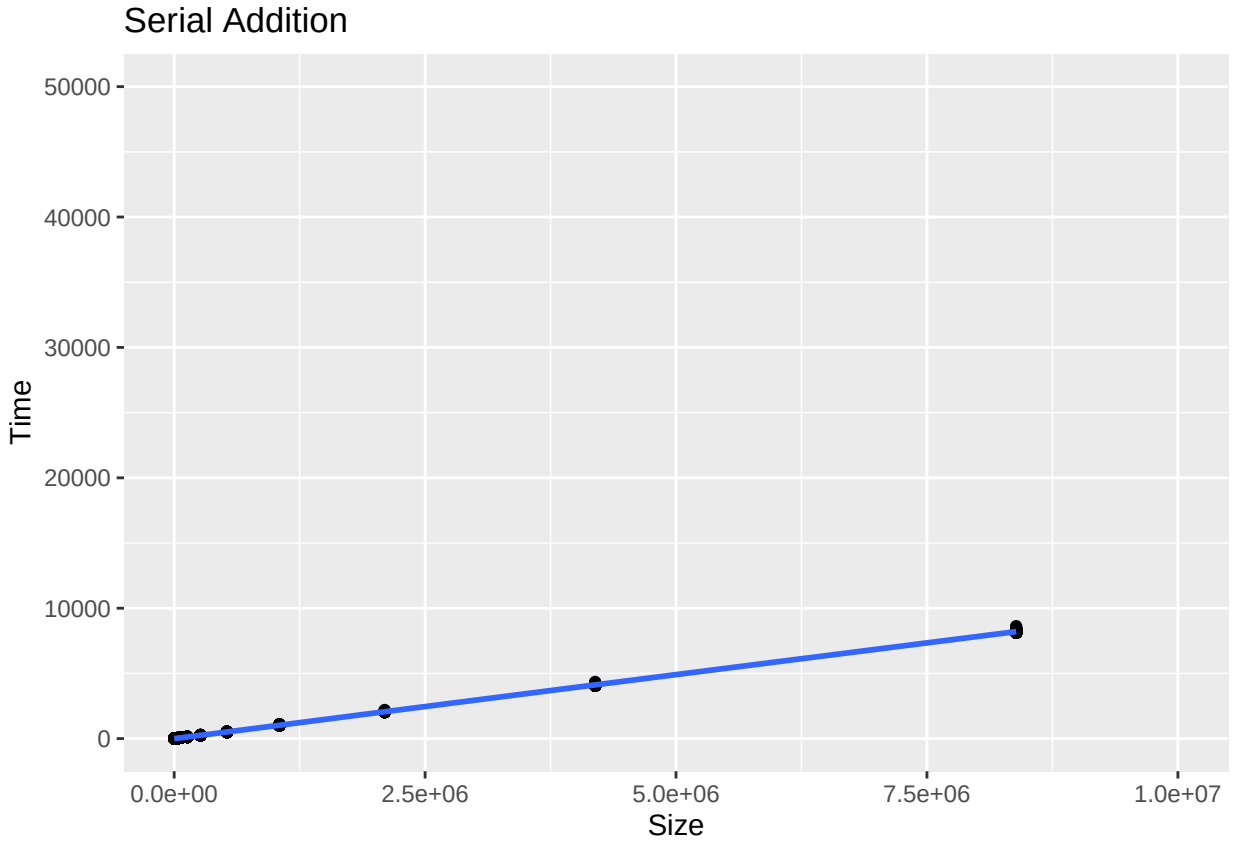
Final Project Analysis

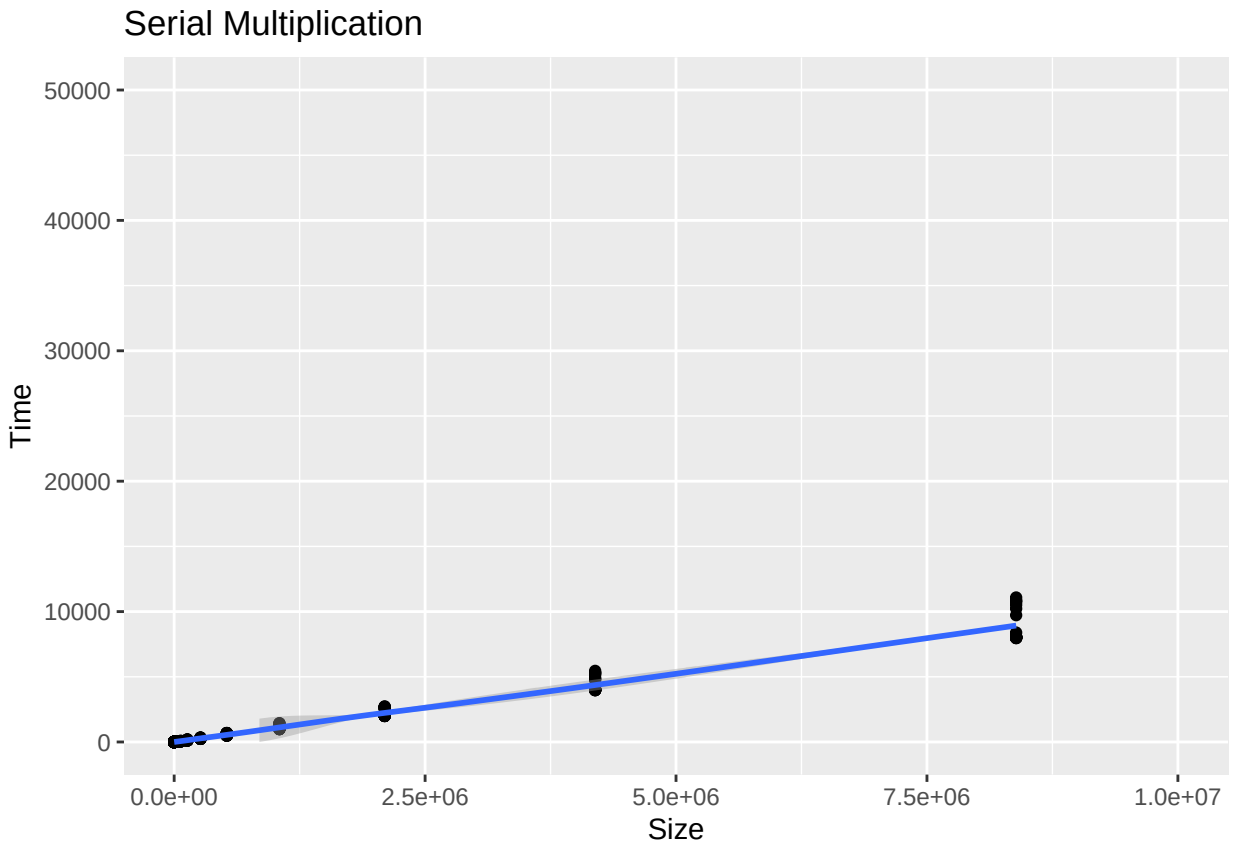
Yousuf Kanan and Derek Allmon

Serial

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'

