The below screenshot gives the command line arguments that are required to be given to run benchmark successfully.

```
cc@sg-instance-hw4:~/home/cs553-spring2024-hw4-SonaGutha$ ./hashgen -h
Help:
 -t <hash_threads>: Specify the number of hash threads
 -o <sort_threads>: Specify the number of sort threads
 -i <write_threads>: Specify the number of write threads
 -f <filename>: Specify the filename
 -m <MAX_MEMORY_MB> : Maximum memory allowed to use
 -s <FILE_SIZE_GB> : File size in GB
 -d <bool>: turns on debug mode with true, off with false
 -h: Display this help message
```

Sample Command to run the experiments :

Ex: ./hashgen -t 1 -o 1 -i 16 -f data.bin -m 128 -s 1 -d true

These experiments are ran on WSL on visual studio code.

1GB Workload:

1. Maximum memory allowed to use (MB): 128

2. Number of hash threads: 1, 4, 16

3. Number of sort threads: 1, 4, 16

4. Number of write threads: 1, 4, 16

After running the 27 experiments, the binary files are verified using hashVerify.

Committed the log file to the git repo, Screenshots for some of the experiments performing hash verifications as follow:

```
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_1_16.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_4_1.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_4_4.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_4_16.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_16_1.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_16_4.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_16_16.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_16_16_16.bin -v true
Read 1073741824 bytes and found all records are sorted.
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_16_4_16.bin -v true
Read 1073741824 bytes and found all records are sorted.
```

To double check the hashes generated, ran the hashverify to print and check is the hashes generated are valid for some of the experiments

```
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_1_1.bin -p 10
Printing first 10 of file 'data_1_1_1.bin'...
[0] Hash: 0000001884fc69b45daf : f0e497c2b06f : 122804969661680
[16] Hash: 0000001b515979e90865 : 580817defd36 : 60463980677208
[32] Hash: 000000496ce8b21c4dc6 : d878fd16355f : 104681623615704
[48] Hash: 000000d58be48e3cd534 : 1339ec453ed0 : 228965879658771
[64] Hash: 00000108bf306de5d665 : 3372ac81c67f : 140490555814451
[80] Hash: 0000013b5a8e3db68de6 : 8a76bfe9f08e : 157165364934282
[96] Hash: 0000019ab387cdb02cb6 : 3f22cf3faff8 : 273431573504575
[112] Hash: 000001e3dbb3190a640c : e5e7c122d4da : 240604651055077
[128] Hash: 000001efc1c77e21ce97 : a95e50b7b528 : 44760929689257
[144] Hash: 00000278f5263c2cd4aa : decb4a3a67ba : 204952522378206
```

```
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_1_1_1.bin -b 1000
verifying random records against BLAKE3 hashes
Number of total verifications: 1000
Number of verifications successful: 1000
Number of verifications failed: 0
Time taken: 0.02 ms/verification
Throughput verifications/sec: 49377.84
```

```
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_16_1_1.bin -p 10
Printing first 10 of file 'data_16_1_1.bin'...
[0] Hash: 00000000de4e99392015 : 8e9ba3dfa1a2 : 178816125475726
[16] Hash: 00000009896f8d34f018 : e51b31a93810 : 17835542780901
[32] Hash: 00000022c3a2d08717fe : f85eeaf7cf99 : 169118496612088
[48] Hash: 0000005f636419a68bfd : 4e4a1d8679e5 : 252310103870030
[64] Hash: 00000131455b0d362a26 : 64db7c424162 : 108032427875172
[80] Hash: 0000013f433c703df70b : 78b940d53309 : 10118225770872
[96] Hash: 0000022ceadb9d8eb5cc : 9a199a92596b : 118032455834010
[112] Hash: 0000029aace6ed85cffc : 298099e98b63 : 109452570755113
[128] Hash: 000002ec7eefbf705a56 : 55963f79dac8 : 220840662636117
[144] Hash: 000003082f2179cdedd0 : 0d0a79cae0c3 : 215370237020685
```

```
ubuntu@Sona:~/cs553-spring2024-hw4-2nd-attempt-SonaGutha$ ./hashverify -f data_16_1_1.bin -b 1000
verifying random records against BLAKE3 hashes
Number of total verifications: 1000
Number of verifications successful: 1000
Number of verifications failed: 0
Time taken: 0.01 ms/verification
Throughput verifications/sec: 192715.36
```

