Parameter Passing - Matrix Multiplication

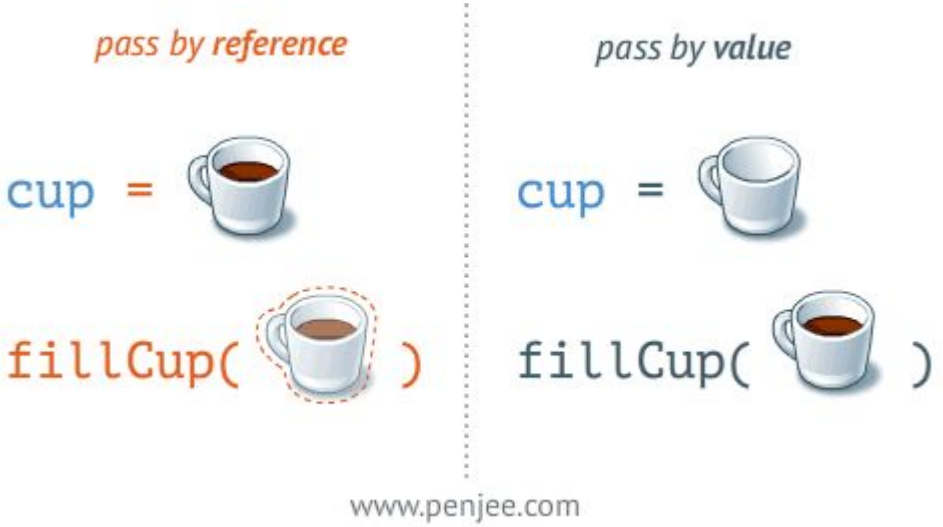
Gary Machorro, Lam Lieu, Juan Vera, Brandon Prak

Abstract

- Explore and compare different parameter passing methods in different languages
  - Java, Python and C++
- By writing a program that multiplies large matrices, we can:
  - compare several aspects of performance
  - compare language specific properties of parameter passing.

Introduction

- Two main passing methods
  - Pass By Value
  - Pass By Reference
- Java - pass by value
- Python - call by object
- C++ - pass by value and pass by reference



Methods/Algorithms/Concepts - Java

Pass By Value

```
private static void value(int[][] matrix1, int[][] matrix2, int[][] matrix3) {
  for (int i = 0; i < matrix1.length; i++) {
    for (int j = 0; j < matrix1.length; j++) {
      for (int k = 0; k < matrix1.length; k++) {
        matrix3[i][j] += matrix1[i][k] * matrix2[k][j];
      }
    }
  }
}
```

Methods/Algorithms/Concepts - Python

Pass By Object

```
#function that has an array as a parameter
def multArray(m1,m2,m3):

   # iterate through rows of m1
   for i in range(len(m1)):
      # iterate through columns of m2
      for j in range(len(m2[0])):
          # iterate through rows of m2
          for k in range(len(m2)):
              m3[i][j] += m1[i][k] * m2[k][j]
   return
```

Methods/Algorithms/Concepts - C++

Pass By Reference

```
void reference(struct Matrix &a, struct Matrix &b, struct Matrix &c) {
	//Performs matrix multiplication for matrix A and B
	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			for (int k = 0; k < size; k++) {
				c.arr[i][j] += a.arr[i][k] * b.arr[k][j];
			}
		}
	}
}
```