## Warning: Removed 1765 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 1765 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



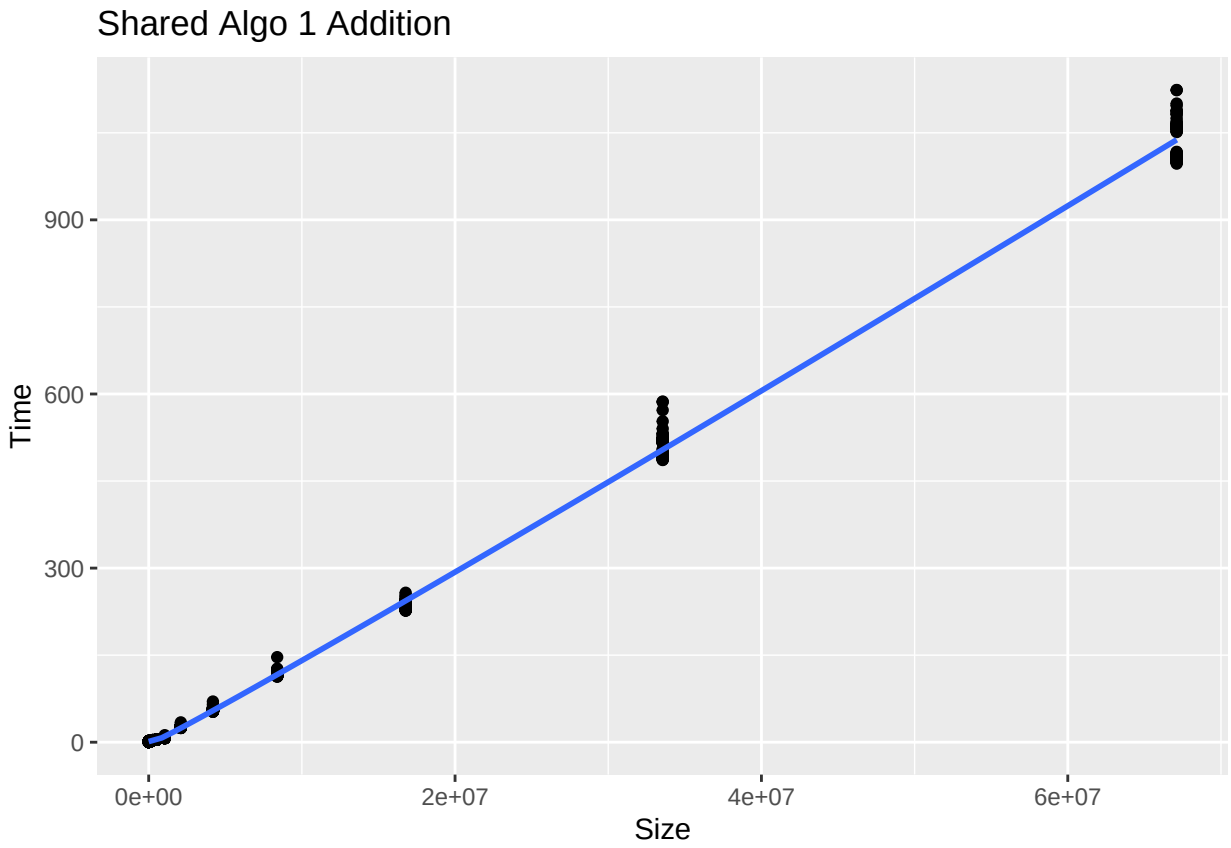

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'