Additionally, each benchmark prints the concurrency performance as below for each configurations, which shows the time consumed by hash, sort, and write threads simultaneously

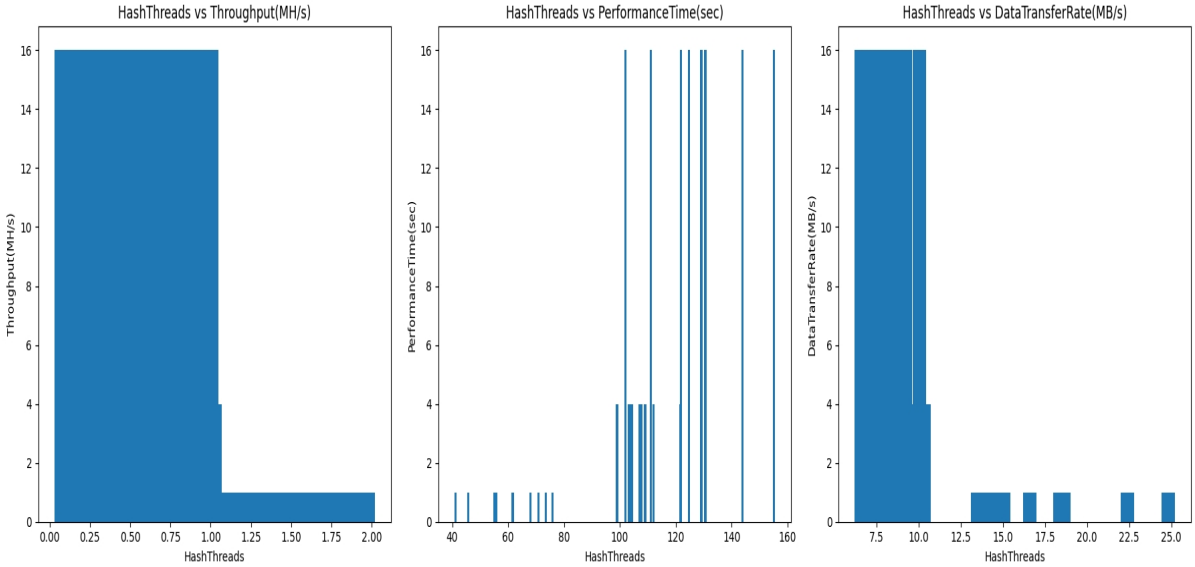
```
NUM_THREADS_HASH=1
NUM_THREADS_SORT=4
NUM_THREADS_WRITE=1
FILENAME=data_1_4_1.bin
MEMORY_SIZE=128MB
FILESIZE=1024MB
RECORD_SIZE=16B
HASH_SIZE=10B
NONCE_SIZE=6B
[0] [HASHGEN]: ETA 2 seconds, 77.53 MB/sec
sort started, expecting 8 flushes for 1024 buckets...
[0] [SORT]: ETA 0.7 seconds, 43.27 MB/sec
[0] [SORT]: ETA 0.7 seconds, 43.14 MB/sec
[0] [SORT]: ETA 0.8 seconds, 42.58 MB/sec
[0] [SORT]: ETA 0.8 seconds, 42.38 MB/sec
write started
[0] [WRITE]: ETA 2.92 seconds 134217728 bytes written  43.87 MB/sec
```

