Multithreaded Web Title Crawler - Code Description

This Python script demonstrates how to crawl multiple websites concurrently using multithreading to extract their HTML titles. It utilizes the `requests` and `BeautifulSoup` libraries for web requests and parsing, and `ThreadPoolExecutor` for concurrent execution.

Step-by-step Description:

1. \*\*Import Required Libraries\*\*:

- `requests`: To make HTTP requests to websites.

- `BeautifulSoup` from `bs4`: For parsing HTML content.

- `ThreadPoolExecutor` from `concurrent.futures`: To perform multithreading.

2. \*\*List of URLs\*\*:

- A list named `urls` is created, containing website URLs to fetch titles from.

3. \*\*fetch\_title(url)\*\*:

- This function takes a single URL.

- It sends an HTTP GET request to fetch the page content.

- Parses the HTML content using `BeautifulSoup`.

- Extracts the `<title>` of the page.

- Handles and returns any exceptions that may occur.

4. \*\*crawl\_sites(url\_list, max\_workers=5)\*\*:

- Accepts a list of URLs and an optional number of worker threads.

- Initializes a `ThreadPoolExecutor` with `max\_workers`.

- Uses `executor.submit()` to schedule each URL fetch.

- `as\_completed()` is used to process results as threads complete.

- Returns a list of results with either titles or error messages.

5. \*\*Main Execution\*\*:

- The script checks if it is the main module and calls `crawl\_sites()`.

- The result of each fetch (URL and title/error) is printed.

This script demonstrates the power of parallel processing in I/O-bound tasks such as web scraping, allowing significant speedup compared to sequential execution.