## Warning: Removed 214 rows containing non-finite outside the scale range
## (`stat_smooth()`).

## Warning: Removed 214 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



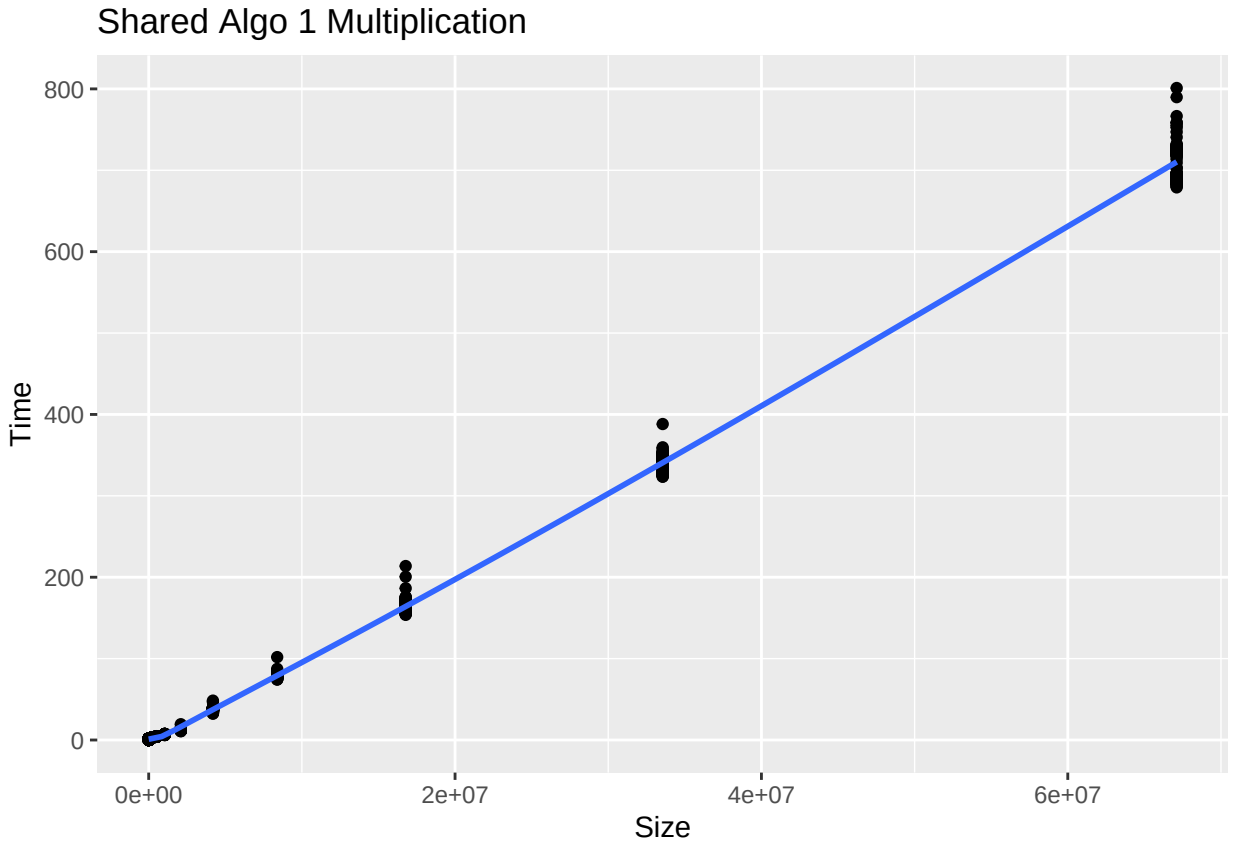
The graphs Serial Addition and Serial Multiplication show a linear relationship between the size and time. The slope of the line is steeper for the multiplication graph than the addition graph. This is because multiplication is more computationally expensive than addition.

