MULTICORE CPU PROCESS SCHEDULING SIMULATOR

PROBLEM STATEMENT

Modern computers are all equipped with multicore processors, but it is hard for students to understand how process execution could be scheduled over multiple cores. Current OS schedulers are typically background tasks that can't be seen. This project implements a simulator of how the processes are scheduled on multi core system using different CPU scheduling algorithms, as well as analyse the performance measures such as waiting time; turn-around time and CPU utilization. Execution results are stored in CSV files and Machine Learning analysis is used to study and compare scheduling performance.

ABSTRACT

CPU scheduling plays a critical role in the efficiency and responsiveness of contemporary computing systems such as desktop computers and servers with multicore processors. Although numerous studies in IEEE literature have addressed CPU scheduling simulators, analytical models, and algorithm-specific comparisons such as Round Robin, Priority, and Shortest Job First many existing solutions lack real-time execution visualization and integrated frameworks for comparative performance analysis. Key performance indicators (KPIs) like latency, response time, and CPU load are often only partially addressed or are evaluated through post-processing.

To overcome these limitations, this paper proposes the Multicore CPU Process Scheduling Simulator, which replicates a real-time multicore scheduling environment and provides a unified platform for evaluating common scheduling algorithms such as FCFS, SJF, Priority Scheduling, and Round Robin. The simulator integrates real-time Gantt chart graphing, response time tracking, and efficiency-driven performance metrics within an interactive GUI. It simulates the behavior of processes using parameters like arrival time, burst time, and priority to realistically reflect OS-level scheduling operations. The framework supports parallelism, preemption, and core-level workload balancing, which are typically absent in other tools.

To enhance analysis and insight, the simulator stores execution results in CSV files and incorporates Machine Learning (ML) techniques to identify scheduling trends, optimize algorithm selection, and predict high-efficiency scheduling outcomes. It objectively computes Waiting Time, Turnaround Time, Response

Time, Throughput, and CPU Utilization across multicore configurations, enabling transparent and consistent performance comparisons.

The originality of this work lies in its real-time, multicore-aware scheduling visualization, ML-enhanced performance analysis, and full-cycle practical implementation that bridges theoretical scheduling models with real OS behaviors. This simulator serves as a robust educational and research tool for understanding and evaluating CPU scheduling in modern computing systems.

LITERATURE SURVEY

REF [1] TITLE: Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling

- **AUTHORS:** A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon, Q. Mao
- **CONFERENCE:** IEEE SoutheastCon
- **YEAR:** 2019
- **SUMMARY / ALGORITHM:**
    - This paper evaluates different variations of the Round Robin scheduling algorithm. The study mainly focuses on improving time quantum selection to reduce waiting time and turnaround time.
    - Experimental results show that dynamic time quantum performs better than traditional Round Robin. However, the work is limited to algorithm performance analysis and does not provide visualization or multicore execution.

REF [2] TITLE: Simulation of CPU Scheduling Algorithms for Efficient Execution of Processes

- **AUTHORS:** K. Vayadande, P. Sheth, D. Pawal, A. Pathak, K. Paralkar, S. Patil
- **CONFERENCE:** International Conference for Advancement in Technology (ICONAT)
- **YEAR:** 2023
- **SUMMARY / ALGORITHM:**
    - This paper presents a simulator to study CPU scheduling algorithms such as FCFS, SJF, Priority, and Round Robin.

    - The simulator calculates performance metrics including waiting time and turnaround time for processes.
    - The study helps in comparing algorithms but mainly considers a single-processor environment and lacks multicore visualization.

REF [3] TITLE: Comparative Analysis of Scheduling Algorithms for Grid Computing

- **AUTHORS:** S. Sharma, A. Chhabra, S. Sharma
- **CONFERENCE:** International Conference on Advances in Computing, Communications and Informatics (ICACCI)
- **YEAR:** 2015
- **SUMMARY / ALGORITHM:**
    - This paper compares CPU scheduling algorithms (FCFS, SJF, Priority, Round Robin) in grid computing using metrics like waiting time, turnaround time, and resource utilization. However, it only provides theoretical simulation and does not store results as datasets or use machine learning.
    - Our project stores results in a CSV dataset and applies machine learning analysis to evaluate and predict scheduling performance.

