**PARALLEL AND DISTRIBUTED COMPUTING**

**ASIIGNMENT NO. 2**

**Parallelized Web Crawler with Load Balancing**

**Group Members:**

|  |  |
| --- | --- |
| **NAME** | **CMS ID** |
| Muhammad Ammar bin Akram | 414563 |
| Malik Muhammad Aman | 409918 |
| Hannan Yousaf Butt | 405326 |

**1. Introduction**

This report analyzes the performance of a web crawler implemented in three versions:

* **Serial** **(Single-threaded)**
* **Multithreaded**
* **MPI-based Distributed**

The goal was to evaluate the **execution time**, understand the **design trade-offs**, and assess the **scalability** of each version using a fixed URL workload.

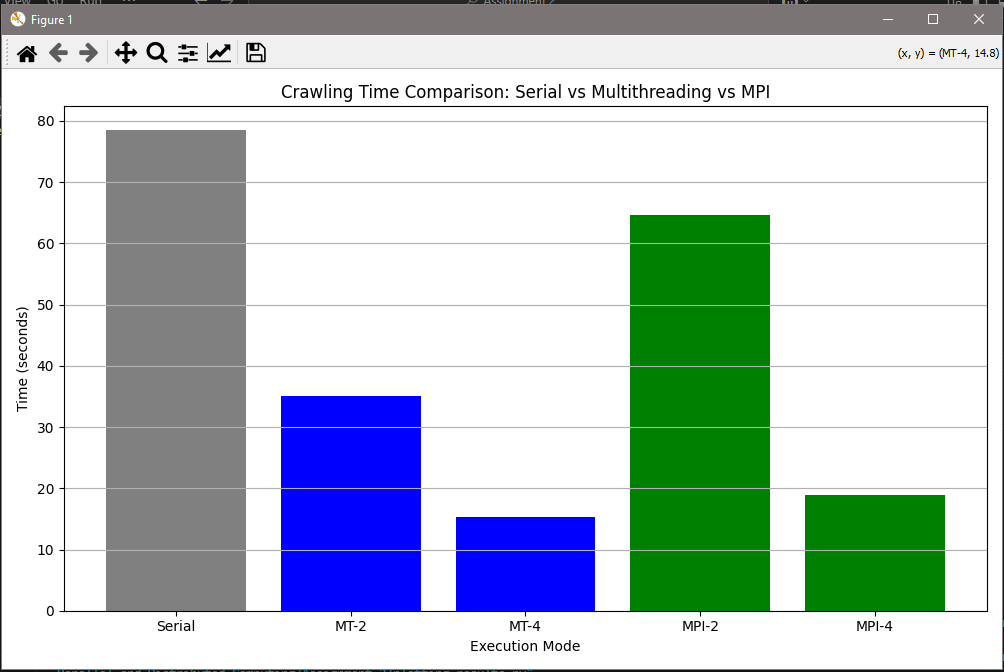
#### ****2. Benchmark Results****

Below are the crawl timings obtained from crawling a fixed set of web pages (50):

|  |  |  |
| --- | --- | --- |
| Version | Configuration | Time taken |
| Sequential | - | 78.47 seconds |
| Multithreaded | 2 threads | 35.01 seconds |
| Multithreaded | 4 threads | 15.31 seconds |
| MPI (Distributed) | 2 workers | 64.68 seconds |
| MPI (Distributed) | 4 workers | 18.85 seconds |

**Visual Comparison:**

A bar chart was generated to visually compare the performance. It shows that both multithreading and MPI significantly reduce the crawl time, especially when the number of workers/threads increases.



#### ****3. Design Trade-offs****

##### **Multithreading:**

* **Pros:**
  + Easy to implement and deploy
  + Shared memory leads to low communication overhead
  + Scales well up to moderate thread counts
* **Cons:**
  + Python’s Global Interpreter Lock (GIL) can limit CPU-bound performance
  + Threads share memory — potential for data corruption without locks

##### **MPI-based Distributed Crawling:**

* **Pros:**
  + Suitable for large-scale distributed environments
  + Can scale across multiple physical machines
* **Cons:**
  + Higher communication overhead due to inter-process messaging
  + Requires MPI setup, making deployment more complex
  + Debugging across processes is harder than with threads

#### ****4. Scalability Analysis****

##### **Speedup (vs. Serial):**

* Multithreaded (2 threads): 78.47 / 35.01 ≈ **2.24×**
* Multithreaded (4 threads): 78.47 / 15.31 ≈ **5.12×**
* MPI (2 workers): 78.47 / 64.68 ≈ **1.21×**
* MPI (4 workers): 78.47 / 18.85 ≈ **4.16×**

