CS 553 Cloud Computing
Homework 3

- Overall Benchmark Design

  - MyDiskBench.cpp is the main file
    - To run the read command first execute the write command to generate the files and then use the read command
    - It takes 3 command line arguments in the given order
      - Argument 1 : Record Size (64KB/1MB/16MB)
      - Argument 2 : No. of threads (1/2/4/8/12/24)
      - Argument 3 : Access Pattern (WS/RS/WR/RR)
    - pthread library is used to create the given number of threads and execute the given function
    - The random/sequential read/write is performed on the following data with the given no of threads:
      - 1 thread -> 1 file of 10GB
      - 2 threads -> 2 files of 5GB
      - 4 threads -> 4 files of 2.5 GB
      - 8 threads -> 8 files of 1.25 GB
      - 12 threads -> 12 files of 833.33 GB
      - 24 threads -> 24 files of 416.67 GB
      - 48 threads -> 48 files of 208.33 GB
    - MyDiskBench.cpp calls 4 different functions:
      - ranRead(void *) : for random read
      - ranWrite(void *) : for random write
      - seqRead(void *) : for sequential read
      - seqWrite(void *) : for sequential write
  - utils.cpp
    - random(filesize,recordsize) : returns a random number between 0 and (filesize – recordsize)
    - getFileSize(filename) : returns size of file in bytes
  - randomRead.cpp is file which contains ranRead(void *)
    - We allocate character read buffer using posix_memalign()
    - Size of the read buffer is equal to the given record size.
    - open() is used to option the file with O_DIRECT flag to avoid reading from the cache
    - Then fseek() is used to change the position of the file pointer
    - random() is a user defined function to give a random position for the file pointer.
    - read() is used to randomly read data from the file
    - We will continue to read until it has read data equivalent to the given file size
  - randomWrite.cpp
    - We allocate character write buffer using posix_memalign()
    - We write data into the write buffer.
    - Size of the write buffer is equal to the given record size.
    - open() is used to option the file with O_DIRECT flag to avoid reading from the cache
    - Then fseek() is used to change the position of the file pointer

        - random() is a user defined function to give a random position for the file pointer.
        - write() is used to randomly read data from the file
        - We will continue to write until it has written data equivalent to the given file size
    - sequentialRead.cpp
        - We allocate character buffer using posix_memalign()
        - Size of the read buffer is equal to the given record size.
        - open() is used to option the file with O_DIRECT flag to avoid reading from the cache
        - read() is used to read data from the file
        - We will continue to read until it has reached the end of the file
    - sequentialWrite.cpp
        - We allocate character write buffer using posix_memalign()
        - We write data into the write buffer.
        - Size of the write buffer is equal to the given record size.
        - open() is used to option the file with O_DIRECT flag to avoid reading from the cache
        - We calculate the no. of iteration required to write data equivalent to file size by dividing filesize by record size.
        - Using a for loop we write the data into the file using write()
    - Makefile
        - Used to compile the code and generate executable file

- Improvements and extensions to the program
    - Read can be improved
        - we don't have to use the write function before read
        - the files to be read will be created implicitly and then read will be performed
    - Random read/write can be more efficient
        - Finding a more efficient way to change the file pointer and read/write data randomly
    - Multiple threads can read/write to a file
        - Right now a one thread can read a single file or write to a single file.
        - We can improve this by using multiple threads to read or write from a given file by using locks or semaphores.

