SNS COLLEGE OF TECHNOLOGY

Coimbatore-35.
An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A+' Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

COURSE NAME : 19CSB201 – OPERATING SYSTEMS

II YEAR/ IV SEMESTER

UNIT – II Process Scheduling And Synchronization

Topic: CPU Scheduling : Algorithm Evaluation

Mr.N.Selvakumar
Assistant Professor
Department of Computer Science and Engineering

Algorithm Evaluation

The first problem is defining the criteria to be used in **selecting an algorithm**.

Criteria are often defined in terms of **CPU utilization, response time, or throughput**.

To select an algorithm, we must first define the relative importance of these elements.

Our criteria may include several measures,

such as these:

- **Maximizing CPU utilization** under the constraint that the maximum response time is 1 second
- **Maximizing throughput** such that turnaround time is (on average) linearly proportional to total execution time

Once the selection criteria have been defined, we want to evaluate the algorithms under consideration.

Analytic evaluation

One major class of evaluation methods is analytic evaluation.

Analytic evaluation uses the given algorithm and the system workload to **produce a formula** or number **to evaluate the performance** of the algorithm for that workload.

1.Deterministic Modeling

- Deterministic modeling is one type of analytic evaluation.
- This method takes a particular predetermined workload and defines the performance of each algorithm for that workload.

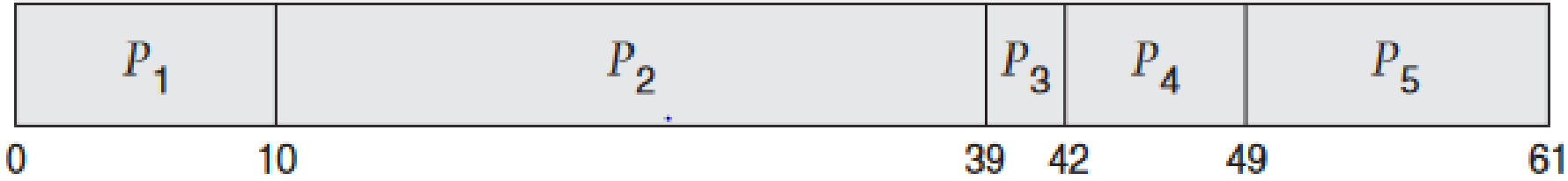
- Deterministic modeling is **simple and fast**.
- It gives us exact numbers, allowing us to compare the algorithms.
- However, it requires exact numbers for input, and its answers apply only to those cases.

For example, assume that we have the workload shown below. All five processes arrive at time 0, in the order given, with the length of the CPU burst given in milliseconds:

Process	Burst Time
P1	10
P2	29
P3	3
P4	7
P5	12

Consider the FCFS, SJF, and RR (quantum = 10 milliseconds) scheduling algorithms for this set of processes. Which algorithm would give the minimum average waiting time?

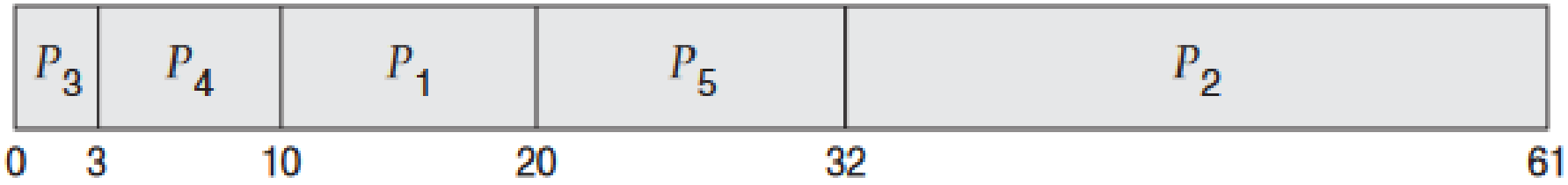
For the FCFS algorithm, we would execute the processes as



The waiting time is 0 milliseconds for process P1 , 10 milliseconds for process P2 , 39 milliseconds for process P3 , 42 milliseconds for process P4 , and 49 milliseconds for process P5 .

