Efficient Task Scheduling and Load Balancing in Multi-Agent Systems

Videep Ekbote

Abstract

In logistics and supply chain management, efficiently packing diverse cargo into containers is a critical problem. This project addresses dynamic bin packing using various algorithmic approaches, specifically designed for managing bins with fixed capacities and objects of different sizes and colors. Each cargo requires special handling based on its characteristics, and the system assigns appropriate bins using optimized strategies. This paper explores two main algorithms — Largest Fit and Compact Fit — and demonstrates their applications in efficient cargo allocation. This project uses the most efficient way of solving the bin packing problem in logarithmic time complexity, using AVL trees.

Introduction

Task scheduling and load balancing are fundamental problems in distributed systems, especially in multi-agent systems (MAS) where multiple entities work together to achieve a common objective. In such environments, efficient scheduling is required to avoid bottlenecks and ensure equitable task distribution. This paper introduces a model wherein tasks of varying sizes arrive at different times and must be processed by agents (or "crewmates") with varying loads. The objective is to balance the workload while minimizing waiting times and optimizing task completion.
We focus on a scenario involving treasure management, where each treasure (task) has a specific processing time, and agents are responsible for managing multiple treasures. We propose a scheduling and load-balancing strategy based on priority and load distribution policies.

System Model

The problem is modeled as a scheduling task with multiple agents responsible for processing tasks (treasures). We define the system formally as follows:

Task Definition

Each task (treasure) j is characterized by:

- **Treasure ID** (id_j): A unique identifier for each task.
- **Treasure Size** ($size_j$): The time required to process the task, given in arbitrary time units.
- **Arrival Time** ($arrival_j$): The time at which the task becomes available for processing.

System State

At any time t, the state of the system is defined by:

- **Remaining Size**: For each task j, the remaining size is given by:

$$remaining_j = size_j - \text{processed time}$$

- **Load on Agents**: The load on agent i is the total remaining size of tasks assigned to them. At any given time, each agent can only process one task.

Scheduling Policy

We introduce a scheduling policy based on two principles: *load balancing* and *priority-based task processing*. Our objective is to assign newly arriving tasks to the agent with the least load and to process the task with the highest priority.

Task Assignment

When a new task arrives, it is assigned to the agent with the least current load, calculated as the total remaining size of the tasks assigned to that agent. Formally, let $L_i(t)$ denote the load of agent i at time t. A new task j is assigned to the agent i^* such that:

$$i^* = \arg\min_i L_i(t)$$

In the case where multiple agents have the same minimum load, the task is assigned arbitrarily among them.

Task Processing

Each agent processes tasks based on their priority, where the priority of task j at time t is defined as:

$$Priority(j) = \text{Wait Time}(j) - \text{Remaining Size}(j)$$

The *Wait Time* is defined as the time elapsed since the arrival of the task:

$$WaitTime(j) = t - arrival_j$$

The agent processes the task with the highest priority. In the case of a tie, the task with the lowest id_j is processed first.

Algorithm Design

We implement the proposed scheduling policy using heap-based data structures for efficient task management. The heap operations allow for efficient insertion and extraction of tasks, crucial for maintaining low time complexity during task assignment and processing.

Heap Operations

The heap structure is utilized to store tasks and agent loads efficiently. The heap supports the following operations:

- **Insert**: Insert a new task into the heap. Time complexity: $O(\log n)$.
- **Extract Min**: Extract the task with the minimum load. Time complexity: $O(\log n)$.

Performance Analysis

The performance of the proposed scheduling policy is evaluated in terms of time complexity and task completion times. The key factors affecting the performance include the number of agents (m) and the number of tasks (n).

Time Complexity

The time complexity of task assignment is $O(\log m)$ due to heap operations for maintaining agent loads. Task processing, which involves selecting the highest-priority task, has a complexity of $O(\log n)$.

Load Balancing Efficiency

We evaluate the load balancing efficiency by examining the variance in agent loads over time. The scheduling policy ensures that the load distribution is equitable across agents, minimizing idle time and maximizing system throughput.

Conclusion

This paper presents a scheduling policy for task assignment and processing in multi-agent systems, focusing on load balancing and task prioritization. Our approach efficiently distributes tasks among agents and processes tasks in a manner that reduces wait times and balances the overall system load. Future work could extend this model to more complex task types and agent capabilities, making the system more adaptable to real-world distributed computing challenges.