- Computing Node
    - Haswell compute
    - Storage
        - Seagate ST9250610NS
        - 7200 RPM
        - Average latency : 4.16ms
        - Average Seek Time: 8.5/9.5
        - Max. Sustained throughput : 115 MB/s

- Output tables

WS

Workload	Concurrency	Record Size	MyDiskBench Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)	Theoretical Throughput (MB/sec)	MyDiskBench Efficiency (%)	IOZone Efficiency (%)
WS	1	64KB	1	1.25	115	1	2
WS	1	1MB	1.50	1.99	115	1.3	1.73
WS	1	16MB	1.687	2.78	115	1.5	2.41
WS	2	64KB	2	4.86	115	1.7	4.22
WS	2	1MB	9	17.77	115	7.8	15.45
WS	2	16MB	47.05	52.88	115	41.3	45
WS	4	64KB	3.2	4	115	2.7	3.4
WS	4	1MB	8.4	10	115	7.3	8.6
WS	4	16MB	40.6	60	115	35.3	52.1
WS	8	64KB	0.587	3	115	0.5	2.6
WS	8	1MB	7.2	5	115	6.5	4.3
WS	8	16MB	41.6	32	115	36.17	27.82
WS	12	64KB	0.56	1.26	115	0.4	1.09
WS	12	1MB	8.5	9.65	115	7.39	8.39
WS	12	16MB	32.10	47	115	27.91	40.86
WS	24	64KB	0.61	1.87	115	0.5	1.6
WS	24	1MB	7.5	8.90	115	6.5	7.7
WS	24	16MB	36.74	43.6	115	31.9	37.9

WS	48	64KB	0.3	2.1	115	0.2	1.8
WS	48	1MB	6.57	14.16	115	5.7	12.3
WS	48	16MB	31.086	45.6	115	27.02	39.6

RS

Workload	**Concurrency**	**Record Size**	**MyDiskBench Measured Throughput (MB/sec)**	**IOZone Measured Throughput (MB/sec)**	**Theoretical Throughput (MB/sec)**	**MyDiskBench Efficiency (%)**	**IOZone Efficiency (%)**
RS	1	64KB	6.36	8.34	115	5.5	7.2
RS	1	1MB	8.7	11.3	115	7.5	9.8
RS	1	16MB	43.70	48	115	38	41.7
RS	2	64KB	6.35	7.8	115	5.5	6.7
RS	2	1MB	8.70	13.1	115	7.5	11.3
RS	2	16MB	44.67	54.88	115	4.06	47.7
RS	4	64KB	0.60	2.4	115	0.5	2.1
RS	4	1MB	8.50	16	115	7.39	14
RS	4	16MB	40.50	88	115	35.2	76.5
RS	8	64KB	0.58	2.56	115	0.5	2.2
RS	8	1MB	8.60	7.67	115	7.4	6.7
RS	8	16MB	51.60	65.76	115	44.9	57.1

RS	12	64KB	0.56	1.78	115	0.48	1.4
RS	12	1MB	8.1	10	115	7	8.7
RS	12	16MB	36.36	56.04	115	31.7	48.7
RS	24	64KB	0.67	2.6	115	0.5	2.2
RS	24	1MB	9.30	11.3	115	8	9
RS	24	16MB	53.6	58.8	115	46.6	51.1
RS	48	64KB	0.46	1.5	115	0.4	1.3
RS	48	1MB	8.07	21.36	115	7	18.6
RS	48	16MB	35	76.8	115	30.4	66.8

WR

Workload	**Concurrency**	**Record Size**	**MyDiskBench Measured Throughput (MB/sec)**	**IOZone Measured Throughput (MB/sec)**	**Theoretical Throughput (MB/sec)**	**MyDiskBench Efficiency (%)**	**IOZone Efficiency (%)**
WR	1	64KB	0.8	1.88	115	0.6	1.6
WR	1	1MB	1	2.7	115	0.8	2.3
WR	1	16MB	1.5	7.2	115	1.3	6.2
WR	2	64KB	1.5	4.1	115	1.3	3.7
WR	2	1MB	7.5	9.2	115	6.5	0.08
WR	2	16MB	42.05	47.7	115	36.7	41.5
WR	4	64KB	2.2	5.8	115	1.9	5.5
WR	4	1MB	6.4	13.69	115	5.6	11.8
WR	4	16MB	35.6	48.64	115	30.9	42.3