Shared

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

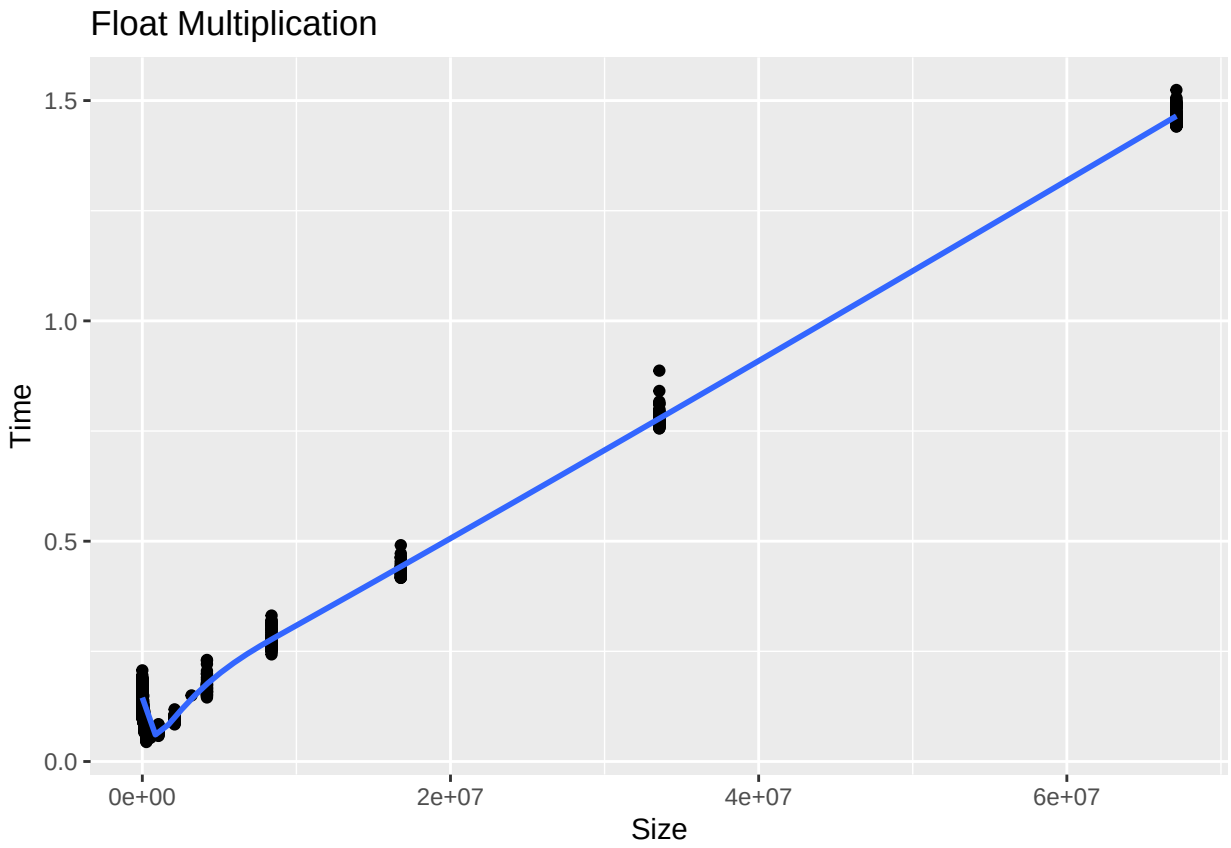


The graphs Shared Algo 1 Addition and Shared Algo 1 Multiplication show a linear relationship between the size and time. The slope for shared is steeper than that of sloped because the algorithm being used for shared is not work efficient and gets even slower as the size increases.

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 213 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

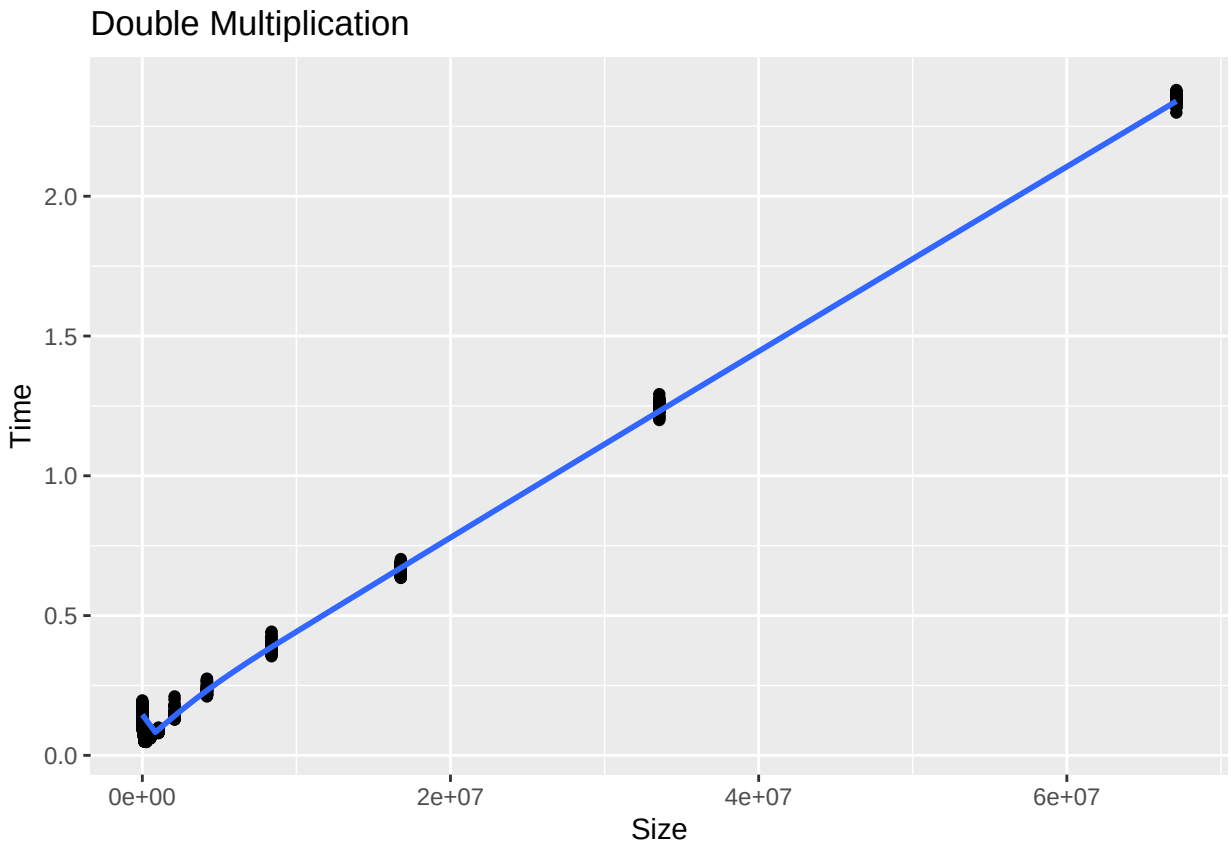
