Abstract

One dimensional arrays and two dimensional arrays will be created and have the addition operation performed upon them. This experiment seeks to discover the difference in performance of one-dimensional and two-dimensional arrays of the same size. At the end of our experiment we discover that multi-dimensional arrays take longer to perform the same task due to the for loops being taken into account when calculating the runtime.

Experiments

*Note: for easy comparison the same tests were done in languages: Python, Java, JavaScript, and C++

- Package used
    - `time` package used in order to measure how long the specific test took to run in nanoseconds
- One dimensional array addition
    - Array size: 29,997,529
    - First array
        - Contains only 5's
    - Second array
        - Contains only 2's
    - Resulting array
        - Contains only 7's since 5+2=7
- Multi-dimensional array addition
    - Array size: 5477 * 5477
        - Multidimensional array of 29,997,529
    - First array
        - Contains only 5's
    - Second array
        - Contains only 2's
    - Resulting array
        - Contains only 7's since 5+2=7

First two one-dimensional arrays are created with the size 29,997,529. One array will be filled with the integer five while the second array will be filled with the integer two. The two separate arrays will then perform the addition operation on each other resulting in an array full of the integer seven provided that everything went accordingly. The time package will be utilized in order to see how long it takes in order to add up the numbers from two arrays of the same dimensions.

In terms of the multi-dimensional array test, arrays of size 5477 by 5477 were created. The arrays filled respectively with five's and two's and performed the addition operation between the two arrays of the same dimension.

Both the one-dimensional and multi-dimensional array test was run 10 times and the average run times were taken and compared.

Results

```
Average of 10 runs for one-dimensional array: 6068685010.0 nanoseconds
Average of 10 runs for multi-dimensional array: 8080533330.0 nanoseconds
```

Although the arrays of the same dimension and data were added together the multi-dimensional array dimension took noticeably longer to perform the same task. In fact, a difference of 2,011,848,320 nanoseconds was found (see equation 1). A percent difference of approximately 33% (see equation 2) was found when comparing the multi-dimensional array runtime and one-dimensional array runtime of addition.

Equation 1: $8,080,533,330 - 6,068,686,010 = 2,011,848,320\ nanoseconds$

Equation 2: $\left|\frac{8,080,533,330-6,068,686,010}{6,068,686,010}\right| * 100 \approx 33.15\%$

Conclusion

Despite the arrays having the exact same size and data, the multi-dimensional array took longer to perform the same task. This is probably due to the implementation and where the time was started. Since python stores multi-dimensional arrays as a list within a list it should store the multidimensional array as two one dimensional arrays of size 5477. However due to where the start time started recording, the two for loops in order to access the elements in the multidimensional array is also taken into account for the calculation of the runtime. Based on the placement of the timestamps it is no wonder that the results reveal that multidimensional arrays take longer to run. This test should be revised and further researched as a possible future work topic.