**NANYANG TECHNOLOGICAL UNIVERSITY**

**SC4079 – Final Year Project**



**Project Plan: Evaluating Job Scheduling Algorithms in Clouds**

**Dixit Ayushman U2121836F**

# Table of Contents

[Table of Contents 2](#_Toc178698252)

[1. Introduction 3](#_Toc178698253)

[1.1. Project Overview 3](#_Toc178698254)

[1.2. Goals 3](#_Toc178698255)

[1.3. Motivation 3](#_Toc178698256)

[2. Related Works 3](#_Toc178698257)

[2.1. Traditional Scheduling Algorithms 4](#_Toc178698258)

[2.2. Cloud Specific Scheduling Algorithms 4](#_Toc178698259)

[2.3. Machine Learning Algorithms 4](#_Toc178698260)

[3. Proposed Idea 5](#_Toc178698261)

[3.1. Overview of Idea 5](#_Toc178698262)

[3.2. Evaluation of Stratus (Done as a team) 5](#_Toc178698263)

[3.3. Individual Algorithm – Q-learning 5](#_Toc178698264)

[3.4. Evaluation of algorithms (Done as a team) 6](#_Toc178698265)

[3.5. Proposed Project Timeline 8](#_Toc178698266)

[4. References 9](#_Toc178698267)

# Introduction

## Project Overview

As cloud computing continues to grow, efficient job scheduling has become a critical issue in managing resources across large-scale distributed systems. In cloud environments, jobs often need to be executed in parallel across multiple virtual machines (VMs) or containers, which are spread across different datacenters. However, due to the competition for limited resources, it becomes challenging to allocate them efficiently and fairly. The goal of this project is to analyze job scheduling algorithms using real-world job trace data from cloud platforms such as Google and Alibaba.

The project will evaluate the cost-effectiveness of several algorithms, starting with the Stratus Algorithm, which has been widely used for optimizing job scheduling in cloud environments. Additionally, the project will explore more advanced techniques, such as machine learning algorithms, to see if these methods can further improve efficiency, particularly in resource usage and cost optimization.

## Goals

The primary goals of this project are:

1. **Understand Cloud Job Scheduling**: Explore how jobs are scheduled in cloud computing environments and the specific challenges involved, such as resource limitations, parallel tasks, and dynamic workloads.
2. **Evaluate the Stratus Algorithm**: Benchmark the performance of the Stratus Algorithm, focusing mainly on cost-effectiveness.
3. **Compare Other Algorithms**: Evaluate and compare other scheduling algorithms to see how they perform relative to Stratus.
4. **Propose Potential Improvements**: Based on the results, identify gaps in the existing algorithms and, if possible, suggest modifications or new approaches to improve scheduling efficiency.

## Motivation

Job scheduling is a key aspect of cloud computing because it directly impacts the cost and performance of applications running on cloud infrastructure. In cloud environments, where services are billed based on resource usage, poorly designed scheduling algorithms can result in increased costs or inefficient use of resources. As cloud providers continue to scale their operations, there is a need for more effective algorithms that can handle complex, dynamic workloads and optimize the use of resources.

One of the exciting opportunities in this project is the potential to apply machine learning techniques, which can learn and adapt to real-time job patterns. These techniques have shown promise in other areas and could potentially lead to more intelligent and adaptive scheduling strategies in cloud computing. By analyzing and comparing different scheduling algorithms, this project aims to find solutions that not only reduce costs but also improve the overall efficiency of cloud systems.

# Related Works

Job scheduling in cloud environments has been an active area of research due to the complexity of resource management in distributed systems. Numerous algorithms have been developed over the years to address the challenges associated with resource allocation, job prioritization, and cost efficiency. In this section, we review some of the key works related to job scheduling algorithms, focusing on both traditional and more recent approaches that aim to optimize cloud environments.

## Traditional Scheduling Algorithms

Early job scheduling algorithms focused on simple heuristics and were primarily designed for grid computing environments, which later laid the groundwork for cloud-based scheduling. One of the most widely studied algorithms is First Come First Serve (FCFS), where jobs are executed in the order they arrive. While FCFS is straightforward and easy to implement, it often leads to inefficient resource usage and increased job completion times due to its lack of prioritization.