REF [4] TITLE: E2C : A Visual Simulator to Reinforce Education of Heterogeneous Computing Systems

- **AUTHORS:** Research group on computer architecture education
- **CONFERENCE:** Computer Architecture and Education Research
- **YEAR:** 2021
- **ALGORITHM / SUMMARY:**
    - This paper introduces an educational simulator (E2C) designed to teach scheduling concepts in heterogeneous and parallel systems. It demonstrates how different processors affect execution performance and helps learners understand parallel execution.
    - The simulator is complex for beginners and not focused on simple operating system scheduling. Multicore Gantt chart visualization and easy algorithm comparison are not clearly provided.

REF [5] TITLE: Visualization Tools for Operating System Learning

- **AUTHORS:** OS education researchers
- **JOURNAL:** Interactive Learning Technologies
- **YEAR:** 2022
- **ALGORITHM / SUMMARY:**
    - The study highlights the importance of visualization tools in teaching operating system concepts. It shows that graphical demonstrations improve student understanding of process execution and scheduling behavior.
    - Existing tools lack multicore execution visualization and do not allow easy comparison of multiple scheduling algorithms with performance metrics.

REF [6] TITLE: CPU Scheduling Algorithms: Case & Comparative Study

- **AUTHORS:** S. Zouaoui, L. Boussaid, A. Mtibaa
- **CONFERENCE:** International Conference on Automatic Control and Computer Engineering (STA)
- **YEAR:** 2016
- **ALGORITHM / SUMMARY:**
    - This study compares major CPU scheduling algorithms such as FCFS, SJF, Priority, and Round Robin. The performance is analyzed using metrics like waiting time and turnaround time.
    - The work provides theoretical comparison but lacks implementation-based teaching support and real-time graphical execution.

REF [7] TITLE: Comparative Analysis of CPU Scheduling Algorithms &Their Optimal Solutions

- **AUTHORS:** M. Rajasekhar Reddy, V. V. D. S. S. Ganesh
- **YEAR:** 2019 (IEEE ICCCMC)
- **ALGORITHM / SUMMARY :**

- The paper presents a comparative study of various CPU scheduling algorithms such as FCFS, SJF, SRTF, Round Robin and Feedback scheduling.
- It explains scheduling concepts including preemptive and non-preemptive scheduling and evaluates performance using parameters like waiting time, turnaround time and response time. The study also proposes modified versions of some algorithms to improve performance.
- The work is mainly theoretical and analytical. It does not provide an interactive simulation tool, real-time execution visualization or multicore CPU scheduling support.

EXISTING SYSTEM

- Several CPU scheduling simulators are already available.
- They implement traditional algorithms such as FCFS, SJF, Round Robin and Priority.
- The simulators evaluate performance metrics like waiting time and turnaround time.
- Execution is modeled using a single processor environment.
- Mainly designed for theoretical understanding of operating system concepts.