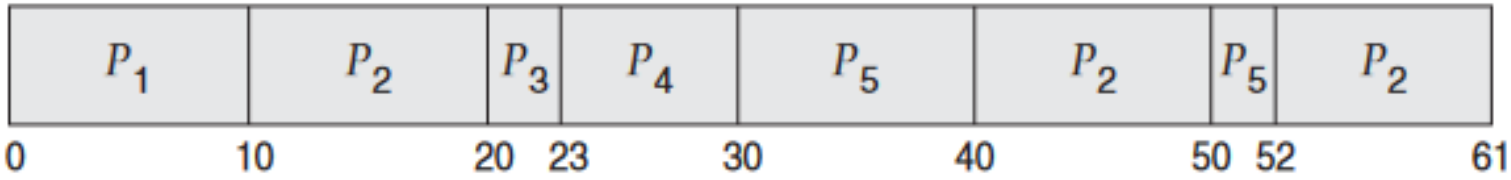
Thus, the average waiting time is (0 + 10 + 39 + 42 + 49)/5 = 28 milliseconds

With nonpreemptive SJF scheduling, we execute the processes as



The waiting time is 10 milliseconds for process P1 , 32 milliseconds for process P2 , 0 milliseconds for process P3 , 3 milliseconds for process P4 , and 20 milliseconds for process P5 . Thus, the average waiting time is (10 + 32 + 0 + 3 + 20)/5 = 13 milliseconds.

With the RR algorithm, we execute the processes as



- The waiting time is 0 milliseconds for process P1 , 32 milliseconds for process P2 , 20 milliseconds for process P3 , 23 milliseconds for process P4 , and 40 milliseconds for process P5 . Thus, the average waiting time is (0 + 32 + 20 + 23 + 40)/5 = 23 milliseconds.

We can see that, in this case, the average waiting time obtained with the SJF policy is less than half that obtained with FCFS scheduling; the RR algorithm gives us an intermediate value.

SJF policy will always result in the minimum waiting time

2.Queueing Models

- The distribution of CPU and I/O bursts.
- These distributions can be measured and then approximated or simply estimated.
- The result is a mathematical formula describing the probability of a particular CPU burst. Commonly, this distribution is exponential and is described by its mean.
- Possible to compute the average throughput, utilization, waiting time, and so on for most algorithms.

The computer system is described as a network of servers. Each server has a queue of waiting processes. The CPU is a server with its ready queue, as is the I/O system with its device queues. Knowing arrival rates and service rates, we can compute utilization, average queue length, average wait time, and so on. This area of study is called **queueing-network analysis**.

As an example, let n be the **average queue length** (excluding the process being serviced), let W be the **average waiting time** in the queue, and let λ be the **average arrival rate for new processes** in the queue (such as three processes per second). We expect that during the time W that a process waits, $\lambda \times W$. new processes will arrive in the queue. If the system is in a steady state, then the number of processes leaving the queue must be equal to the number of processes that arrive. Thus,

$$n = \lambda \times W.$$

This equation, known as **Little's formula**, is particularly useful because it is valid for any scheduling algorithm and arrival distribution.

3.Simulations

To get a more accurate evaluation of scheduling algorithms, we can use simulations. Running simulations involves programming a model of the computer system. Software data structures represent the major components of the system. The simulator has a variable representing a clock. As this variable's value is increased, the simulator modifies the system state to reflect the activities of the devices, the processes, and the scheduler. As the simulation executes, statistics that indicate algorithm performance are gathered and printed.

The data to drive the simulation can be generated in several ways. The most common method uses a random-number generator that is programmed to generate processes, CPU burst times, arrivals, departures, and so on, according to probability distributions.

A **distribution-driven simulation may be inaccurate**, however, because of relationships between successive events in the real system. The frequency distribution indicates only how many instances of each event occur; it does not indicate anything about the order of their occurrence.

To correct this problem, we can use **trace tapes**. We create a trace tape by **monitoring the real system and recording the sequence of actual events**. We then use this sequence to drive the simulation. Trace tapes provide an excellent way to compare two algorithms on exactly the same set of real inputs. This method can produce accurate results for its inputs.