##### **Observations:**

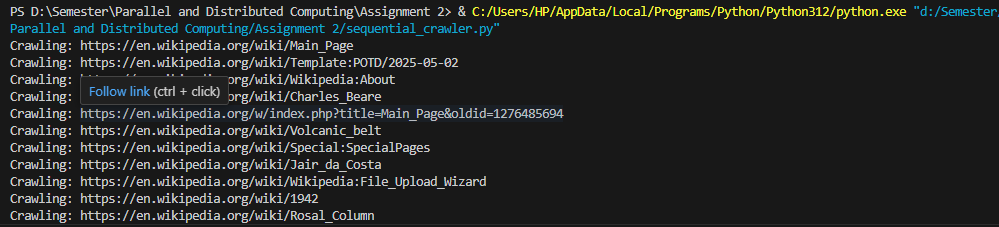
* **Multithreading** achieves better scaling at low thread counts due to minimal overhead.
* **MPI** shows improved performance with more workers, but suffers from higher overhead at low counts (2 workers).
* **Diminishing returns** begin to appear as the number of threads/workers increases, especially if the workload is not large enough to fully utilize them.

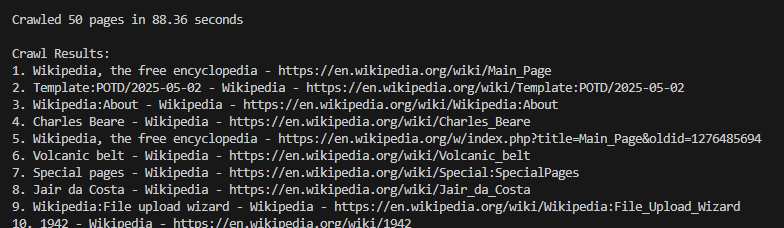
**5. Conclusion & Recommendations**

* For **moderate workloads**, **multithreading is the most efficient** in terms of simplicity and performance.
* For **scalable distributed crawling across multiple machines**, **MPI is more suitable**, albeit with increased complexity.
* To further improve performance:
  + Optimize HTML parsing (e.g., limit tags extracted)
  + Avoid re-crawling duplicate URLs
  + Consider asynchronous I/O (e.g., using aiohttp) for I/O-bound performance

**ScreenShots of Code Running:**

**Sequential Crawler:**

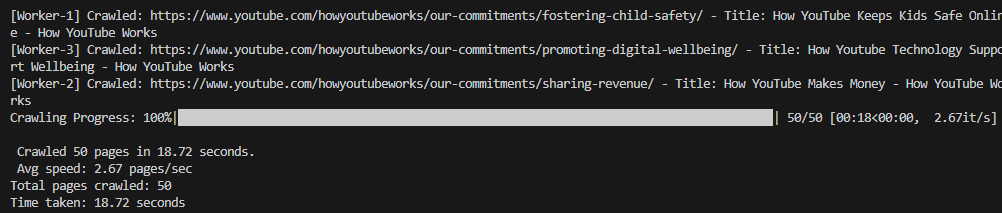
****

****

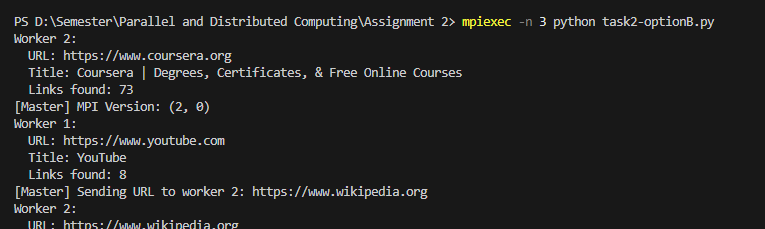
**Multi-Threaded Crawler:**

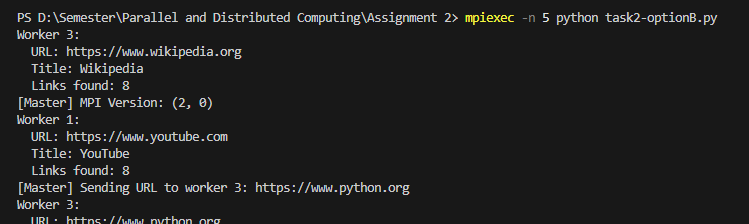
****

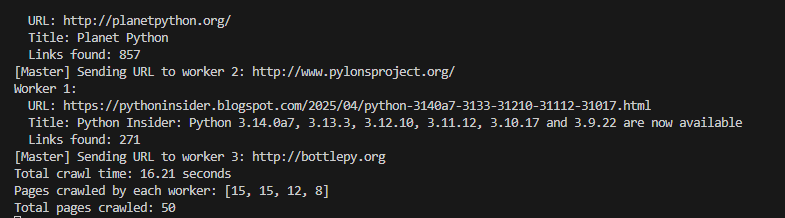
****

****

**MPI based Crawler:**

****

****

****