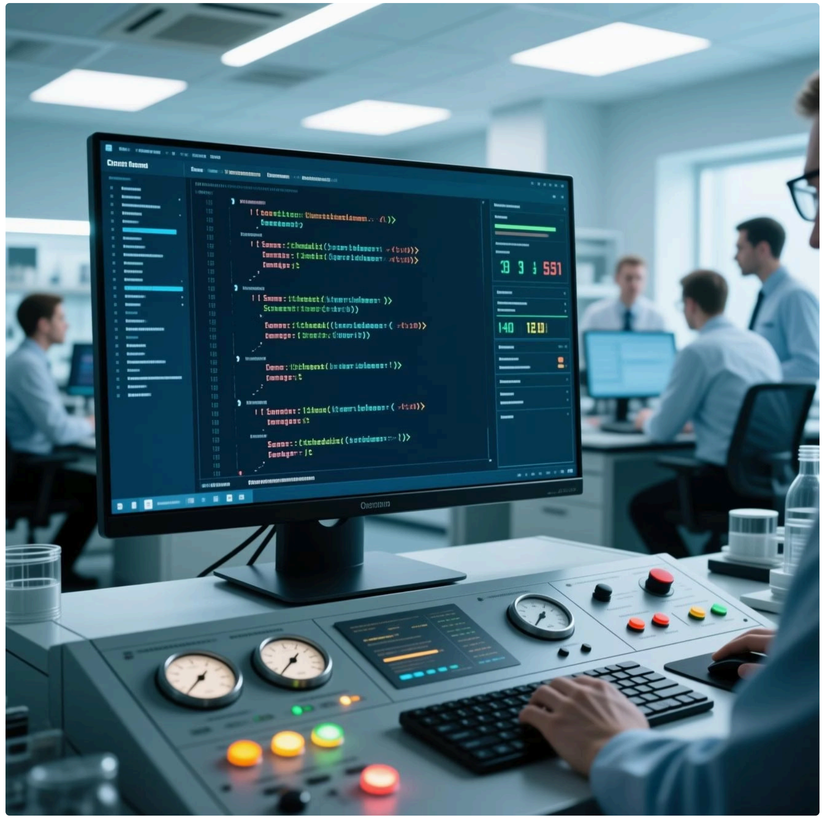
Resource Utilisation

Efficient scheduling balances CPU usage and reduces idle time

The **ideal scheduler** depends on your system's specific needs and workload characteristics:

- SJF provides optimal TAT but requires foreknowledge of burst times
- RR delivers fairness but may increase average TAT
- FCFS is simple but can lead to the "convoy effect" where short processes wait behind long ones

Real-World Considerations



Implementation Challenges

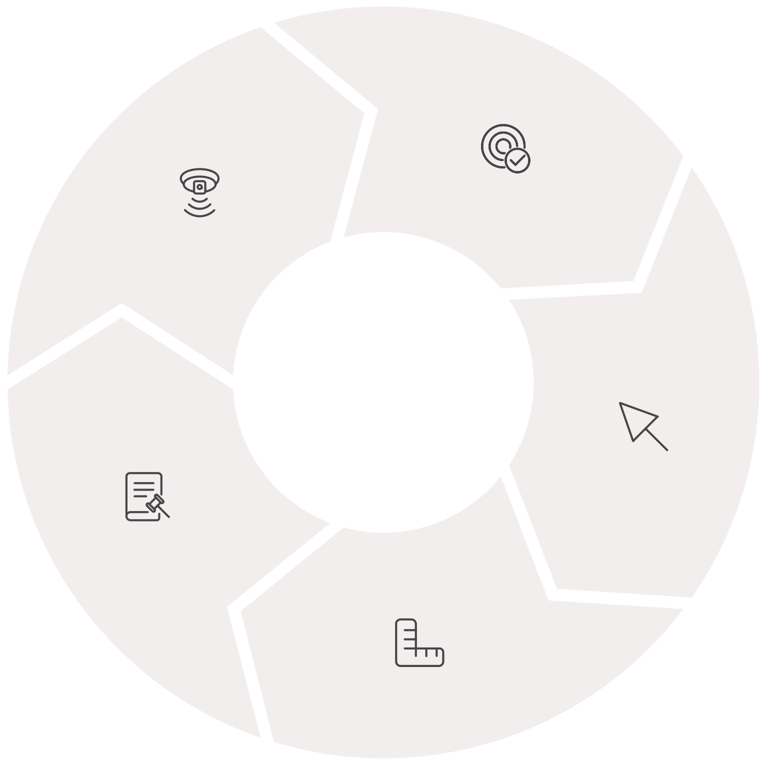
- **Burst time prediction** is difficult, making pure SJF hard to implement
- Most modern systems use **multilevel feedback queues** that combine aspects of multiple algorithms
- Context switching overhead becomes significant with small time quantums

Additional Performance Metrics

- **Waiting time:** Time spent in ready queue
- **Response time:** Time until first CPU burst
- **Throughput:** Processes completed per unit time
- **CPU utilisation:** Percentage of time CPU is busy

Real-world schedulers must balance turnaround time with fairness, predictability, and overhead considerations.

Conclusion: Choosing the Better Scheduler



Define Objectives

Prioritize metrics based on system goals (TAT, fairness, responsiveness)

Select Algorithm

Choose the scheduler that best matches your requirements

Measure Performance

Gather real-world data on turnaround time and other metrics

Tune Parameters

Adjust time quantum, priorities, or other settings to optimize

Analyze Workload

Understand process characteristics (CPU vs I/O bound, typical burst times)

Key Takeaways:

- For lowest turnaround time with known burst times, SJF is optimal
- For interactive systems needing fairness, Round Robin is preferred
- Modern systems typically use hybrid approaches to balance competing goals
- Understanding these concepts is crucial for system designers and administrators