Chinese University of Hong Kong,
Shenzhen

CSC4303: Network Programming

Final Project Report

Author:

120090266 Yutong Feng

120090222 Zhixin Chen

120090716 Jiayu Chen

1 Introduction

The open-source Lux project aims to enhance the video download process by introducing an advanced scheduling system. Traditional methods in open-source video downloaders, though effective, often lack adaptability to varying network conditions and user preferences. Our approach involves a globally accessible task queue and an adaptable thread pool that optimizes the downloading process by dynamically adjusting to real-time network conditions and prioritizing tasks based on user-defined settings.

2 System Design and Implementation

2.1 System Architecture

The proposed system architecture incorporates a task queue and a thread pool. The task queue stores video download tasks with attributes like URL, video segment start, and end points. The thread pool is engineered to dynamically access tasks from the queue, optimizing resource utilization and adapting to changing network conditions.

2.2 Implementation Details

- **Initialization of Download Queue and Thread Pool:**
  - A globally accessible task queue is created at the start of the system to hold tasks, which are subdivided and enqueued as new download requests are received.
  - A thread pool is initialized where each thread can access and retrieve tasks from the queue, enhancing the efficiency of task management and execution.
- **Main Thread Scheduling Algorithm:**

- **Task Object Definition:** Each task object contains information such as the video source URL, start point, and endpoint. When a new video request is received, these tasks are subdivided and added to the global queue.
- **Task Scheduling Based on Dynamic Load Balancing:** The system includes a network monitoring module that collects and analyzes real-time network bandwidth and quality. It dynamically adjusts task allocation based on the current load of threads, using simple round-robin or more complex load-based strategies.
- Adjustments are made based on:
  * If the remaining data of a video file is small, its priority is increased to complete the download as soon as possible.
  * If a video file has finished downloading, the thread's resources are released and allocated to other video download tasks.
  * If a thread's download speed is significantly lower than others, its priority is temporarily decreased, and the download of other videos is prioritized.

- **Thread Workflow:**
  - Each thread starts by pulling tasks from the queue. It downloads the specified video segments, handling potential exceptions such as network errors.
  - Upon completing a task, the thread returns to the queue to pull the next task. If the queue is empty, the thread enters a sleep state until new tasks are enqueued.

- **Priority Management:**
  - Downloads are prioritized based on user-defined preferences which will be available in Parallel priority download mode.

- **Error Handling:**

  - The system supports resumable downloads and robust error handling mechanisms to manage download failures, ensuring reliability and continuity in downloading activities.

- **Mutiple Download Modes:**
  - The system supports three download modes: Origin Lux download mode, Parallel download mode, Parallel priority download mode for user preference.

3 Results

3.1 Mutiple Download Modes

The system supports three download modes: origin Lux download mode, parallel download mode, and parallel priority download mode by the input args.

```
// default lux download mode
// -1 as download mode para
lux -1 "site1" "site2"
go run main.go -1 "site1" "site2"

// parallel download mode
// -2 as download mode para
lux -2 "site1" "site2"
go run main.go -2 "site1" "site2"

// parallel priority download mode
// -3 as download mode para, -hp: has priority setting
// paras hp is a priority map. e,g: 1:3;2:5
// means site1 has priority 3, site2 has
// prioirty 5 (higher, download first)
lux -3 -hp "1:3;2:5" "site1" "site2"
go run main.go -3 -hp "1:3;2:5" "site1" "site2"

// see template results in readme
```

3.2 Performance Evaluation

Upon implementation, the system was subjected to a series of tests under varying network conditions. The performance metrics were documented through a line graph and a bar chart, showing a noticeable improvement in download speeds and efficiency compared to traditional methods. The tests confirmed that our dynamic load balancing significantly reduces the time to download multiple videos simultaneously.

As the number of videos increased, the enhancement in speed became more pronounced. Specifically, when downloading eight videos concurrently, our system demonstrated an approximate 80% reduction in download time compared to the baseline. This remarkable efficiency gain underscores the effectiveness of our optimization strategy, particularly in scenarios involving heavy data transfers. The graphically represented results, depicted in Figures 1 and 2, illustrate the scalability and robustness of our network enhancements.

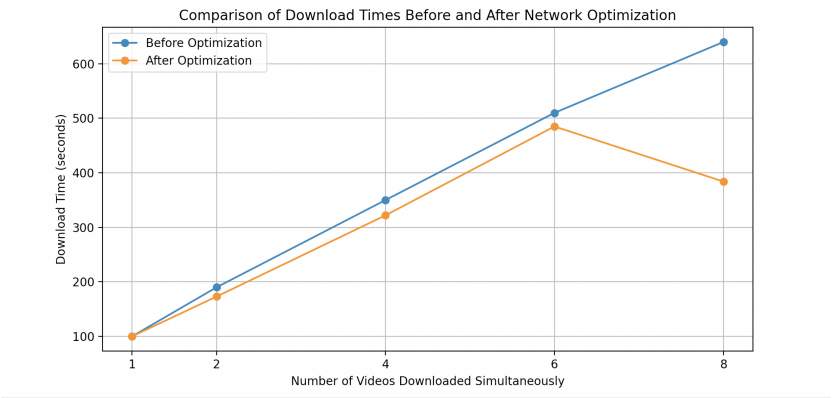

Figure 1: Line graph showing the improvement in download times as the number of videos increases.

3.3 Analysis and Discussion

The results clearly demonstrate the advantages of our system design. The dynamic task allocation effectively reduces idle time and enhances through-

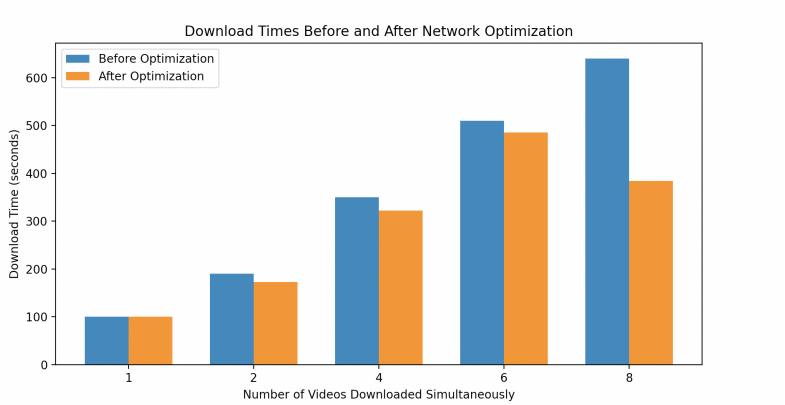

Figure 2: Bar chart comparison of download times before and after optimization for multiple videos.

put. The ability to prioritize tasks ensures that user preferences are respected, adding an extra layer of customization to the download process. The integration of error handling and resumable downloads enhances the robustness of the system, making it more reliable under different network conditions.

4 Conclusion

The project has successfully demonstrated a significant improvement in the efficiency and adaptability of the video downloading process for the Lux open-source project. The innovative use of a dynamic task queue and thread pool, coupled with real-time network monitoring, sets a new standard for open-source video downloaders. The system's ability to integrate user preferences and adapt dynamically to network conditions can serve as a model for future enhancements in similar applications.