Another common approach is the Shortest Job First (SJF) algorithm, which prioritizes jobs with the smallest execution time. SJF aims to reduce overall job completion time, but it has limitations, especially in cloud environments, where estimating job execution time is not always reliable.

## Cloud Specific Scheduling Algorithms

With the advent of cloud computing, new scheduling algorithms were developed to address the unique characteristics of the cloud, such as dynamic resource allocation and the pay-per-use pricing model. The Stratus Algorithm, which is central to this project, was specifically designed for optimizing cost-effectiveness in cloud environments. [1] Stratus focuses on dynamically adjusting resource allocation based on job runtime estimates, which can lead to significant cost savings by optimizing the use of cloud resources. However, the performance of Stratus heavily relies on accurate job runtime predictions, which can be a challenge in real-world environments where job characteristics vary significantly.

In addition to Stratus, several other cloud-specific algorithms, such as Min-Min and Max-Min, have been explored. Min-Min selects the smallest jobs to be executed first, aiming to free up resources quickly, while Max-Min prioritizes larger jobs. Both approaches have their advantages, but they tend to focus on job completion times rather than cost optimization, which is a crucial factor in cloud environments.

## Machine Learning Algorithms

In recent years, machine learning has been applied to job scheduling to develop adaptive algorithms that can learn from past data and improve decision-making over time. Reinforcement Learning (RL), in particular, has gained attention for its ability to optimize complex scheduling scenarios.

These techniques has been demonstrated to improve both resource utilization and cost efficiency compared to traditional heuristic-based algorithms.[2] However, one of the challenges with machine learning-based approaches is the computational cost and training time required, especially when using large-scale datasets like the ones provided by Google and Alibaba.

# Proposed Idea

## Overview of Idea

In this project, we aim to evaluate the cost-effectiveness and performance of various job scheduling algorithms in cloud environments, starting with the Stratus Algorithm as the benchmark. We will also explore the application of other algorithms to potentially improve scheduling efficiency and fairness. This section outlines the methodologies for evaluating Stratus and provides a rough implementation plan for other algorithms.

## Evaluation of Stratus (Done as a team)

The Stratus Algorithm is designed to optimize resource utilization and reduce cloud costs, making it an ideal baseline for this project. The evaluation of Stratus will focus on the following steps:

1. Simulation Setup: We will simulate a cloud environment using real job trace data from production systems, such as the Google Cluster Data and Alibaba Cluster Data. This simulation will allow us to replicate real-world cloud workloads and evaluate the performance of the Stratus Algorithm under realistic conditions.
2. Performance Metrics: The key metric for evaluating Stratus will be cost-effectiveness, which is crucial in cloud environments where resources are charged on a pay-per-use basis. Other secondary metrics will include:
   * Resource Utilization: Measuring how efficiently Stratus allocates and uses available resources (CPU, memory, etc.).
   * Job Completion Time: Analyzing how quickly jobs are completed under Stratus scheduling.
   * Fairness: Assessing whether Stratus ensures a fair distribution of resources among competing jobs, preventing monopolization of resources.
3. Comparative Analysis: Stratus will be compared with other traditional scheduling algorithms, such as First Come First Serve (FCFS) and Min-Min, to highlight its advantages and disadvantages. By running these algorithms on the same job trace data, we can compare their performance in terms of cost savings, resource efficiency, and overall cloud workload management.
4. Results Interpretation: The results of the Stratus Algorithm will be thoroughly analyzed, focusing on its strengths in cost optimization as well as any weaknesses, such as reliance on accurate job runtime predictions. This analysis will set the foundation for comparing Stratus with other more advanced algorithms.

## Individual Algorithm – Q-learning

Q-learning algorithm offers a flexible and adaptive approach for job scheduling in cloud environments. [3] In this project, it will be implemented and simulated using real-world job trace data from Google and Alibaba, aiming to assess its effectiveness compared to the Stratus Algorithm, particularly in terms of cost optimization.