WR	8	64KB	0.58	2.77	115	0.5	2.4
WR	8	1MB	6.2	6.71	115	5	5.9
WR	8	16MB	38.6	5.23	115	33.6	4.5
WR	12	64KB	0.46	2	115	0.4	1.7
WR	12	1MB	7.4	3.57	115	6.5	3.1
WR	12	16MB	30.1	2.87	115	26.2	2.4
WR	24	64KB	0.4	3.2	115	0.3	2.8
WR	24	1MB	6	7.8	115	5.2	6.7
WR	24	16MB	34.47	38.8	115	30	33.7
WR	48	64KB	0.2	0.8	115	0.1	0.6
WR	48	1MB	5.57	33.38	115	4.8	29
WR	48	16B	30.6	40	115	26.6	34.8

RR

Workload	**Concurrency**	**Record Size**	**MyDiskBench Measured Throughput (MB/sec)**	**IOZone Measured Throughput (MB/sec)**	**Theoretical Throughput (MB/sec)**	**MyDiskBench Efficiency (%)**	**IOZone Efficiency (%)**
RR	1	64KB	2.34	3	115	2	2.6
RR	1	1MB	6.87	8.2	115	5.9	7.1
RR	1	16MB	43.80	66.8	115	38.0	57.9
RR	2	64KB	2.34	4.8	115	2.03	2.4
RR	2	1MB	6.7	11.56	115	5.8	10
RR	2	16MB	43.20	72	115	37.6	62.6

RR	4	64KB	0.059	7.5	115	0.04	6.5
RR	4	1MB	8.69	14.17	115	7.5	12.3
RR	4	16MB	40.58	78.24	115	35.2	68
RR	8	64KB	0.51	3.2	115	0.4	2.7
RR	8	1MB	8.36	9	115	7.2	7.8
RR	8	16MB	44.16	77.28	115	38.4	67.2
RR	12	64KB	0.54	1	115	0.4	0.8
RR	12	1MB	8.11	10	115	7.7	8.6
RR	12	16MB	32.89	67.44	115	28.5	58.6
RR	24	64KB	0.51	2	115	0.4	1.7
RR	24	1MB	8.50	5	115	7.3	4.3
RR	24	16MB	41.09	62.4	115	35.7	54.3
RR	48	64KB	0.95	5	115	0.8	4.3
RR	48	1MB	12.16	10.7	115	10.5	9.3
RR	48	16MB	33.22	57.6	115	28.9	50.1

For 4KB using 1GB dataset

WR

Workload	**Concurrency**	**Record Size**	**MyDiskBench Measured IOPS (OPS/sec)**	**IOZone Measured IOPS (OPS/sec)**	**Theoretical IOPS (OPS/sec)**	**MyDiskBench Efficiency (%)**	**IOZone Efficiency (%)**
WR	1	4KB	18	20	79	88.6	25.3
WR	2	4KB	20	45	79	25.3	56.9
WR	4	4KB	27	89	79	34	112.6
WR	8	4KB	33	100	79	41.7	126.7
WR	12	4KB	46	133	79	58.2	168.3
WR	24	4KB	50	150	79	63.2	189.8
WR	48	4KB	70	191	79	88.6	241.7

RR

Workload	Concurrency	Record Size	MyDiskBench Measured IOPS (OPS/sec)	IOZone Measured IOPS (OPS/sec)	Theoretical IOPS (OPS/sec)	MyDiskBench Efficiency (%)	IOZone Efficiency (%)
RR	1	4KB	8	20	79	10.1	25.3
RR	2	4KB	10	45	79	12.6	56.9
RR	4	4KB	21	100	79	26.5	126.6
RR	8	4KB	33	185	79	41.6	234.1
RR	12	4KB	46	222	79	548.2	281
RR	24	4KB	66	380	79	83.5	481
RR	48	4KB	123	420	79	155.6	531

- Contribution per student
  - Shreshtha Rojindar : iozone benchmark and execution
  - Prajakta Rajhans : MyDiskBench code and benchmarking