Limitations of Existing System

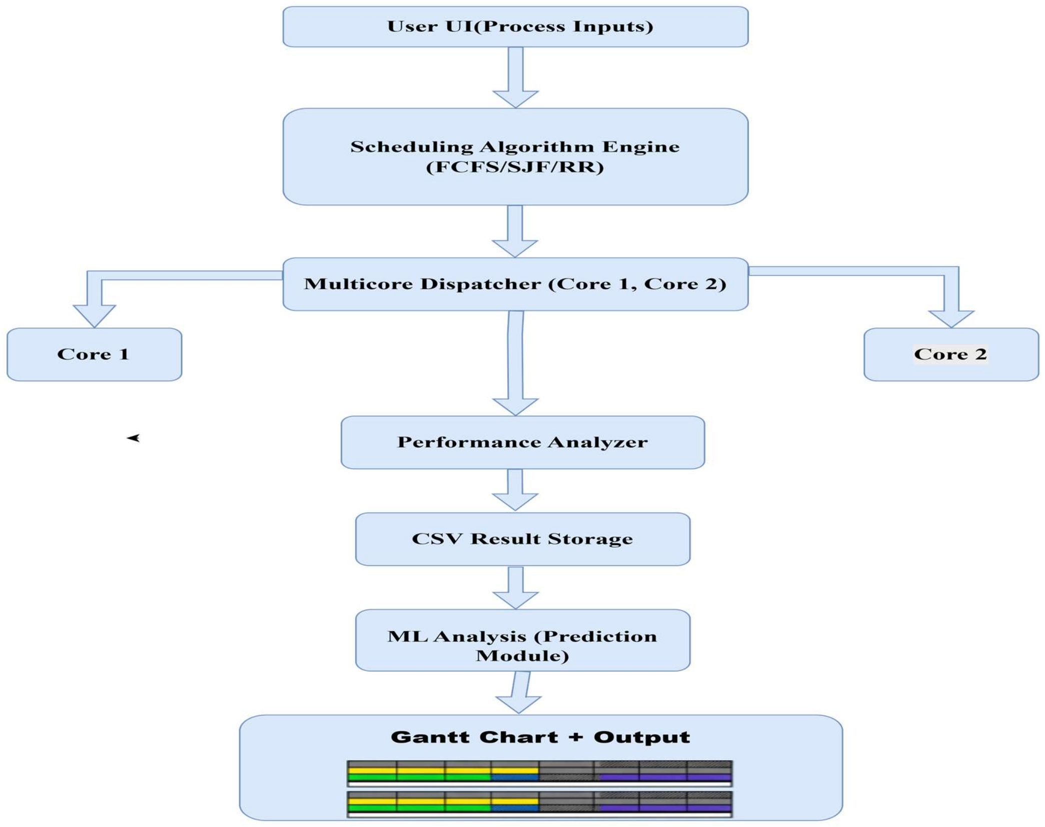
- Designed for single-core CPU architecture only.
- Cannot simulate parallel execution of processes.
- No visualization of process allocation across different CPU cores.
- Unable to analyze core-wise CPU utilization.
- Limited support for comparative analysis of algorithms in modern systems.
- Not suitable for understanding real multicore processor behavior.
- Students cannot clearly observe how scheduling works in practical systems.
- Execution results are not stored as datasets (CSV).

- No Machine Learning analysis or prediction of performance.

PROPOSED SYSTEM

- User enters process details.
- Selects number of CPU cores (2 or 4).
- Chooses scheduling algorithm.
- System allocates processes to selected cores.
- Processes are executed in the simulator.
- Gantt chart is displayed after execution.
- Performance metrics are calculated.
- Metrics include waiting time, turnaround time, and CPU utilization.
- The system stores scheduling results in CSV files and performs Machine Learning analysis to evaluate and predict scheduling performance.

HIGH-LEVEL ARCHITECTURAL DIAGRAM


User UI(Process Inputs)
Scheduling Algorithm Engine (FCFS/SJF/RR)
Multicore Dispatcher (Core 1, Core 2)
Core 1
Core 2
Performance Analyzer
CSV Result Storage
ML Analysis (Prediction Module)
Gantt Chart + Output

Methodology

- User enters process details.
- Selects scheduling algorithm.
- Processes are allocated to available CPU cores.
- Simulation is executed.
- Gantt chart is generated to visualize execution.
- Performance metrics are calculated.
- Metrics include waiting time, turnaround time, and CPU utilization.
- Execution results are stored in a CSV dataset file.
- The ML analysis module applies Machine Learning to analyze and predict performance.

MODULES

Module 1: Process Creation Module

- Accepts user input for process details:
    - Process ID
    - Arrival Time
    - Burst Time
    - Priority
- Validates inputs for correctness and completeness.
- Stores process data in a temporary in-memory structure for scheduling.

Module 2: Multicore Scheduler

- Supports simulation of 2 or 4 CPU cores.
- Assigns processes to available cores based on scheduling algorithm.
- Ensures parallel execution simulation for multicore processing.

Module 3: Scheduling Algorithms Engine

- Implements multiple CPU scheduling strategies:
    - First-Come-First-Served (FCFS)

    - Shortest Job First (SJF) – Non-pre-emptive
    - Round Robin (RR) – with configurable time quantum
    - Priority Scheduling – supports pre-assigned priority values
- Determines execution order of processes per core.

