Matrix Multiplication

Nick Alexander
Division of Computer Science
Marshall University
Huntington, WV 25703
(Kat Richards)

I. Problem Statement

Given a matrix with M rows and K columns and another matrix with K rows and N columns, the program was expected to calculate the dot product of the two matrices. Each element of the result matrix was to be calculated inside a different thread. The purpose of the problem was to explore the creation, execution, and termination of threads inside a Java application.

II. Methodology

The major concept of the assignment was learning how to use Java's Thread API to create a thread for each calculation. As such, understanding how the threads should be generated was the first step in solving the problem. Using a simple for loop, the program generates *M x N* worker threads and stores them in a simple array. Using a for-each iterator over the worker threads array, the threads were joined which ensured all the calculation threads finished execution and terminated before the program continued. Joining the threads is a crucial step because if a particular thread has not finished executing, there is at least one element in the result matrix that does not yet exist. If a user attempts to access that particular element, the program will either crash or return a garbage value. Inside the worker threads, a simple for loop iterates over the currently row and column to calculate a single element for the result matrix.

III. Results

The program successfully calculated the dot product of the supplied test matrices. Furthermore, the program also successfully calculated the dot product of additional tests with varying rows and columns. Additionally, the program will accurately report errors in the matrix configuration (e.g. matrix A has X columns but matrix B does not have X rows). Upon analyzing the algorithm, there are some concerns of efficiency. The resources required to create and execute multiple threads produces an overhead as opposed to using a series of loops to calculate the dot product in the main thread. The threads are not extremely inefficient - there is just a small resource overhead.

IV. References

1. Java Thread API (http://docs.oracle.com/javase/7/docs/api/java/lang/Thread.html)
2. Galvin et al., *Operating System Concepts with Java*, 8th ed. 2010.