Rough Steps for Implementation:

1. Preprocessing the Dataset: The first step is to preprocess the job trace data to a format compatible with Q-learning simulations. This will involve:
   * Extracting relevant features from the dataset, such as job submission times, resource requests (CPU, memory), and job execution status (success or failure).
   * Structuring the data to represent different states of the cloud environment, such as the available resources and the queue of jobs waiting for execution.
2. State and Action Definitions:
   * State: The state in this context will represent the current status of the cloud environment, including the number of jobs in the queue, available resources (CPU, memory, VMs), and the progress of running jobs.
   * Action: The actions will involve scheduling decisions, such as selecting which job to schedule next and determining the amount of resources to allocate for each job.
3. Simulation of Q-Learning on the Dataset:
   * A simulation environment will be set up where the Q-learning algorithm interacts with the pre-processed job trace data.
   * The algorithm will use the Q-table to store the state-action pairs, gradually updating the table as it receives feedback from the environment in the form of rewards.
   * The simulation will run multiple iterations, allowing the algorithm to explore different scheduling strategies and optimize its policy for minimizing resource costs and improving job completion times.
4. **Reward Structure**: The reward function will guide the Q-learning algorithm toward optimal scheduling decisions. It will be designed to reward:
   * Cost Savings: Positive rewards for actions that minimize the overall cost of resource usage.
   * Efficient Job Scheduling: Rewards for reducing job completion time and utilizing resources effectively.
   * Resource Optimization: Incentives for actions that minimize idle resources or avoid over-provisioning.

This reward structure will help the algorithm focus on cost-effectiveness, which aligns with the overall goal of the project.

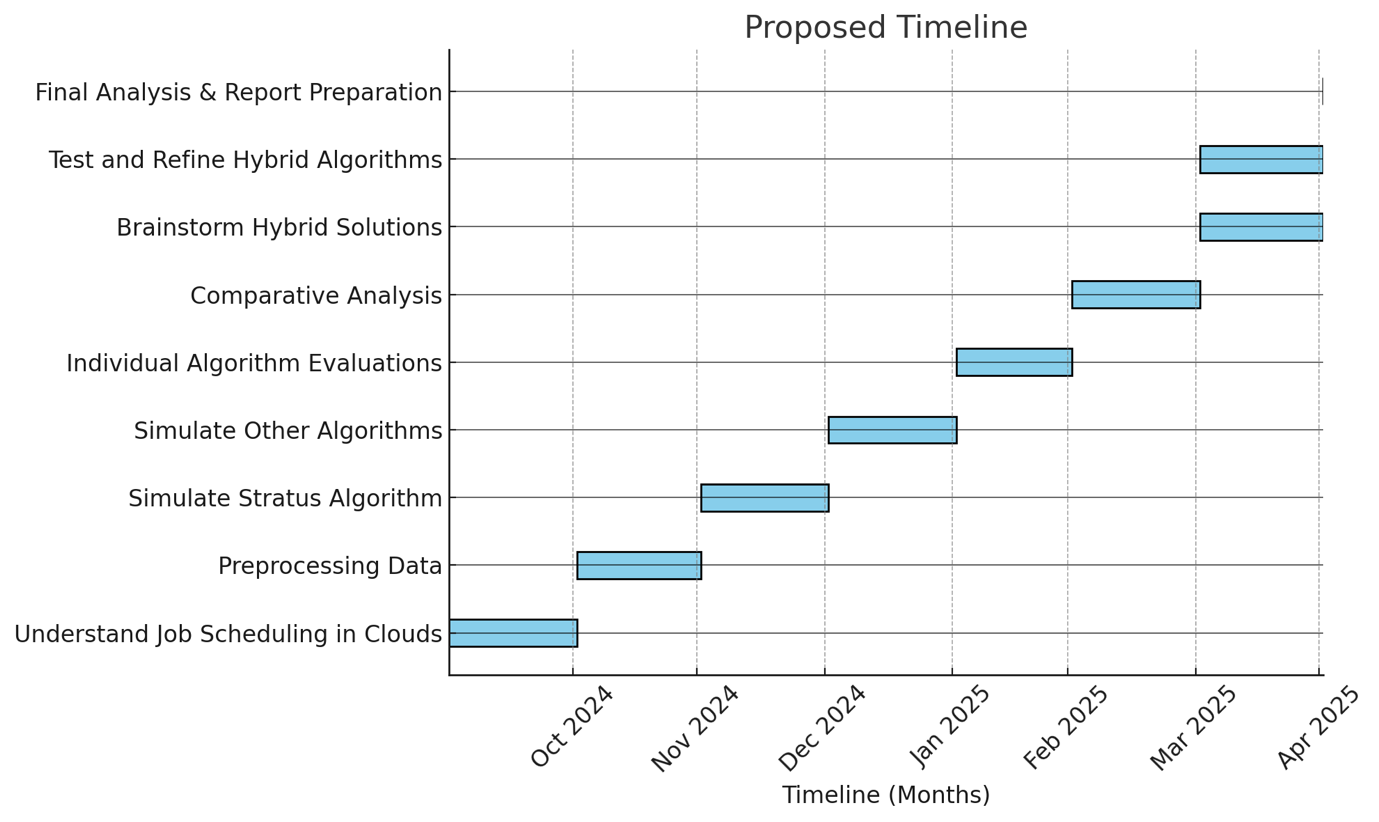
1. Evaluation and Comparison:
   * The performance of the Q-learning algorithm will be compared to the Stratus Algorithm using the same job trace data.
   * Key performance metrics such as cost optimization, resource utilization, and job completion time will be used to assess how well the Q-learning algorithm performs in a simulated cloud environment.
   * The results will provide insights into whether the Q-learning approach offers any advantages over Stratus, particularly in terms of reducing costs while maintaining efficiency.