```
## Warning: Removed 213 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

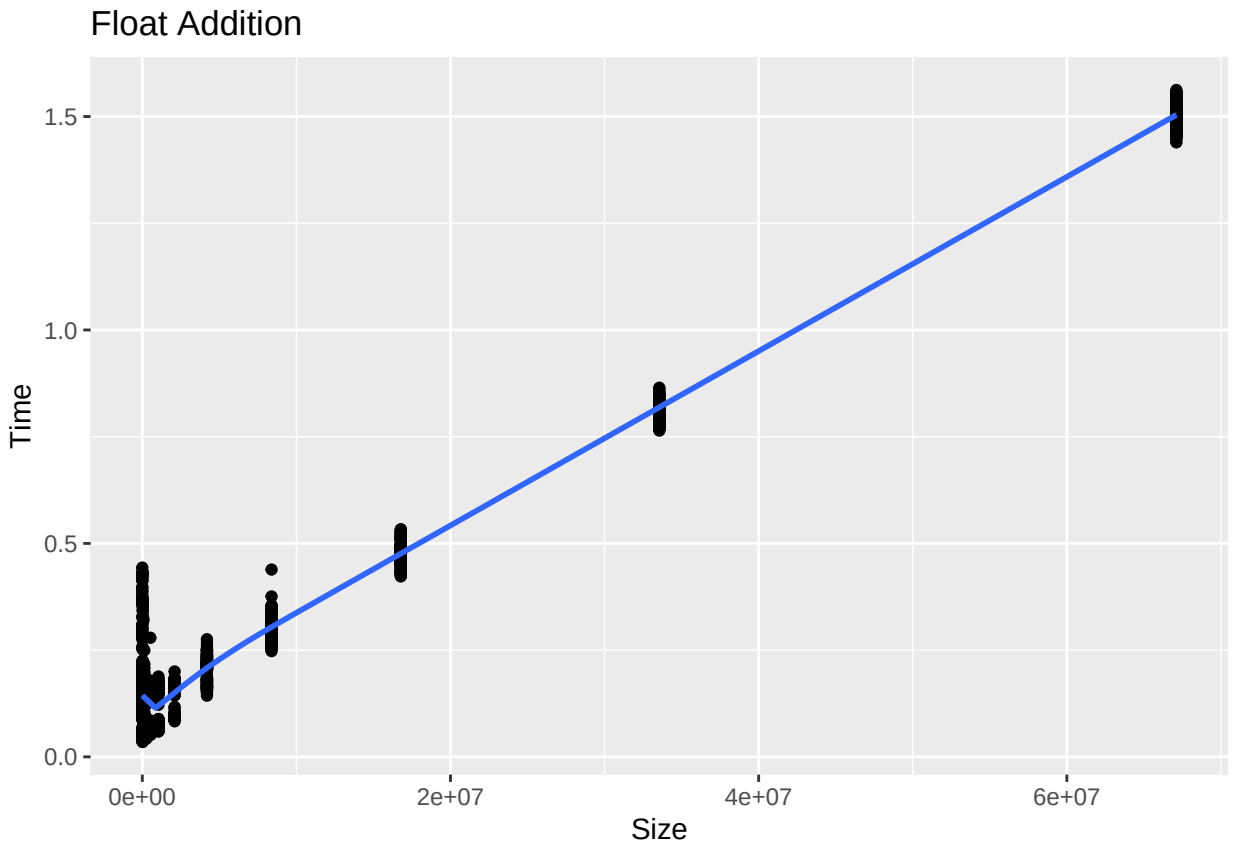
```
## Warning: Removed 156 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 156 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The two graphs Double Multiplication and Float Multiplication show a linear relationship between the size and time. As expected the float multiplication is faster than the double multiplication because the float multiplication is more efficient on a GPU.