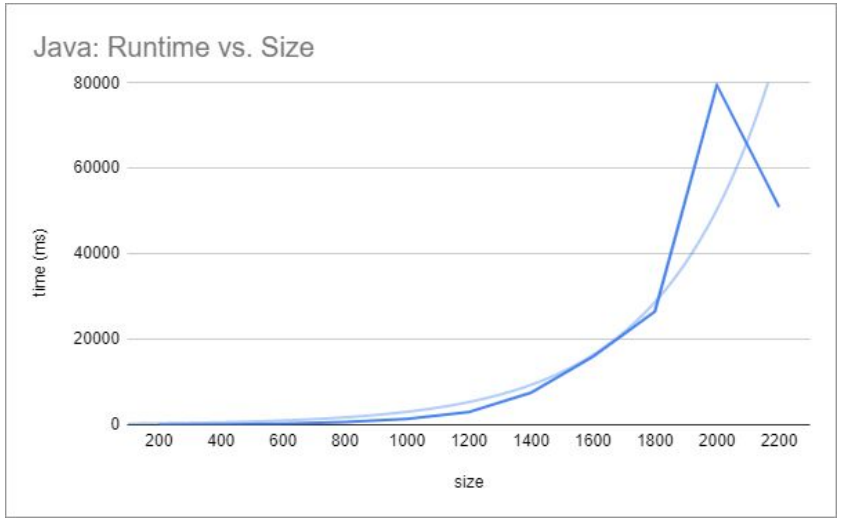
Pass by Value

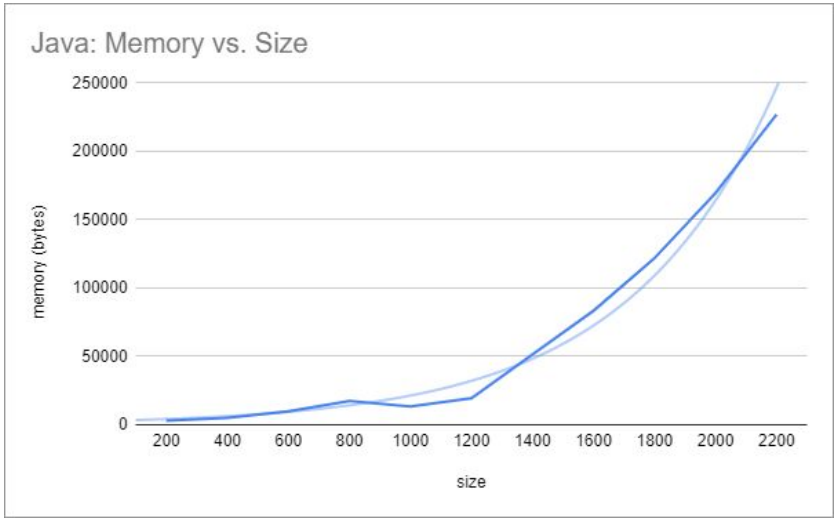
```
void value(struct Matrix a, struct Matrix b, struct Matrix c) {
	//Performs matrix multiplication for matrix A and B
	for (int i = 0; i < size; i++) {
		for (int j = 0; j < size; j++) {
			for (int k = 0; k < size; k++) {
				c.arr[i][j] += a.arr[i][k] * b.arr[k][j];
			}
		}
	}
}
```

Experiments

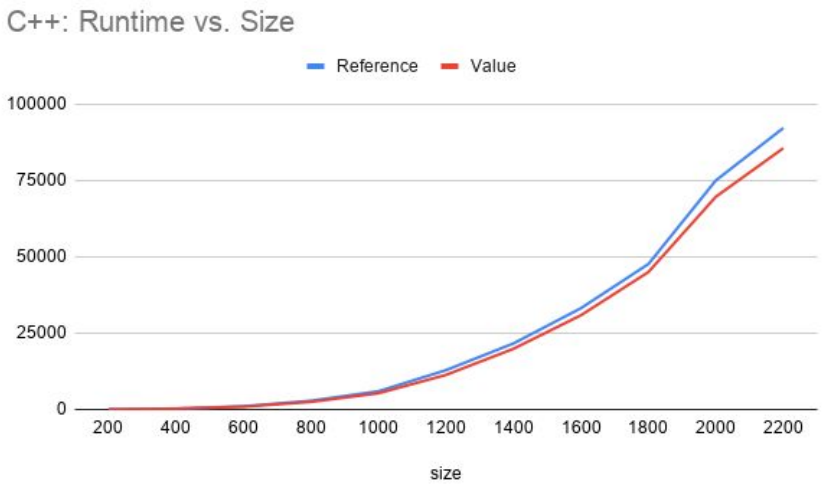
- Measure time and memory spent running algorithm
  - Starting at size 200, incrementing by 200 each iteration
  - Disregard initializations and setups for each iteration
- Create comparison graphs
  - Find averages via multiple samples
  - Time vs. Size, Memory vs. Size

