Experiments:

Changing sparsity and matrix size

Using multithreading:

Matrix Size	Sparsity	Thread count	Time Elapsed (s)
1000 x 1000	90%	4	0.00659879
1000 x 1000	50%	4	0.00466412
1000 x 1000	10%	4	0.00879442
2000 x 2000	90%	4	0.0562355
2000 x 2000	50%	4	0.0411221
2000 x 2000	10%	4	0.0465174
5000 x 5000	90%	4	0.397277
5000 x 5000	50%	4	0.393068
5000 x 5000	10%	4	0.380396

Using SIMD

Matrix Size	Sparsity	Thread count	Time Elapsed (s)
1000 x 1000	90%	4	3.36808
1000 x 1000	50%	4	3.79081
1000 x 1000	10%	4	3.43697
2000 x 2000	90%	4	511%813
2000 x 2000	50%	4	51.2454
2000 x 2000	0%	4	55.4488
5000 x 5000	90%	4	1150.48
5000 x 5000	50%	4	1149.81
5000 x 5000	10%	4	1148.68

Using Optimization

Matrix Size	Sparsity	Thread count	Time Elapsed (s)
1000 x 1000	90%	4	3.43727
1000 x 1000	50%	4	3.505595
1000 x 1000	10%	4	3.47907
2000 x 2000	90%	4	83.2711
2000 x 2000	50%	4	85.9108
2000 x 2000	0%	4	83.7722
5000 x 5000	90%	4	2086.50
5000 x 5000	50%	4	2083.50
5000 x 5000	10%	4	2088.50

In general, as matrix size increases the time needed to complete the multiplication increases.
As sparsity increases, time is somewhat inconsistent.

Vary matrix size:

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	90%	0.00659879
Multithreading	2000 x 2000	90%	0.0562355
Multithreading	5000 x 5000	90%	0.397277
SIMD (Neon)	1000 x 1000	90%	3.36808
SIMD (Neon)	2000 x 2000	90%	511.813
SIMD (Neon)	5000 x 5000	90%	1150.48
Cache optimization	1000 x 1000	90%	3.43727
Cache optimization	2000 x 2000	90%	83.2711
Cache optimization	5000 x 5000	90%	208650.5

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	50%	0.00466412
Multithreading	2000 x 2000	50%	0.0411221
Multithreading	5000 x 5000	50%	0.393068
SIMD (Neon)	1000 x 1000	50%	3.79081
SIMD (Neon)	2000 x 2000	50%	51.2454
SIMD (Neon)	5000 x 5000	50%	1149.81
Cache optimization	1000 x 1000	50%	350.5595
Cache optimization	2000 x 2000	50%	85.9108
Cache optimization	5000 x 5000	50%	2083.508

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	10%	0.00879442
Multithreading	2000 x 2000	10%	0.0465174
Multithreading	5000 x 5000	10%	0.380396
SIMD (Neon)	1000 x 1000	10%	3.43697
SIMD (Neon)	2000 x 2000	10%	55.4488
SIMD (Neon)	5000 x 5000	10%	1148.68
Cache optimization	1000 x 1000	10%	3.47907
Cache optimization	2000 x 2000	10%	83.7722
Cache optimization	5000 x 5000	10%	2088.501

As the matrix size increases, the time needed to multiply the matrices increase for all methods used.
Cache optimization has the greatest difference in time as matrix size increases. This is because of how modern memory hierarchies handle data access - there is a great difference between the speed at which memory can be written vs how fast memory can be accessed.

Vary sparsity:

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	90%	0.00659879
Multithreading	1000 x 1000	50%	0.00466412
Multithreading	1000 x 1000	10%	0.00879442
SIMD (Neon)	1000 x 1000	90%	3.36808
SIMD (Neon)	1000 x 1000	50%	3.79081
SIMD (Neon)	1000 x 1000	10%	3.43697
Cache optimization	1000 x 1000	90%	3.43727
Cache optimization	1000 x 1000	50%	350.015595
Cache optimization	1000 x 1000	10%	3.47907

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	2000 x 2000	90%	0.0562355
Multithreading	2000 x 2000	50%	0.0411221
Multithreading	2000 x 2000	10%	0.0465174
SIMD (Neon)	2000 x 2000	90%	51.1813
SIMD (Neon)	2000 x 2000	50%	51.2454
SIMD (Neon)	2000 x 2000	10%	55.4488
Cache optimization	2000 x 2000	90%	83.2711
Cache optimization	2000 x 2000	50%	85.9108
Cache optimization	2000 x 2000	10%	83.7722

Method used	Matrix Size	Sparsity	Time Elapsed (s)
Multithreading	5000 x 5000	90%	0.397277
Multithreading	5000 x 5000	50%	0.393068
Multithreading	5000 x 5000	10%	0.380396
SIMD (Neon)	5000 x 5000	90%	1150.48
SIMD (Neon)	5000 x 5000	50%	1149.81
SIMD (Neon)	5000 x 5000	10%	1148.68
Cache optimization	5000 x 5000	90%	208650.01
Cache optimization	5000 x 5000	50%	208350
Cache optimization	5000 x 5000	10%	208850