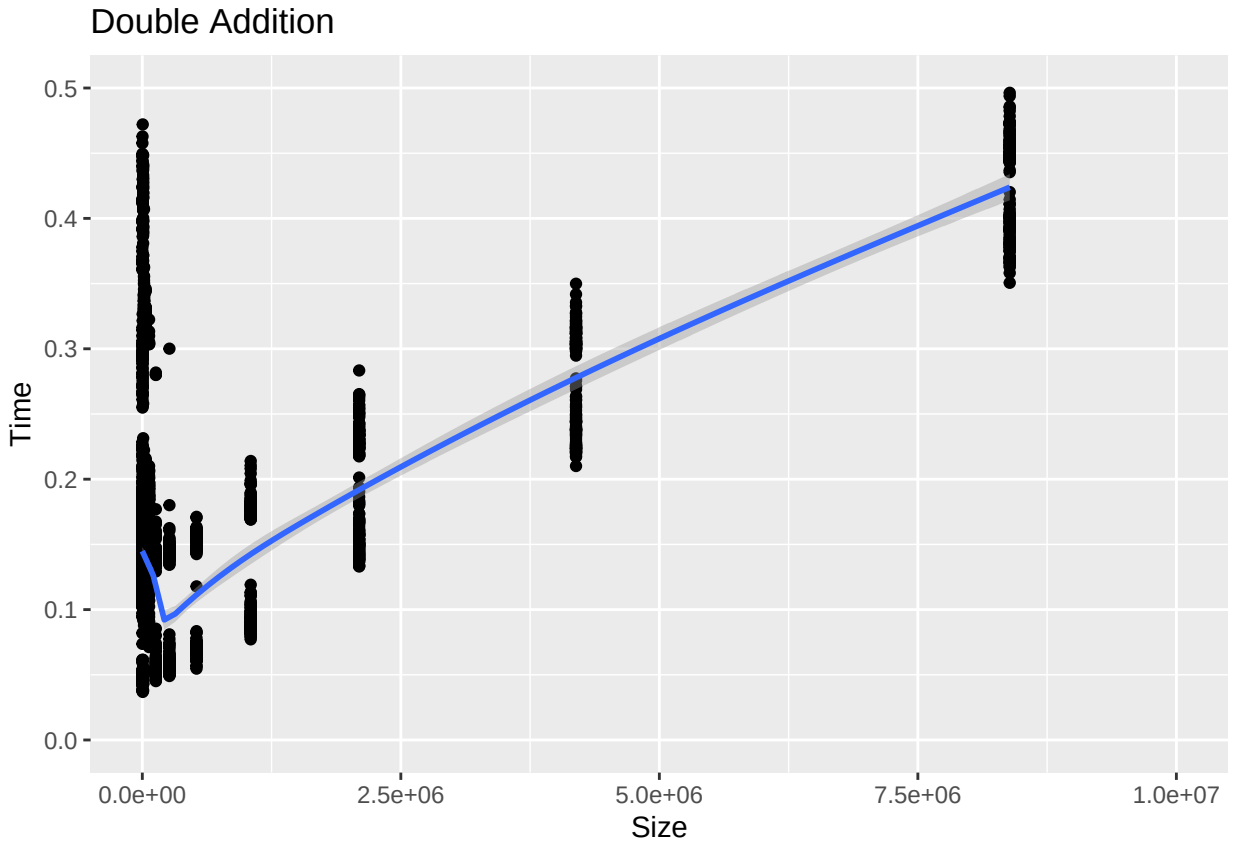
```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```