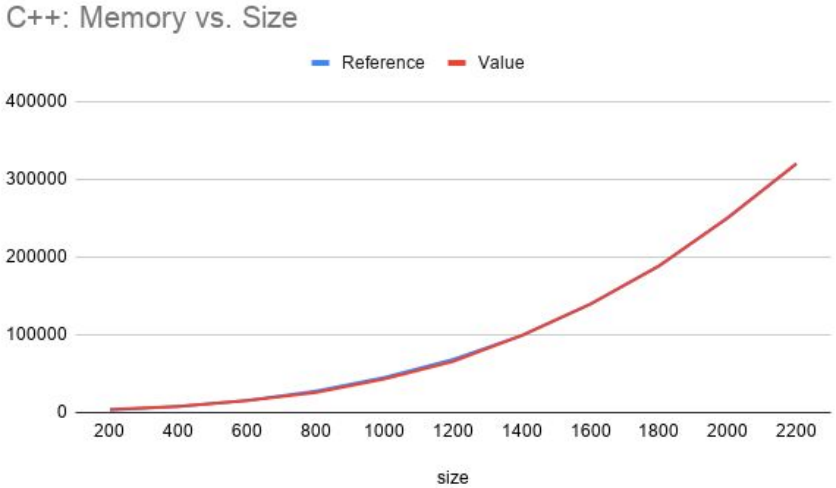
Results - Java



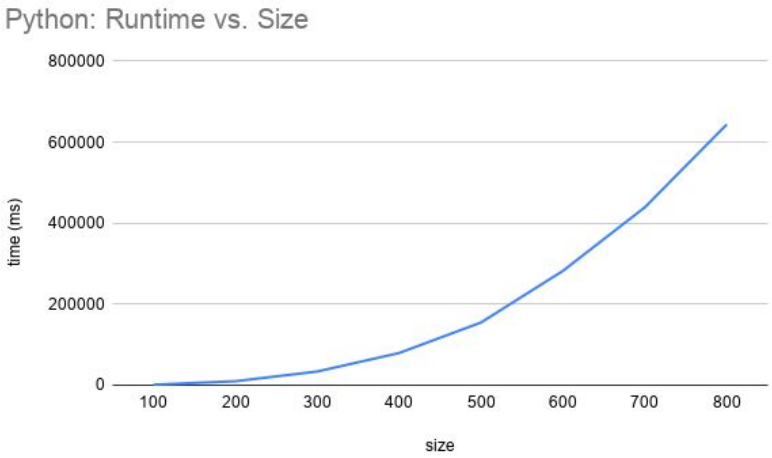


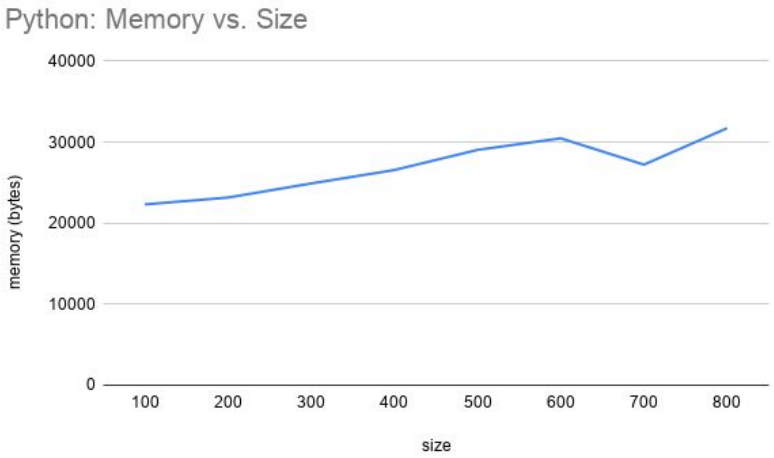
Results - C++


C++: Runtime vs. Size
Reference
Value
100000
75000
50000
25000
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
size

