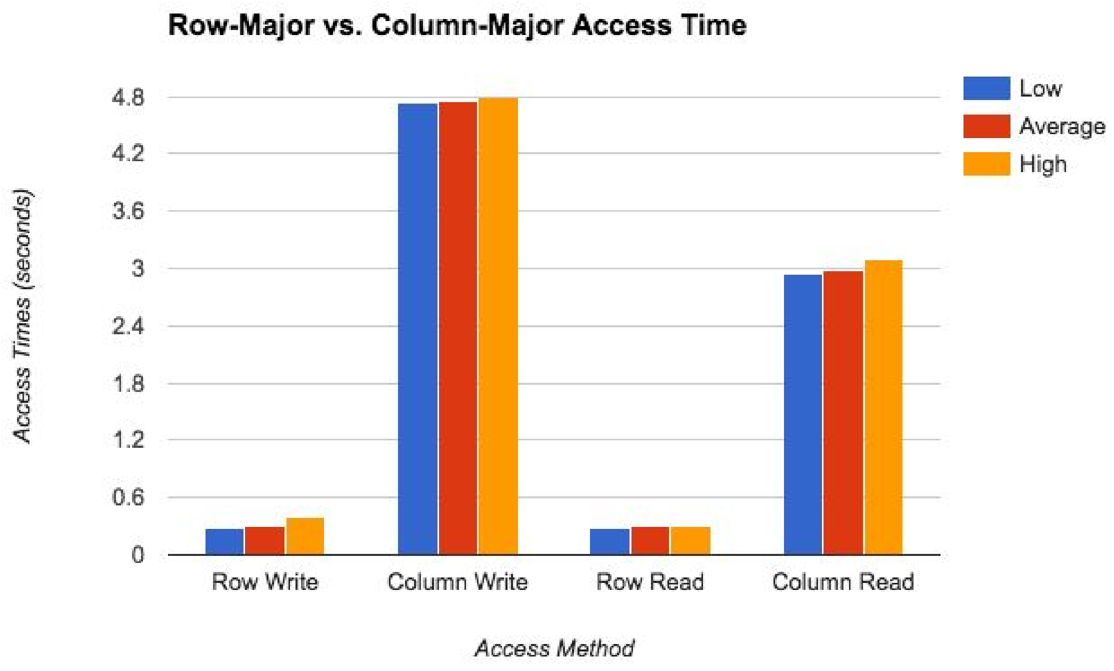
Luke Kledzik and Adam Mooers
Project 4 Runtime Experiment

Before Testing

Before testing this simulation, the `matrix.c` file was developed locally and then transferred to the `ssh.cs.uwf.edu` server. After compiling on the ssh server and fixing new warnings produced in the process, we could then run the program on the server and gather our runtime experiment data.

Testing Results

	Low	Average	High	Std. Dev.
Row Write	0.290	0.31	0.390	0.029
Column Write	4.730	4.755	4.790	0.016
Row Read	0.290	0.298	0.310	0.006
Column Read	2.950	2.978	3.100	0.045

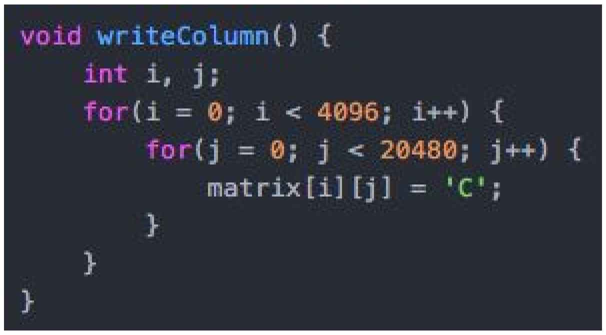


Interpretation of Results

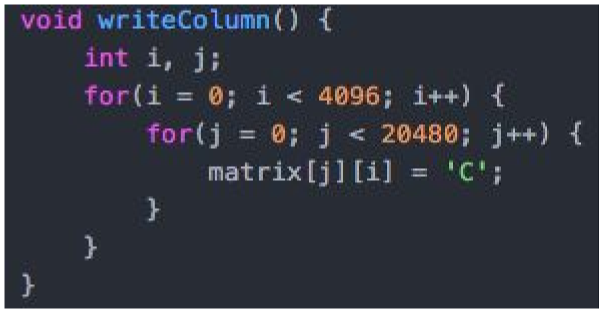
When running this test, it was quite surprising how slow the column-by-column access was compared to the row-by-row. After further consideration on the problem, it became apparent regarding what could be going on. When accessing the data row-by-row, the order of data being accessed is logically adjacent to one another, therefore offering quick access due to the data already being loaded into main memory. When accessing data column-by-column, due to the way 2-dimensional arrays are stored in memory, adjacent elements in the same column are actually separated by 4096 other values in the logical memory. This can cause a page fault due to the desired data not being currently loaded into main memory. The MMU catches this and the data then has to be retrieved. This is what is responsible for the significantly slower access speed. The program is constantly trying to access data that is not loaded into main memory and making up for this situation is extremely time costly.

Problems Testing

Initially when testing the simulation, we were getting similar times for row-by-row access and column-by-column access. After gaining suspicion, we analyzed our code and found the bug. The variables used to access the rows and columns in the global array were swapped. The first picture below shows the incorrect variable usage and the second picture shows the correct usage. (`matrix[i][j] → matrix[j][i]`)



```
void writeColumn() {
    int i, j;
    for(i = 0; i < 4096; i++) {
        for(j = 0; j < 20480; j++) {
            matrix[i][j] = 'C';
        }
    }
}
```



```
void writeColumn() {
    int i, j;
    for(i = 0; i < 4096; i++) {
        for(j = 0; j < 20480; j++) {
            matrix[j][i] = 'C';
        }
    }
}
```