```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 451 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

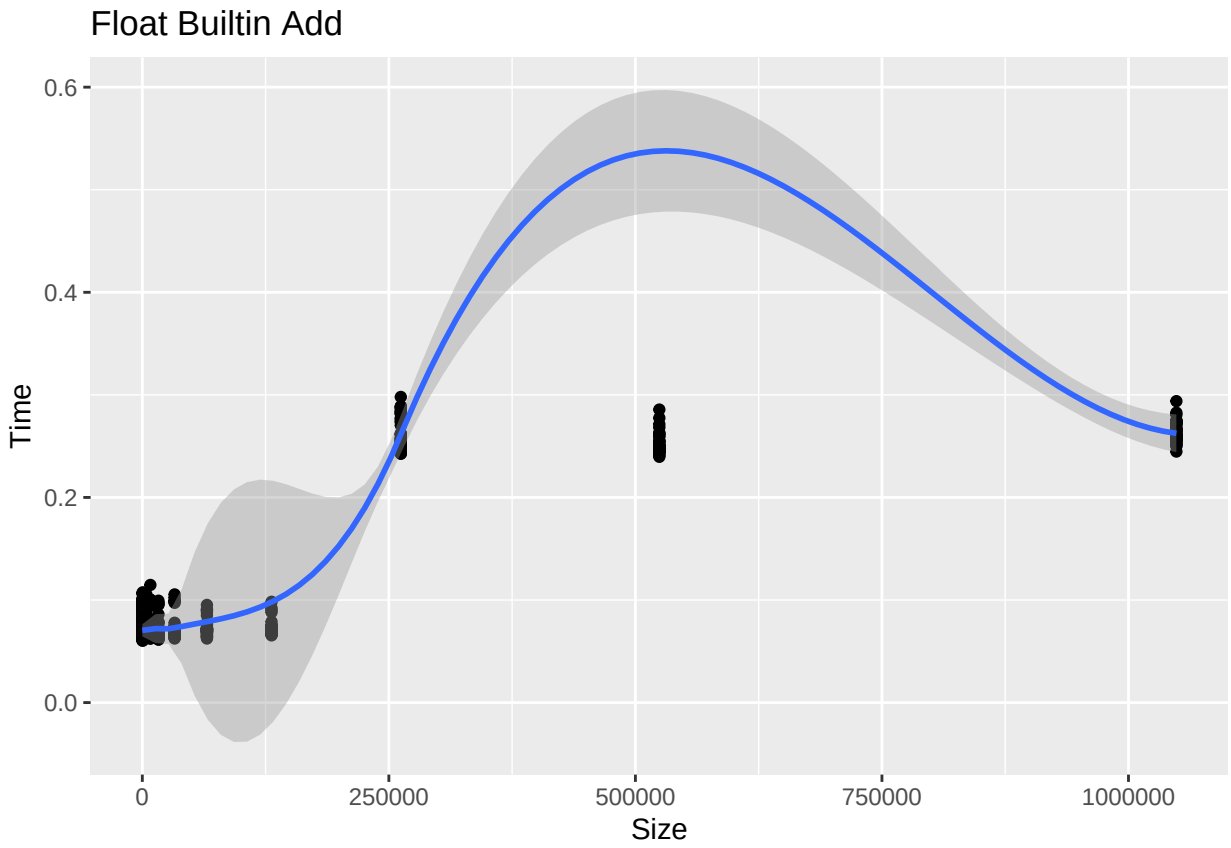
```
## Warning: Removed 451 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



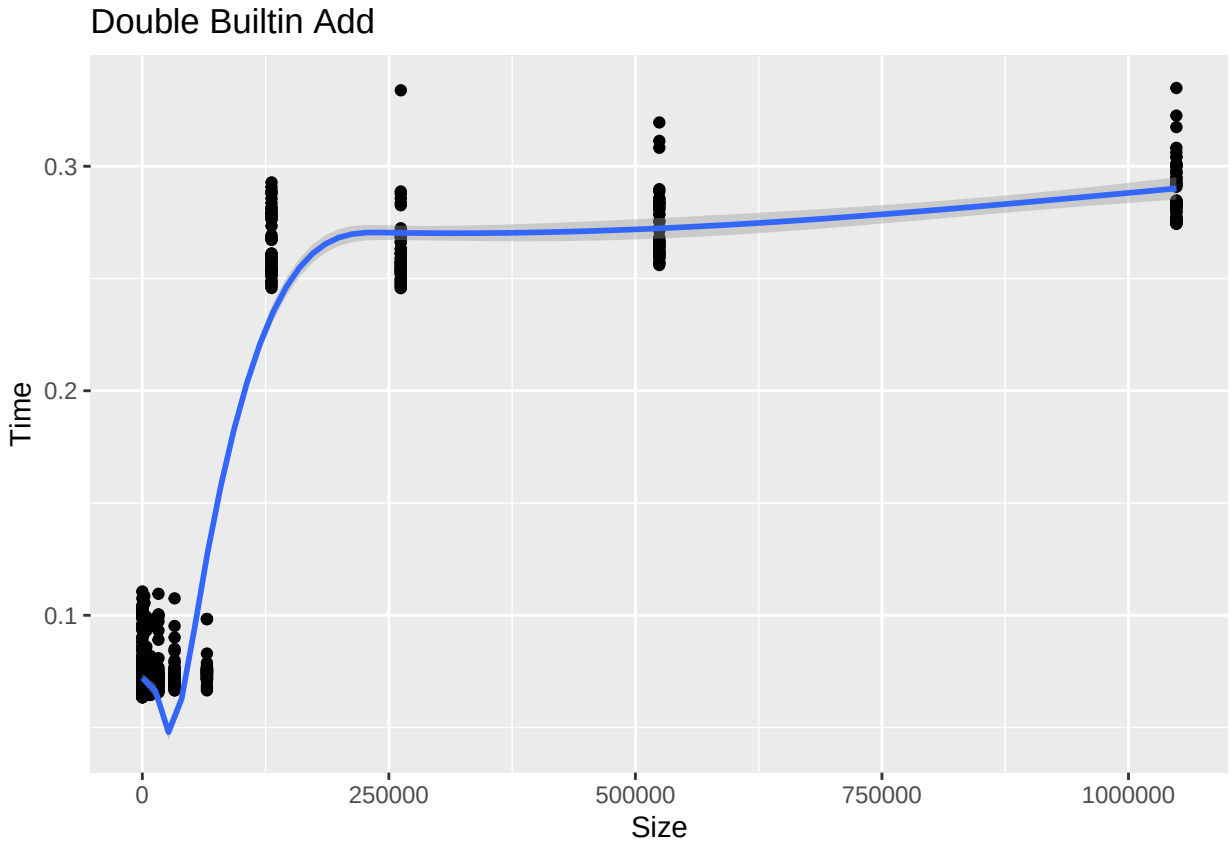
The two graphs Double Addition and Float Addition show a linear relationship between the size and time. As expected the float addition is faster than the double addition because the float addition is more efficient on a GPU.

Although the float addition and multiplication are faster than the double addition and multiplication, the difference is not as significant as we expected.

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

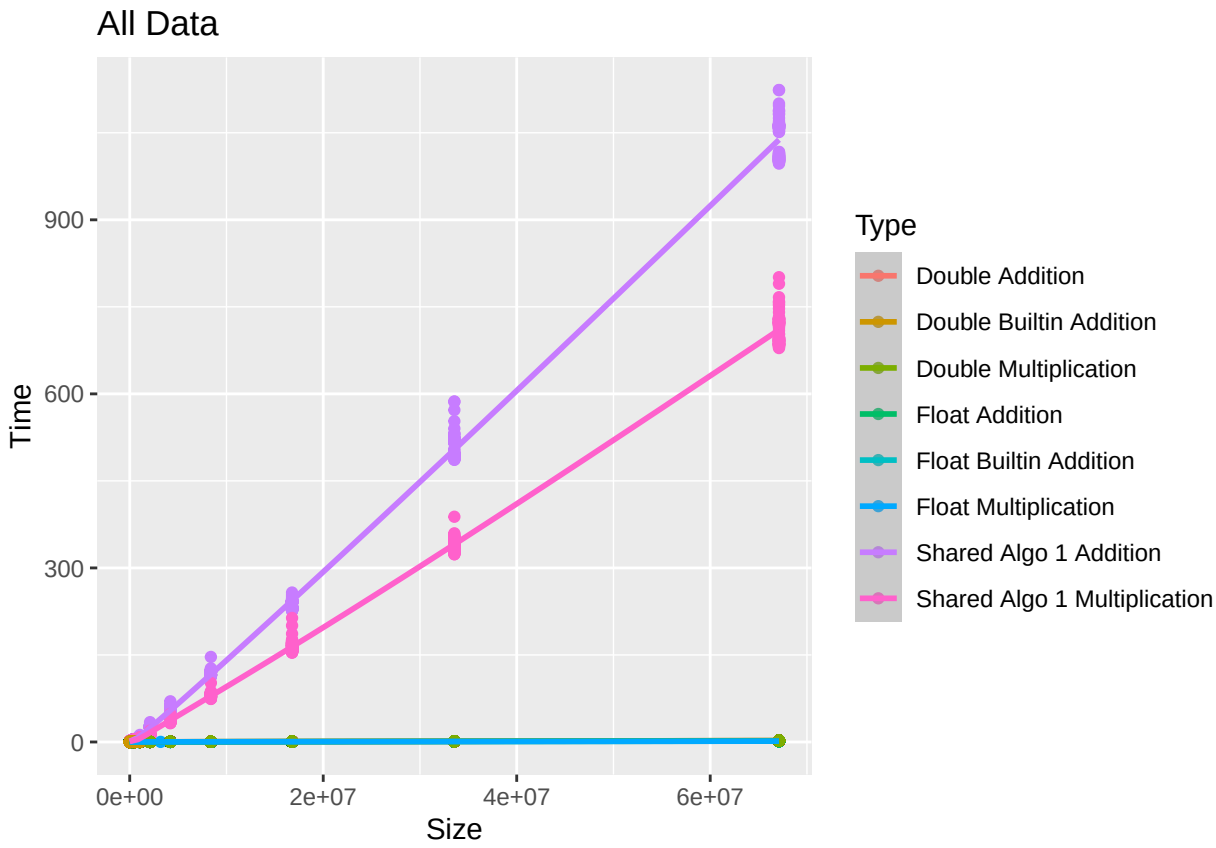


The graphs Double Built in Addition and Float Built in Addition show a logarithmic relationship between the size and time. The built-in addition is faster than the custom addition because the built-in addition is more efficient because it has had years of optimization. Another potential reason for the difference in time is that the built-in addition was tested on different data than the custom addition.

