Simple Parallel Web Scraper

Functional and Parallel Programming

What is a web scraper ?

A web scraper is a software tool designed to extract data from websites. It works by visiting web pages, reading their content, and collecting specific information automatically.

Similar to Web Crawler in A3

Web crawler explore the internet systematically by following links from one page to another. It is used to index and catalog web pages meanwhile **web scraper f**ocuses on extracting specific data from web pages.

What tools are used in a parallel web scraper ?

Tools Used:

1. **Jsoup:** A Java library for working with HTML, providing easy methods for parsing, extracting, and manipulating data from HTML pages.
2. **Scala Futures:** For handling asynchronous computations. Futures represent a value that may become available at some point, allowing non-blocking operations.
3. **Await**: The Await.result method is used to block the main thread until all the Futures complete. It's important to note that while this method blocks the main thread, the individual Futures are still running in parallel.

Parallel vs Sequential

Efficiency:

- **Parallel Web Scraper:** Utilizes concurrent execution with `Futures`, allowing multiple URLs to be processed simultaneously, improving speed, especially for I/O-bound tasks. It has robust error handling that recovers from individual failures without affecting the overall process.
- **Sequential Web Scraper:** Processes URLs one by one, which can be inefficient for large datasets as it does not leverage concurrent execution. Error handling is basic and may cause the process to stop if an exception occurs.

Scalability:

- **Parallel Web Scraper:** Better for handling large-scale scraping tasks due to concurrent processing. It can utilize multi-core processors but may be limited by system hardware and server rate limits. Higher resource consumption is possible.
- **Sequential Web Scraper:** Less scalable as it processes URLs sequentially, causing performance to degrade linearly with the number of URLs. It uses fewer resources but does not leverage multi-core processors.

Overall Performance:

- **Parallel Web Scraper:** Generally faster due to simultaneous processing of multiple requests, though it is more complex to implement and manage due to concurrency and potential thread safety issues.
- **Sequential Web Scraper:** Slower as it handles URLs one at a time. It is simpler to implement and manage but may become inefficient for large datasets.

Benchmark Tests

1. **Execution Time:**
   - **Parallel Web Scraper:** The execution time is expected to be significantly shorter compared to the sequential scraper. This is because the parallel scraper can handle multiple web requests simultaneously, which reduces the overall time needed to complete the task, especially when dealing with a large number of URLs.
   - **Sequential Web Scraper:** The execution time will be longer, as it processes each URL one after another. The total time will increase linearly with the number of URLs and the time taken to scrape each URL.
2. **Resource Utilization:**
   - **Parallel Web Scraper:** Utilizes more CPU and network resources due to concurrent execution. If the system has multiple cores, the parallel scraper will take advantage of them, potentially leading to higher resource consumption. This could also result in increased load on the web server being scraped.
   - **Sequential Web Scraper:** Utilizes fewer resources as it operates in a single-threaded manner. CPU and network usage will be lower, but the overall efficiency will be reduced due to the lack of concurrency.