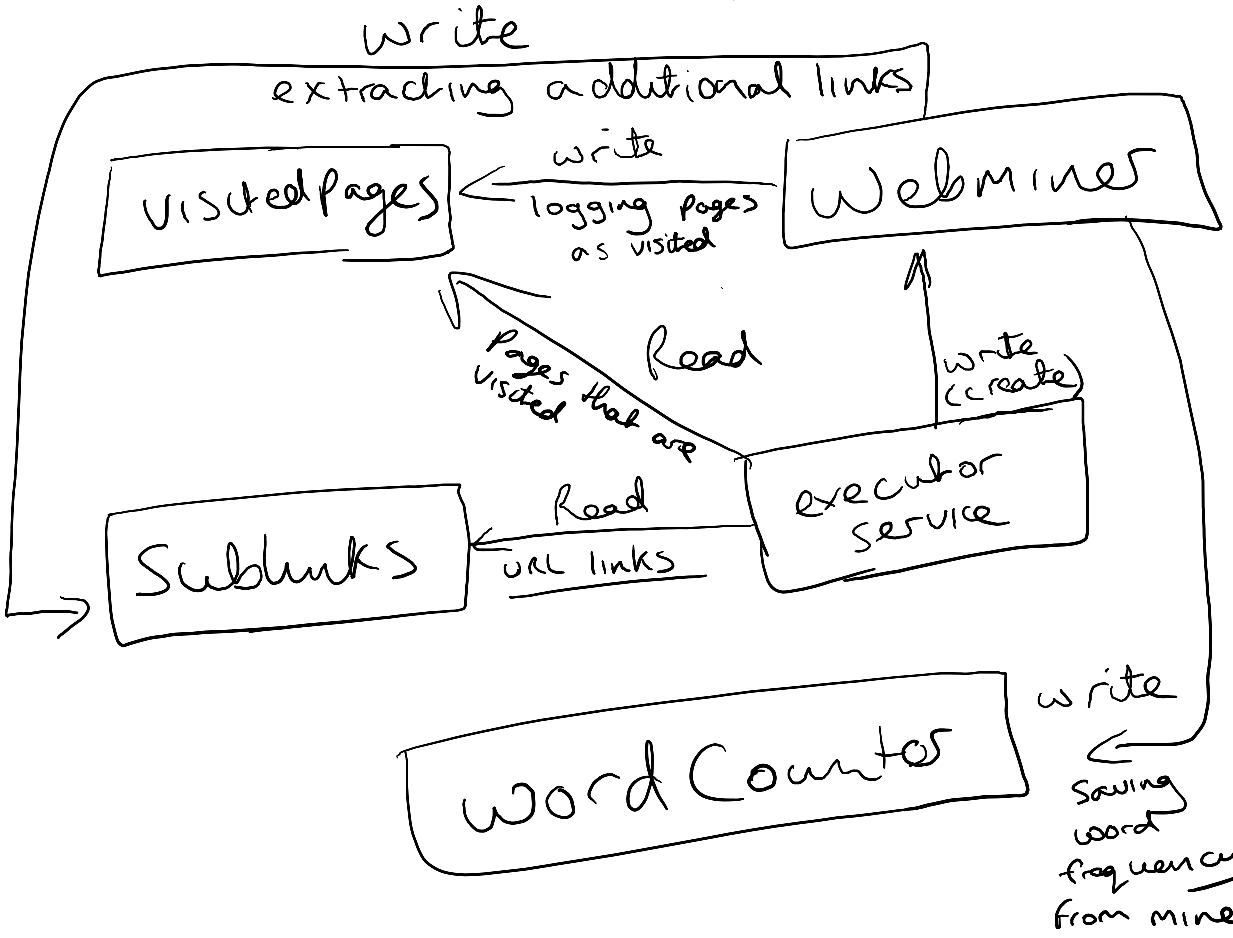
CO3090: Distributed System Coursework 1

Task 3:

1. The first characteristic that makes a multi-threading implementation better than a single threaded counterpart is the total time required for doing the task will be shorter, because in this context, a single threaded program will have to juggle going through each webpage, saving the data and moving on, this wastes efficiency as it will waste time fetching and managing the all of the URLs, thus it will take longer to do the same amount of work as the multi-threaded version because it has to do EVERYTHING. The multi-threaded program can make more efficient use of time because different threads can be delegated roles/tasks to perform and just report back, e.g its quicker to have 100 workers to 1 tasks each, against 1 worker doing 100 tasks. Hence efficiency is one characteristic.

1.2. The 2nd characteristic which makes the first characteristic possible is the parallelization of work load. The main thread starts a MinerManager and just waits, the MinerManager only has to fetch URLs and collect results, while it gives the tasks to the WebMiners on other threads to do the work simultaneously, meaning the mining isn't happening one by one, they are happening at the same time, with 8 webminer threads, 8 pages are being mined for data at the same time, versus 1 thread mining 1 page, while managing other tasks that in a multi-threaded program are just delegated to some other thread. Parallelization of the work load means multiple tasks can be worked on at the same time.

2.



3. The main way that I prevented threads from interfering with each other was primarily using synchronized methods and the Synchronized data types from the collections in java, the synchronized Map and List were fine for my simple implementation, I didn't really have to create my own data types. The synchronized methods are the only way for the threads to access the shared resources, they will only let one thread obtain the lock (java's Monitor lock) on them and then executed the method, while the others have to wait for the thread that is using the method is finished and releases the lock, then they can use it, like a queue to use some equipment. To help make sure only one thread can use a shared resource at a time, the data types themselves are synchronized. This help reinforce the synchronized nature of the shared resource. I feel this was adequate to deal with threads interfering with one another.
4. The WebManager using the ExecutorService to manage the WebMiner threads, when it tells the threads to stop either by shutdown or shutdown now, from what I've read and seen in the program, the executor will attempt to shut down all of the threads, by interrupting them, but it isn't guaranteed to end them immediately, as some way be waiting for synchronized methods or for other reason which is preventing them from immediately shutting down, while others will shut down immediately. I guess it is hit or miss, there might be another way to force them to halt execution, but I can't think of a way at the moment, other than using Stop.