## Evaluation of algorithms (Done as a team)

The evaluation of the various job scheduling algorithms will be a collaborative effort, where each team member will be responsible for simulating a specific algorithm. By analyzing the performance of each algorithm under similar conditions, we will be able to compare their strengths and weaknesses in a comprehensive and systematic manner.

1. Individual Simulations: Each team member will run simulations of their chosen scheduling algorithm, using the same real-world job trace data from Google or Alibaba. These simulations will be conducted in a controlled environment to ensure consistency across the evaluations. The key performance metrics, including cost-effectiveness, resource utilization, job completion time, and fairness, will be recorded for each algorithm.
2. Performance Comparison: Once the individual simulations are complete, the team will come together to compare the results. We will create comparative charts and tables to analyze which algorithms perform better under specific conditions. For example, some algorithms may excel at minimizing costs, while others may be more effective at reducing job completion time or maximizing resource efficiency.
3. Identifying Gaps and Potential Improvements: After evaluating each algorithm’s performance, we will collectively identify any gaps or inefficiencies in their scheduling strategies. For instance, one algorithm may struggle with resource utilization under certain workloads, while another may have high costs due to inefficient job prioritization.
4. Exploring New/Hybrid Algorithms: Based on the insights gained from the evaluation, the team will brainstorm potential new approaches or hybrid algorithms that combine the strengths of the different scheduling methods. For example, we may explore a hybrid model that leverages the cost-saving features of the Stratus Algorithm while incorporating the adaptive learning capabilities of machine learning algorithms. By merging the best aspects of the evaluated algorithms, we hope to design a more efficient and cost-effective scheduling solution for cloud environments.
5. Final Analysis and Documentation: Once we have explored the performance of all algorithms and potentially developed new or hybrid approaches, the final step will be to document the comparative analysis. This will include detailed results, charts, and explanations for why certain algorithms performed better in specific scenarios. The team will also outline any proposed new algorithms and highlight their potential benefits over existing methods.

## Proposed Project Timeline



**Figure 1: Gantt Chart of FYP Project**

# References

1. <https://pdl.cmu.edu/PDL-FTP/CloudComputing/p121-Chung.pdf>
2. <https://journals.uob.edu.bh/bitstream/handle/123456789/5604/1571017581.pdf?sequence=1&isAllowed=y>
3. <https://www.sciencedirect.com/science/article/abs/pii/S0020025519309971>