AP Ex 2 Report

by Nigel Ng

2427257N UoG

# Status Report

Successful compilation using javac, java se 7 and above. It accepts the supplied input as specified in the assignment brief. It accepts a pattern followed by zero or more directories. If no directories are specified, then it takes the root folder where the solution file – fileCrawler.java – is in to search.

The implementation of the solution uses multiple workers in a producer/consumer system to traverse through the specified directory and find any files that matches the supplied pattern.

Testing the solution and the speeds of the producer and the consumers, the solution runs faster as more consumers threads are created but starts to degrade as the number of threads start to be more than the system can handle. Results will be posted later in the report.

Thread safe structures have been used.

# Bugs and Limitation

As no specific Java version has been mentioned in the assignment brief, Java SE 7 or higher has been used in my implementation of the solution. The linux VM from UoG used for testing also only has up to Java SE 7 installed.

By measuring the completion timings of the producer, consumer and main thread, it was found out that the usage of the File class from Java library JAVA.IO.FILE has a significant impact on read speeds and is constantly causing the producer to be a bottleneck. This leaves the consumers idle quite often while the producer spends some time processing the File class.

Instead I have switch to using the relatively new JAVA.NIO class mainly the DirectoryStream and Files classes. Using these classes provides a significant increase in producer and consumer performance. Thus, allowing the producers to work faster and multiple consumers to make an actual difference.

However, it should be noted and emphasized that JAVA.NIO class availability is on Java SE 7 and higher.

# Results

Testing Conditions:

SITVM will be used

1 Producer will be used throughout

1, 2, 8, 25, 100 consumers will be used to test the timings

As per the test command said to be used only 1 directory TestDIr will be used.

Will use search for pattern \*.c to standardize.

The Producer time and Consumer Time will be measured and a difference will be taken. Producer time starts when the producer thread is started and ends upon the producer finishing work production. Consumer start time will essentially be the same as the producer but ends when the solution has finished printing the harvest queue. Note that the consumers start just before the producer starts.

## SITVM

### 1 Consumer

Producer Time: 206ms, Consumer TIme: 311ms, Difference: 105ms

### 2 Consumers

Producer Time: 233ms, Consumer Time: 264ms, Difference: 31ms

### 8 Consumers

Producer Time: 238ms, Consumer Time: 278ms, Difference: 40ms

### 25 Consumers

Producer Time: 263ms, Consumer Time: 302ms, Difference: 39ms

### 100 Consumers

Producer Time: 276ms, Consumer Time: 324ms, Difference: 48ms

# Rubrics

## Workable Solution

Implemented Multiple Workers

## Correct Argument Processing

The pattern is correctly read.

## Correct Processing of CRAWLER\_THREADS

Using export CRAWLER\_THREADS=<number>, the number of consumer threads can be set for the solution to declare.

## For Successful Compilation

Program compiles with JAVAC

## For Successful Compilation With No Warnings

Used JAVAC with -Werror. No errors or warnings were produced.

## Reasonable, Concurrency-Safe Class Used For Work Queue

Using LinkedBlockingQueue which is thread-safe as specified in the JDK.

## Reasonable, Concurrency-Safe Class Used For Another Queue

Using LinkedBlockingQueue which is thread-safe as specified in the JDK.

## Effective Mechanism For Determination When No More Directories

The Main thread will join() the producer thread. Using the poison pill method, the producer will put() a FileObject with the safety number 0. However once it is done processing the directories, it will put a FIleObject with the poison pill number 1 to signal that there are no more directories.

## Efficient Mechanism For Determination When Worker Thread Has Finished

After the producer is done, the Main thread will join() the consumer threads. When a worker reads the poison pill, if it is not the last worker it will put another FileObject with the poison pill for the other Workers to read. Then break out of the loop and end the thread. Going back to the Main thread it will repeat the process with each other consumer thread.

## If It works Correctly With The Files In The Test Folder

Did a differential check with the provided search results. Different order of entries but same number of entries found.

## If It Works Correctly With The Files In An Unseen Folder Of Files

Folders and files are unseen.

## Runtime Performance With 1 Worker On Test Folder Is Similar To Single Threaded Implementation

Similar performance timing.

## Runtime Performance On Test Folder First Improves, Then Degrades As Number Of Threads Is Increased

Tested in the university’s provided Linux VM. Increasing the number of consumer threads increases the performance which then starts to plateau at about 8 workers. Performance really starts to degrade when an excessive amount of consumer threads are created, such as 50 or 100.