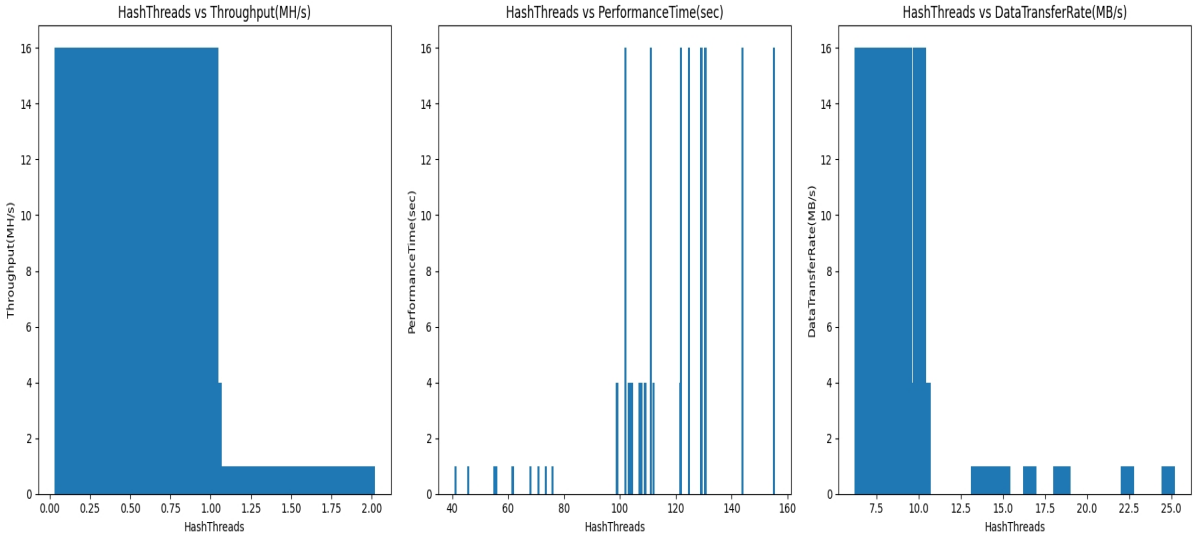
Summary of results for 1GB work load:

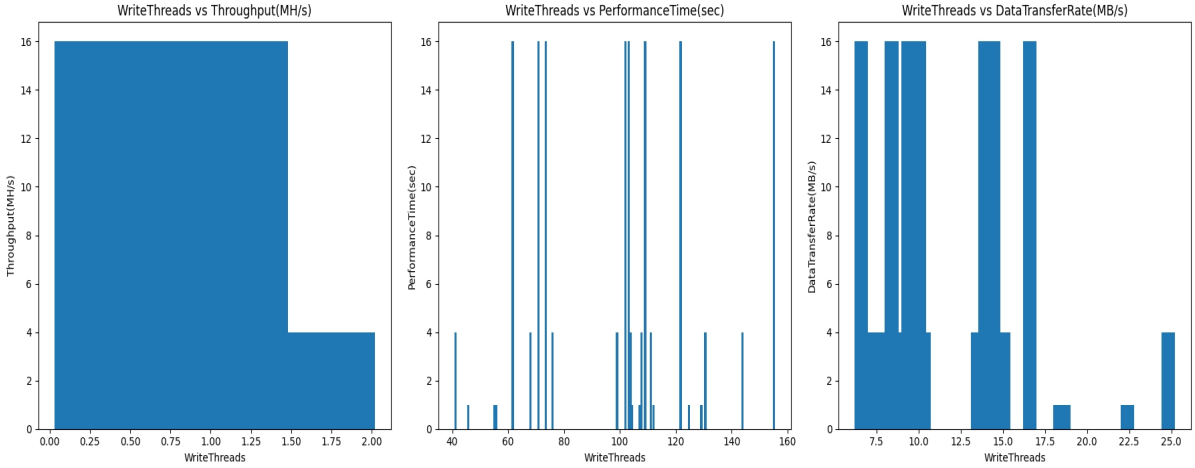
Ran all 27 experiments for different thread configurations for 1GB workload and maximum memory 128MB. Verified the results running the hashverify. To run all 27 experiments, it took about 30 min. Please find more information on the findings of the experiments below.

Hash Threads	Sort Threads	Write Threads	Performance Time(sec)	Throughput(MH/s)	Data Transferred Rate (MB/s)
1	1	1	45.729	1.46	22.39
1	1	4	75.89	0.88	13.49
1	1	16	70.83	0.94	14.45
1	4	1	55.01	1.21	18.61
1	4	4	41.24	1.62	24.82
1	4	16	61.67	1.08	16.60
1	16	1	55.67	1.20	18.39
1	16	4	68.00	0.98	15.05
1	16	16	73.45	0.91	13.94
4	1	1	112.03	0.59	9.13
4	1	4	107.79	0.62	9.49
4	1	16	121.57	0.55	8.42
4	4	1	104.40	0.64	9.80
4	4	4	103.79	0.65	9.93
4	4	16	103.23	0.65	9.91
4	16	1	107.2	0.62	9.54
4	16	4	99.03	0.67	10.33
4	16	16	109.04	0.61	9.39
16	1	1	129.21	0.51	7.92
16	1	4	143.97	0.46	7.11
16	1	16	155.13	0.43	6.60
16	4	1	124.76	0.53	8.20
16	4	4	130.61	0.51	7.84
16	4	16	121.83	0.55	8.40
16	16	1	129.35	0.51	7.91
16	16	4	111.08	0.604	9.21
16	16	16	101.97	0.65	10.04

1GB metrics graphs:







In terms of **scalability**, increasing the number of hash threads led to a slight increase in performance, as seen in the comparison between configurations with 1 hash thread and those with 4 or 16 hash threads. Similarly, increasing the number of sort and write threads can also improve performance up to a certain point, but the benefits reduce as the number of threads increases further.