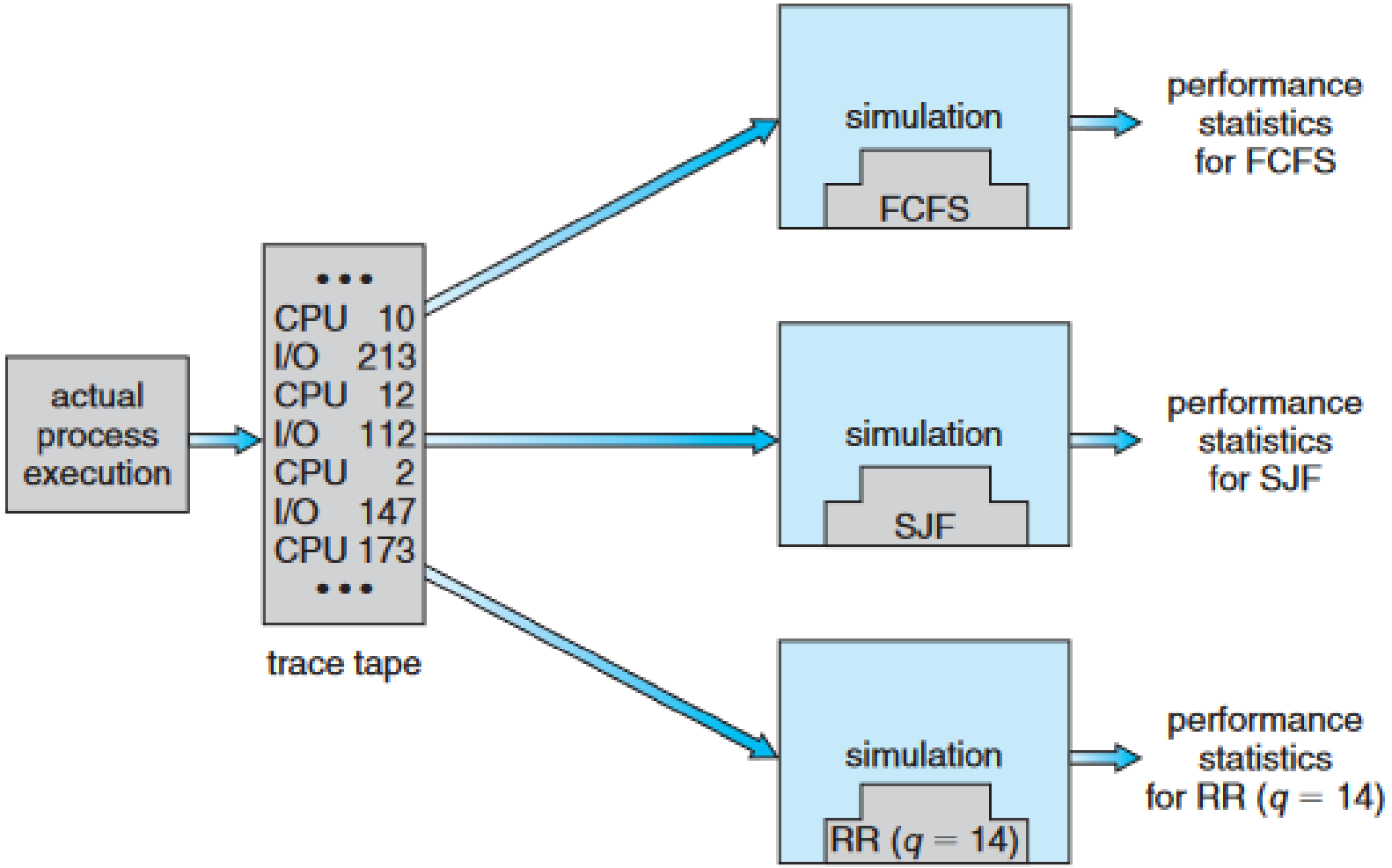


Figure 6.25 Evaluation of CPU schedulers by simulation.

Simulations can be expensive, often **requiring hours of computer time.**

A more detailed simulation provides **more accurate results**, but it also takes more computer time.

In addition, trace tapes can require **large amounts of storage** space.

Finally, the **design, coding, and debugging** of the simulator can be a **major task**.

4.Implementation

Even a **simulation is of limited accuracy**. The only completely accurate way to evaluate a scheduling algorithm is to **code it up, put it in the operating system, and see how it works**. This approach puts the actual algorithm in the real system for evaluation under real operating conditions.

The major difficulty with this approach is the **high cost**. The expense is incurred not only in coding the algorithm and modifying the operating system to support it (along with its required data structures) but also in the reaction of the users to a constantly changing operating system.

REFERENCES

TEXT BOOKS:

T1 Silberschatz, Galvin, and Gagne, “Operating System Concepts”, Ninth Edition, Wiley India Pvt Ltd, 2009.)

T2. Andrew S. Tanenbaum, “Modern Operating Systems”, Fourth Edition, Pearson Education, 2010

REFERENCES:

R1 Gary Nutt, “Operating Systems”, Third Edition, Pearson Education, 2004.

R2 Harvey M. Deitel, “Operating Systems”, Third Edition, Pearson Education, 2004.

R3 Abraham Silberschatz, Peter Baer Galvin and Greg Gagne, “Operating System Concepts”, 9th Edition, John Wiley and Sons Inc., 2012.

R4. William Stallings, “Operating Systems – Internals and Design Principles”, 7th Edition, Prentice Hall, 2011


THANK
YOU!