# Project Report: Load Balancing in Multiprocessor Scheduling

**Team Members :**

**22BAI1466 - Vishwa**

**22BRS1362 - Hari Shankar S**

**22BRS1112 – Jesseman Devamirtham N**

**22BRS1100 - Adithya P.R.**

**Problem Statement:**

In the realm of multiprocessor scheduling, the challenge lies in achieving optimal resource utilization and minimizing processing time. The imbalance in task distribution among processors can lead to inefficient resource utilization, increased waiting times, and suboptimal system throughput. The need for effective load balancing algorithms is evident to address these challenges. This project aims to design, implement, and analyse two distinct load balancing algorithms in a multiprocessor environment, evaluating their performance based on waiting times, task execution times, and overall system throughput. The problem at hand is to explore and recommend efficient load balancing strategies for dynamic task environments, considering the nuances of both round-robin and priority-based task assignment approaches in multiprocessor scheduling.

**Pre-requisite knowledge:**

**Master slave processing:**

Master-slave processing in the context of load balancing for multiprocessor scheduling involves a hierarchical architecture where a central "master" node takes on the role of orchestrating task distribution among multiple "slave" nodes. The master node assumes the responsibility of load monitoring, dynamic task assignment, and real-time feedback to ensure an even workload distribution, minimize waiting times, and optimize system throughput. This approach provides centralized decision-making, facilitating efficient load balancing across the multiprocessor system, and it can adapt to changing workloads and system configurations. By introducing a master-slave architecture, the project can achieve a synchronized and effective load balancing strategy, contributing to enhanced performance and resource utilization in multiprocessor environments.

**Scope of the Project:**

Multiprocessor Scheduling Environment:

The project focuses on the context of a multiprocessor scheduling environment, where computational tasks need to be efficiently distributed among multiple processors.

Two Load Balancing Algorithms:

The project involves the implementation and analysis of two specific load balancing algorithms:

Algorithm 1: Load Balancing in Processor Queues (Round-Robin Approach)

Algorithm 2: Task Execution in Multi-Core Environment (Priority-Based Task Assignment)

Performance Metrics:

The evaluation of each algorithm's effectiveness includes key performance metrics such as waiting times, task execution times, and overall system throughput.

Real-time Updates and Visualization:

Real-time updates and visualizations are integrated into the simulation to provide dynamic insights into load balancing dynamics. This includes the visualization of processor queues and the on-going task transfers.

Simulation Environment:

The project creates a simulation environment with two processors, each having its own task queue. The simulation allows for a detailed analysis of the load balancing algorithms in action.

Comparison and Recommendations:

A thorough comparison is conducted to assess the strengths and weaknesses of the two algorithms. Recommendations are then provided based on the observed performance, guiding the selection of load balancing strategies in different scenarios.

Task Prioritization (Algorithm 2):

Algorithm 2 introduces task prioritization, reflecting real-world scenarios where tasks may have varying degrees of importance. The effectiveness of the priority-based task assignment is a crucial aspect of the project.

Applicability in Dynamic Task Environments:

The project considers the applicability of the load balancing algorithms in dynamic task environments, where tasks have different burst times and priorities. This reflects the complexity of real-world computing scenarios.

Contribution to System Performance Enhancement:

The project's overarching goal is to contribute to the broader discussion on enhancing system performance through the implementation of efficient load balancing strategies in multiprocessor scheduling.

Guidance for Practical Implementation:

The project outcomes aim to provide guidance for the practical implementation of load balancing strategies in real-world multiprocessor scheduling scenarios, assisting system administrators and developers in optimizing their systems.

**Architecture:**

Simulation Environment:

Representation of two processors with task queues.

Algorithm 1: Round-Robin Load Balancing:

Block for round-robin scheduling, including task queue management.

Algorithm 2: Priority-Based Load Balancing:

Block for priority-based task distribution using Python's multiprocessing.

Real-Time Updates and Visualization:

Block for capturing and visualizing dynamic task distribution.

Performance Metrics and Analysis:

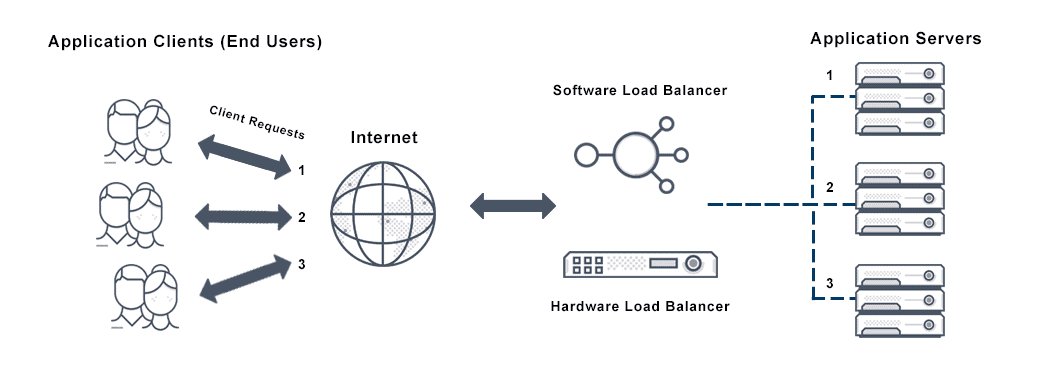
Block for calculating and analysing waiting times, task execution times, and system throughput.

User Interface (Optional):

Block for a graphical interface (GUI) allowing user interaction and visualization.

External Libraries and Dependencies:

Blocks representing any external tools or libraries used.



**Outcomes:**

The practical outcomes of this multiprocessor load balancing project provide actionable solutions for optimizing task distribution in computer systems. The implemented algorithms, real-time updates, and performance metrics offer practical tools for system administrators to minimize waiting times, enhance task execution efficiency, and improve overall system throughput. The project's documentation and recommendations guide decision-making, enabling administrators to implement efficient load balancing strategies based on the specific requirements of their systems. The optional user interface component adds accessibility, allowing administrators to interact with the system in real-time. In summary, the project's practical outcomes empower administrators with the tools and insights needed to enhance the performance of multiprocessor systems in real-world applications.

Github link : https://github.com/adithya3108/os\_project