As sparsity gets closer to 100%, there are more 0s in the matrix.
As there are more 0s in the matrix, the time should generally gets faster. However, during these sets of experiimets, the time was the fastest around sparsity 50%
More sparse matrices are generally faster for matrix multiplication because there are less operations to be done

Dense x sparse: (sort by sparsity)

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	90%	1000 x 1000	10%	0.00505158
Multithreading	2000 x 2000	90%	2000 x 2000	10%	0.0402326
Multithreading	5000 x 5000	90%	5000 x 5000	10%	0.421675
SIMD (Neon)	1000 x 1000	90%	1000 x 1000	10%	3.88991
SIMD (Neon)	2000 x 2000	90%	2000 x 2000	10%	48.4245
SIMD (Neon)	5000 x 5000	90%	5000 x 5000	10%	
Cache optimization	1000 x 1000	90%	1000 x 1000	10%	4.03076
Cache optimization	2000 x 2000	90%	2000 x 2000	10%	79.7586
Cache optimization	5000 x 5000	90%	5000 x 5000	10%	

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	95%	1000 x 1000	5%	0.0109655
Multithreading	2000 x 2000	95%	2000 x 2000	5%	0.0691733
Multithreading	5000 x 5000	95%	5000 x 5000	5%	0.618273
SIMD (Neon)	1000 x 1000	95%	1000 x 1000	5%	4.46767
SIMD (Neon)	2000 x 2000	95%	2000 x 2000	5%	60.823
SIMD (Neon)	5000 x 5000	95%	5000 x 5000	5%	
Cache optimization	1000 x 1000	95%	1000 x 1000	5%	4.67814
Cache optimization	2000 x 2000	95%	2000 x 2000	5%	91.0695
Cache optimization	5000 x 5000	95%	5000 x 5000	5%	

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	99%	1000 x 1000	1%	0.00741558
Multithreading	2000 x 2000	99%	2000 x 2000	1%	0.0530113
Multithreading	5000 x 5000	99%	5000 x 5000	1%	0.631483
SIMD (Neon)	1000 x 1000	99%	1000 x 1000	1%	4.8207
SIMD (Neon)	2000 x 2000	99%	2000 x 2000	1%	57.4393
SIMD (Neon)	5000 x 5000	99%	5000 x 5000	1%	1207.36
Cache optimization	1000 x 1000	99%	1000 x 1000	1%	6.87271
Cache optimization	2000 x 2000	99%	2000 x 2000	1%	83.5883
Cache optimization	5000 x 5000	99%	5000 x 5000	1%	

As sparsity increases in matrix A and decreases in matrix B the time needed to run the multiplication gets shorter. This goes against what should theoretically happen because as matrix get more sparse, less operations are needed.

Dense x sparse: (sort by size)

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	1000 x 1000	90%	1000 x 1000	10%	0.00505158
Multithreading	1000 x 1000	95%	1000 x 1000	5%	0.0109655
Multithreading	1000 x 1000	99%	1000 x 1000	1%	0.00741558
SIMD (Neon)	1000 x 1000	90%	1000 x 1000	10%	3.88991
SIMD (Neon)	1000 x 1000	95%	1000 x 1000	5%	4.46767
SIMD (Neon)	1000 x 1000	99%	1000 x 1000	1%	4.8207
Cache optimization	1000 x 1000	90%	1000 x 1000	10%	4.03076
Cache optimization	1000 x 1000	95%	1000 x 1000	5%	4.67814
Cache optimization	1000 x 1000	99%	1000 x 1000	1%	6.87271

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	2000 x 2000	90%	2000 x 2000	10%	0.0402326
Multithreading	2000 x 2000	95%	2000 x 2000	5%	0.0691733
Multithreading	2000 x 2000	99%	2000 x 2000	1%	0.0530113
SIMD (Neon)	2000 x 2000	90%	2000 x 2000	10%	48.4245
SIMD (Neon)	2000 x 2000	95%	2000 x 2000	5%	60.823
SIMD (Neon)	2000 x 2000	99%	2000 x 2000	1%	57.4393
Cache optimization	2000 x 2000	90%	2000 x 2000	10%	79.7586
Cache optimization	2000 x 2000	95%	2000 x 2000	5%	91.0695
Cache optimization	2000 x 2000	99%	2000 x 2000	1%	83.5883

Method used	Matrix A Size	Sparsity	Matrix B Size	Sparsity	Time Elapsed (s)
Multithreading	5000 x 5000	90%	5000 x 5000	10%	0.421675
Multithreading	5000 x 5000	95%	5000 x 5000	5%	0.618273
Multithreading	5000 x 5000	99%	5000 x 5000	1%	0.631483
SIMD (Neon)	5000 x 5000	90%	5000 x 5000	10%	
SIMD (Neon)	5000 x 5000	95%	5000 x 5000	5%	
SIMD (Neon)	5000 x 5000	99%	5000 x 5000	1%	1207.36
Cache optimization	5000 x 5000	90%	5000 x 5000	10%	
Cache optimization	5000 x 5000	95%	5000 x 5000	5%	
Cache optimization	5000 x 5000	99%	5000 x 5000	1%	

For dense x sparse multiplication, the time increases as the matrices get bigger.

Final Analysis

In general most of the results i got lined up with what was expected. The only thing that was different (that i can't figure out why) is when sparcity changed, it seemed to take the least amount of time when the sparsity was at 50%