Technical Summary

Team Members: Muneeba Badar and Shifa Shah

April 8, 2025

1 Problem and Contribution

The paper addresses the issue of inefficient resource utilization and longer execution times in cloud computing environments caused by the original Min-Min heuristic's lack of consideration for load distribution when scheduling workflows. To solve this, the paper's main algorithmic contribution is a modified Min-Min heuristic that integrates load-balancing principles. This new method is designed for dependent tasks represented as Directed Acyclic Graphs (DAGs), respects precedence constraints, utilizes a new priority attribute for task selection, and includes a step to duplicate the entry task across all virtual machines. The algorithm operates in two phases: task selection based on priority and resource allocation with entry task duplication, aiming to optimize load distribution and minimize overall workflow execution time.

2 Algorithmic Description

The proposed algorithm enhances workflow scheduling in cloud environments by modifying the Min-Min heuristic to consider task dependencies and load balancing. It prioritizes tasks based on their earliest completion time while maintaining precedence constraints and introduces entry-task duplication to reduce communication overhead and improve execution time.

2.1 Inputs and Outputs

Input: Workflow as a Directed Acyclic Graph (DAG) and Estimated Computation Time (ECT) for each task on each machine.
Output: Makespan (total completion time), Load Balancing (resource utilization), and QoS metrics (Speedup, Efficiency).

2.2 High-Level Logic

The modified algorithm addresses the limitations of the traditional Min-Min heuristic through two main phases and a modification regarding the entry task:

- Phase 1: Task Selection based on Priority and Precedence Constraints (PC): Instead of simply selecting the task with the overall minimum completion time, the modified approach calculates a priority for each task. This priority considers the minimum Estimated Completion Time (ECT) of the task across all available virtual machines while ensuring that the precedence constraints of the workflow (dependencies between tasks) are satisfied. Tasks whose predecessors have been completed are eligible for selection and are placed into a priority queue (PQ)

- Phase 2: Resource Selection with Entry Task Duplication: Tasks are dequeued from the priority queue based on their assigned priority. For entry task, the proposed algorithm duplicates the entry task and allocates it to all available virtual machines to reduce the communication time between the entry task and its immediate successor tasks. For subsequent tasks, the algorithm determines the Earliest Start Time (EST) and Earliest Finish Time (EFT) on each virtual machine, considering the completion times of their predecessors and the communication times. The task is then allocated to the virtual machine that results in the earliest finish time, aiming to minimize the overall makespan.

2.3 Example/Diagram

The following flowchart illustrates the two main phases of the algorithm and the steps involved in each phase.