Module 4: Performance Calculator

- Computes key performance metrics:
    - Average Waiting Time
    - Average Turnaround Time
    - Throughput (number of processes completed per unit time)
    - CPU Utilization for each core

Module 5: Visualization

- Generates Gantt Chart for each core showing process execution timeline.
- Highlights start and end times for each process per core.
- Provides visual comparison of scheduling efficiency across cores.

Module 6: CSV Storage

- Saves simulation results in CSV format.
- Stores process execution details and performance metrics.
- Facilitates later analysis or sharing of results.

Module 7: ML Analysis

- Uses machine learning techniques to analyse scheduling performance.
- Predicts expected metrics (e.g., waiting time, turnaround time) based on historical data.
- Provides suggestions for optimal scheduling parameters.

Expected Output

- Provides multicore execution visualization.
- Displays Gantt chart for each CPU core.
- Calculates waiting time.

- Calculates turnaround time.
- Calculates CPU utilization.
- Enables comparison of different scheduling algorithms.
- Helps analyze algorithm performance effectively.
- Automatic saving of execution results in CSV dataset files.
- Machine Learning analysis showing performance evaluation and prediction of scheduling behavior.

TOOLS & PLATFORM

Category	**Tool**	**Purpose in Project**
Programming Language	Python	Core logic implementation
GUI	Tkinter	User interface for input & visualization
Scheduling Logic	Python Data Structures	Queue, Lists for process scheduling
Data Storage	CSV	Store results dataset
Data Processing	Pandas	Handle scheduling data
Machine Learning	Scikit-learn	Performance analysis & prediction
Visualization	Matplotlib	Gantt chart display for CPU cores
Development Environment	VS Code	Coding and debugging
Platform	Windows	Running the simulator

12-WEEK PROJECT PLAN

Week 1–2: Topic selection, guide discussion, and research paper collection.
Week 3–4: Literature survey, research gap and problem statement identification.
Week 5–6: System design, architecture diagram, and tool selection (Python, Tkinter, Matplotlib).

Week 7–8: GUI development and implementation of FCFS, SJF, Priority & Round Robin algorithms.
Week 9: Multicore process allocation and performance metrics calculation (WT, TAT, CPU Utilization).

Week 10:CSV result storage and ML analysis module implementation.
Week 11: Gantt chart generation and execution visualization**.**
Week 12: Testing, debugging, documentation, and final presentation preparation.

REFERENCES

1. A. A. Alsulami, Q. A. Al-Haija, M. I. Thanoon and Q. Mao, "Performance Evaluation of Dynamic Round Robin Algorithms for CPU Scheduling," 2019 SoutheastCon, Huntsville, AL, USA, 2019, pp. 1-5.
2. K. Vayadande, P. Sheth, D. Pawal, A. Pathak, K. Paralkar and S. Patil, "Simulation of CPU Scheduling Algorithms for Efficient Execution of Processes," 2023 International Conference for Advancement in Technology (ICONAT), Goa, India, 2023, pp. 1-6.
3. S. Sharma, A. Chhabra and S. Sharma, "Comparative analysis of scheduling algorithms for grid computing," 2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Kochi, India, 2015, pp. 349-354.
4. S. Zouaoui, L. Boussaid and A. Mtibaa, "CPU scheduling algorithms: Case & comparative study," 2016 17th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA), Sousse, Tunisia, 2016, pp. 158-164.
5. Educational Computer Architecture Research Group, "E2C: A Simulator for Heterogeneous and Parallel Systems, Computer Architecture and Education Research", 2021.
6. Operating Systems Education Researchers, "Visualization Tools for Operating System Learning", Interactive Learning Technologies Journal, 2022.
7. M. Rajasekhar Reddy, V. V. D. S. S. Ganesh, S. Lakshmi and Y. Sireesha, "Comparative Analysis of CPU Scheduling Algorithms and Their Optimal Solutions," IEEE International Conference on Communication, Computing and Machine Learning (ICCCML/ICCCMC), 2019.