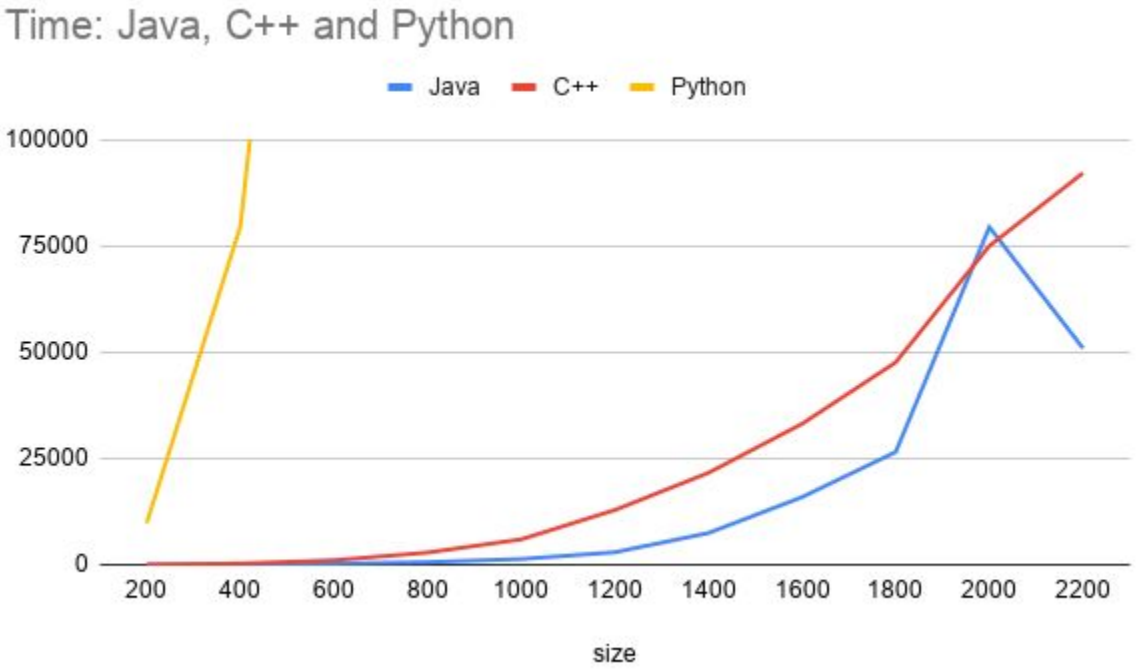

C++: Memory vs. Size
Reference
Value
400000
300000
200000
100000
0
200
400
600
800
1000
1200
1400
1600
1800
2000
2200
size

Results - Python

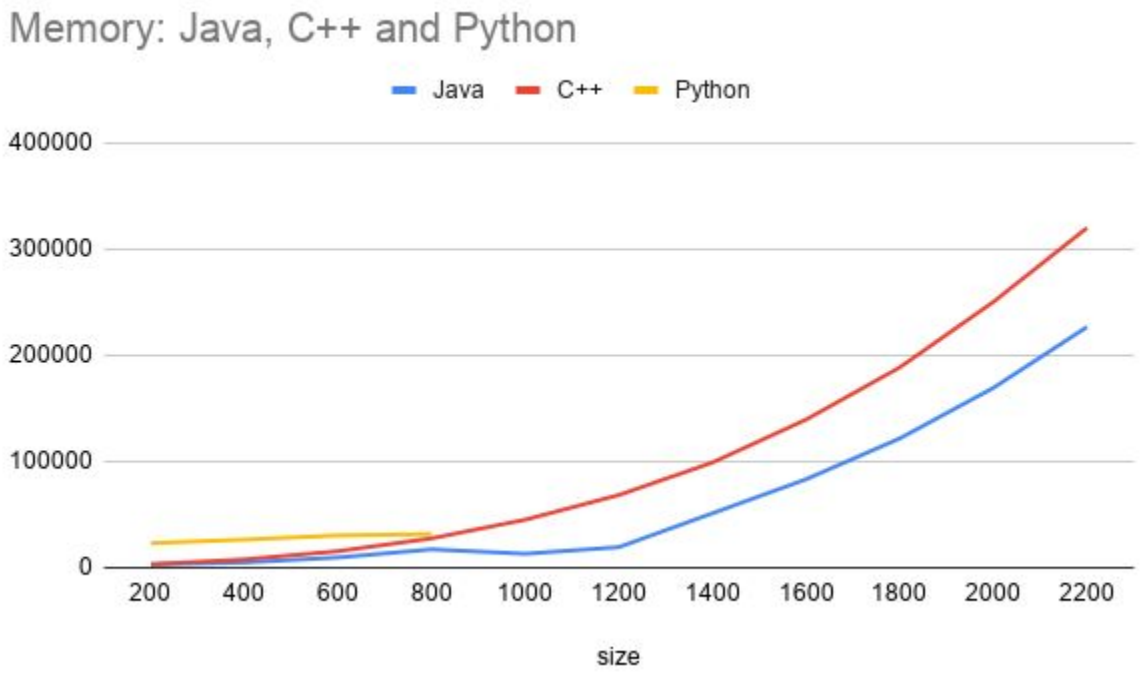




Results Comparison



Results Comparison (cont'd)



Future Work

- Improve Precision
  - More samples for average
  - Better methods to get memory usage
- Test on multiple desktops
  - More accurate averages
- Figure out spikes in Java data

Conclusion

- Passing Parameters methods have little to no effect on memory/time when performing matrix multiplications
    - Passing by value preferred so that data is more secure
- Java’s time/memory is most likely to fluctuate
    - Garbage Collector
- Python cannot handle large matrices multiplication
- C++ has steadiest growth and data
    - Difficult to implement C++ Matrix Multiplication via pass by value