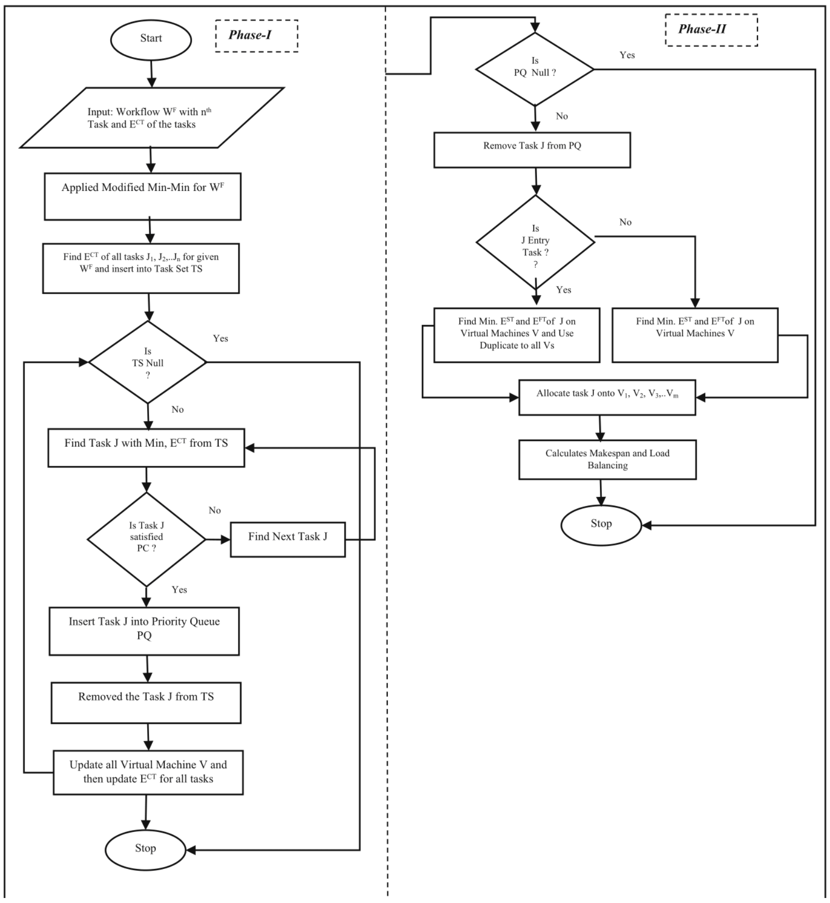
Phase-I
Start
Input: Workflow W^F with n^{th} Task and E^{CT} of the tasks
Applied Modified Min-Min for W^F
Find E^{CT} of all tasks $J_1, J_2,..J_n$ for given W^F and insert into Task Set TS
Is TS Null ?
Yes
No
Find Task J with Min, E^{CT} from TS
Is Task J satisfied PC ?
No
Find Next Task J
Yes
Insert Task J into Priority Queue PQ
Removed the Task J from TS
Update all Virtual Machine V and then update E^{CT} for all tasks
Stop
Phase-II
Is PQ Null ?
Yes
No
Remove Task J from PQ
Is J Entry Task ? ?
No
Yes
Find Min. E^{ST} and $E^{FT}$of J on Virtual Machines V and Use Duplicate to all Vs
Find Min. E^{ST} and $E^{FT}$of J on Virtual Machines V
Allocate task J onto $V_1, V_2, V_3,..V_m$
Calculates Makespan and Load Balancing
Stop

Figure 1: Flowchart of the algorithm

3 Comparison

The proposed modified Min-Min heuristic improves workflow scheduling by handling dependent tasks within a DAG while preserving precedence constraints. It enhances load balancing by considering both task completion time and resource load and introduces a novel entry-task duplication strategy to reduce communication overhead. Extensive simulations show that it outperforms heuristics like HEFT and PETS in makespan, speedup, efficiency, and resource utilization, leading to better-balanced resource allocation and overall system performance.

4 Data Structures and Techniques

The modified Min-Min heuristic utilizes the following key data structures and mathematical tools:

- **Directed Acyclic Graph (DAG):** Represents tasks and their dependencies, where nodes correspond to tasks and edges indicate communication links.
- **Priority Queue (PQ):** Stores tasks based on priority, ensuring that precedence constraints are satisfied before scheduling.
- **ECT Matrix/Table:** A matrix or table stores the Estimated Computation Time (ECT) of each task on different virtual machines for efficient lookup.
- **Min/Max Operations:** The algorithm finds the minimum ECT and Earliest Finish Time (EFT) for scheduling while using maximum values to determine Earliest Start Time (EST) and makespan.

5 Implementation Outlook

The implementation of the modified Min-Min heuristic may face several technical challenges:

- **Numerical Precision Issues:** Calculations involving Estimated Computation Time (ECT), Earliest Start Time (EST), and Earliest Finish Time (EFT) may suffer from floating-point precision errors, leading to inaccuracies in scheduling decisions.
- **Handling Large Input Sizes:** Complex workflows represented as Directed Acyclic Graphs (DAGs) with numerous tasks and dependencies may result in high memory usage and computational overhead, requiring efficient memory management.
- **Computational Complexity:** The addition of priority-based task selection and entry task duplication increases the time complexity of the modified Min-Min heuristic compared to the traditional approach, potentially impacting performance for large-scale workflows.
- **Scalability Concerns:** As the number of tasks and virtual machines increases, scheduling decisions become more complex, necessitating optimization techniques to maintain efficiency and achieve effective load balancing.
- **Communication Overhead:** The duplication of the entry task across multiple virtual machines may introduce synchronization and communication overhead, potentially affecting overall performance if not handled efficiently.