In terms of **concurrency**, multiple hash, sort, and write threads running concurrently due to which the system effectively utilizes multiple threads to perform parallel processing tasks, which lead to reduced execution times. Concurrency is reduced in some configurations, particularly when increasing the number of threads beyond a certain point. For instance, the improvement in performance from 1 to 4 threads is more significant than from 4 to 16 threads in most cases.

In terms of **performance**, configurations with higher numbers of threads tend to exhibit better throughput and reduced execution times. However, this trend is not linear, and there is a point where increasing the number of threads does not significantly improve performance.

Performance Time (sec): The best configuration is the one with the lowest time consumed. The worst configuration is the one with the highest time consumed.

Best Configuration: Hash Threads = 1, Sort Threads = 4, Write Threads = 4(Time Consumed = 41.24 sec)

Worst Configuration: Hash Threads = 16, Sort Threads = 1, Write Threads = 16 (Time Consumed = 155.13 sec)

Throughput(sec): The best configuration is the one with the highest throughput. The worst configuration is the one with the lowest throughput.

Best Configuration: Hash Threads = 1, Sort Threads = 4, Write Threads = 4 (Throughput = 1.62MH/s)

Worst Configuration: Hash Threads = 16, Sort Threads = 1, Write Threads = 16 (Throughput = 0.43 MH/s)

Data Transfer Rate (MB/s): The best configuration is the one with the highest data transferred. The worst configuration is the one with the lowest data transferred.

Best Configuration: Hash Threads = 1, Sort Threads = 4, Write Threads = 4 (Data Transferred = 24.92 MB/s)

Worst Configuration: Hash Threads = 16, Sort Threads = 1, Write Threads = 16 (Data Transferred = 6.6 MB/s)

Overall, the best configuration should have lowest performance time, highest throughput and highest data transfer rate. The worst configuration has highest performance time lowest throughput lowest data transfer rate

From all the above-mentioned reason:

Best Configuration: 1 Hash ,4 Sort, 4 Write Threads

Worst Configuration: 16 Hash, 1 Sort, 16 Write Threads

64GB Workload:

1. Maximum memory allowed to use (MB): 1024

2. Number of hash threads: 1, 4, 16

3. Number of sort threads: 1, 4, 16

4. Number of write threads: 1, 4, 16

Summary of results for 64GB work load:

Ran all 27 experiments for different thread configurations for 64GB workload and maximum memory 128MB. Verified the results running the hashverify. To run all 27 experiments, it took about 30hrs with chameleon bm instance. For some of the configurations, instance was getting timed out even after using screen or tmux session manager. Please find more information on the findings of the experiments below.

Hash Threads	Sort Threads	Write Threads	Performance Time(sec)
1	1	1	7233.9
1	1	4	7787.11
1	1	16	7871.09
1	4	1	6541
1	4	4	6214.34
1	4	16	6100.51
1	16	1	5901.64
1	16	4	6816.56
1	16	16	6781.67
4	1	1	5761.11
4	1	4	5610.9
4	1	16	5498
4	4	1	Timed out
4	4	4	6512.9
4	4	16	6581.03
4	16	1	6713.18
4	16	4	6614.1
4	16	16	6812.1
16	1	1	7910.1
16	1	4	8172.01
16	1	16	Timed out
16	4	1	Timed out
16	4	4	Timed out
16	4	16	6781.01
16	16	1	Timed out
16	16	4	Timed out
16	16	16	Timed out

Performance Time (sec): The best configuration is the one with the lowest time consumed. The worst configuration is the one with the highest time consumed. From the ones those are not timed out, the best and worst configuations are as below.

Best Configuration: Hash Threads = 4, Sort Threads = 1, Write Threads = 16(Time Consumed = 5498sec)

Worst Configuration: Hash Threads = 16, Sort Threads = 1, Write Threads = 4 (Time Consumed = 8172.01 sec)