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

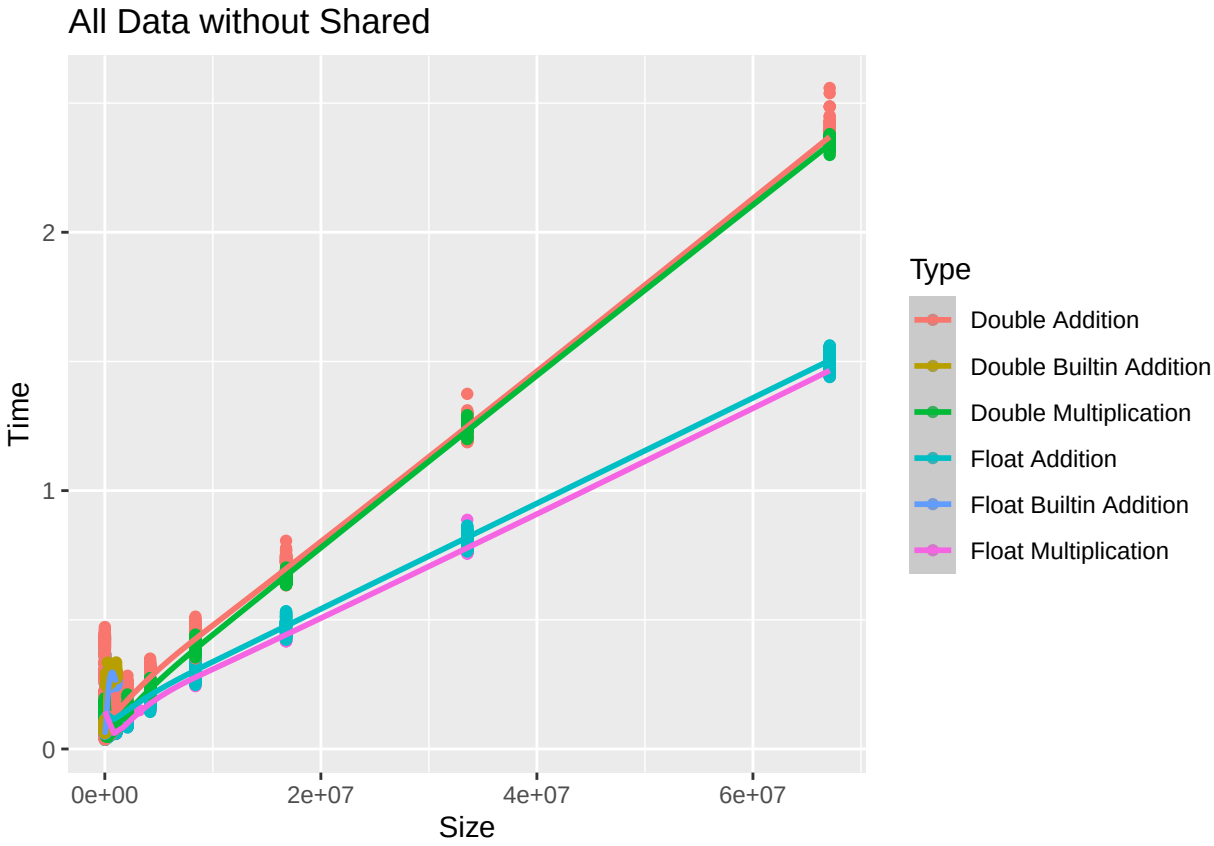
```
## Warning: Removed 370 rows containing non-finite outside the scale range
## (`stat_smooth()`).
```

```
## Warning: Removed 370 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

```
## Warning: Removed 370 rows containing non-finite outside the scale range
## (`stat_smooth()`).
## Removed 370 rows containing missing values or values outside the scale range
## (`geom_point()`).
```



The first graph shows all the data and the second graph shows all the data without the shared data. The graphs show that the shared data is not as efficient as the other data. The shared data is not as efficient as the other data because the shared data is not work efficient and gets even slower as the size increases.

Lookng at the data we can say that shared has negative parallelism and the cuda portion is the most efficient.

It is importnt to ote that for the algorithms used above the number of threads is set to 8. 8 is the maximum number of threads on the mucluster and the number of threads is set to 8 to take most advatage of our algorithm.