Tiffany Wong
Lab 3: I/O Benchmarking

1 Disk Throughput (MBs/s)

1a. WS - Write Sequential

Workload	Concurrency	Record Size	FileIO Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)
WS	1	64KB	68.632009	346.5520605
WS	1	1MB	52.412339	504.8049902
WS	1	16MB	89.474801	557.0145898
WS	2	64KB	37.891430	467.4756152
WS	2	1MB	68.573523	584.5515723
WS	2	16MB	65.673801	668.7902832
WS	4	64KB	36.596766	514.6049316
WS	4	1MB	74.335592	557.505459
WS	4	16MB	61.104413	652.1648828
WS	8	64KB	34.263920	506.1122949
WS	8	1MB	80.951838	607.7898242
WS	8	16MB	75.832702	687.8084863

Best: Concurrency - 8 and Record Size - 64KB
Worst: Concurrency - 1 and Record Size - 16MB
Writing sequentially means I'm writing over the 1GB file in record size increments, so it makes sense that the worst benchmark for WS is when there's only 1 process being run in the largest record size of 16MB, because you're writing in large increments sequentially, taking up most of the time with just filling in that one increment until it can move on to the next increment. It also makes sense for the best benchmark for WS to be 8 processes in increments of 64KB, the smallest record size, because there are 8 processes running at the same time and they write over in small increments meaning they can jump to the next one quickly.

1b. RS - Read Sequential

Workload	Concurrency	Record Size	FileIO Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)
RS	1	64KB	365.640896	851.2307715

RS	1	1MB	839.527307	2439.158447
RS	1	16MB	962.772484	2602.525146
RS	2	64KB	455.136926	1218.734619
RS	2	1MB	1132.550288	2851.92334
RS	2	16MB	1009.347568	3019.181758
RS	4	64KB	343.371926	966.1615918
RS	4	1MB	1019.747409	1469.037021
RS	4	16MB	1105.702995	1477.995059
RS	8	64KB	438.530327	735.2122949
RS	8	1MB	961.227911	1491.201152
RS	8	16MB	1001.826329	1534.421436

Best: Concurrency - 4 and Record Size - 64KB
Worst: Concurrency - 2 and Record Size - 1MB
Reading sequentially means I'm reading in the 1GB file in record size increments.
I think ideally, my results in RS and WS would be similar to what is the best and worst benchmarks. However, the worst benchmark for RS is when there are 4 processes being run in the middle record size of 1MB. This means I'm reading in large-ish increments sequentially, taking up most of the time with just reading in that one increment until it can move on to the next increment. Ideally, it would be when there's only 1 process that runs the slowest. It can make sense for the best benchmark for WS to be 4 processes in increments of 64KB, the smallest record size, because there are 8 processes running at the same time and they read in small increments meaning they can jump to the next one quickly.

1c. WR - Write Random

Workload	Concurrency	Record Size	FileIO Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)
WR	1	64KB	42.963273	400.092832
WR	1	1MB	106.220210	762.3100732
WR	1	16MB	101.557389	805.1358838
WR	2	64KB	49.053522	649.5368701
WR	2	1MB	88.167149	838.0722217
WR	2	16MB	127.441525	824.5866504

WR	4	64KB	48.182396	115.8395801
WR	4	1MB	92.699214	386.7795313
WR	4	16MB	137.296234	423.5775977
WR	8	64KB	44.742297	106.517832
WR	8	1MB	99.071885	237.9181641
WR	8	16MB	146.388986	212.234082

Best: Concurrency - 1 and Record Size - 64KB
Worst: Concurrency - 8 and Record Size - 16MB
Writing randomly means I'm jumping to a random place in the 1GB file in order to write over a record size number of data and then jumping to another random place in the file to write over next. It makes sense to me that the best configuration for WR is when there's 1 process, writing randomly in a record size of 64KB because that means there's one process going through the 1GB file and writing over in the smallest increment size and then moving on to the next random place to write over. To complement this, the worst configuration is when there are 8 processes, writing randomly over in a record size of 16MB, the largest record size. This also makes sense because there are 8 processes running all at once and it's writing over in the largest possible record size, meaning it takes longer to write over that section of the 1GB file first than it would for a smaller record size.

1d. RR - Read Random

Workload	Concurrency	Record Size	FileIO Measured Throughput (MB/sec)	IOZone Measured Throughput (MB/sec)
RR	1	64KB	139.971940	664.1962305
RR	1	1MB	441.617060	2560.486572
RR	1	16MB	1198.744596	2613.878174
RR	2	64KB	160.774832	1114.80585
RR	2	1MB	352.114288	3235.178828
RR	2	16MB	1467.857588	3067.521484
RR	4	64KB	169.397381	155.7041113
RR	4	1MB	338.686155	478.8265039
RR	4	16MB	1597.676340	1280.212803
RR	8	64KB	124.760180	110.3136816
RR	8	1MB	342.844291	278.2297559

RR	8	16MB	2209.681055	960.8535449

Best: Concurrency - 8 and Record Size - 64KB
Worst: Concurrency - 8 and Record Size - 16MB
Reading randomly means I'm jumping to a random place in the 1GB file in order to read in a record size number of data and then jumping to another random place in the file to read over next. Ideally, the best and worst configurations would be the same as WR. However, the best configuration for RR is when there are 8 processes, reading randomly in a record size of 64KB. The benchmark for this was close to the benchmark for 1 process and 64KB record size, so this configuration could make sense because that means there are 8 processes going through the 1GB file and reading in the smallest increment size and then moving on to the next random place to write over. It's reading it much faster because there are 8 processes doing it all at once and it's doing it in small increments, which moves on to the next increment faster. To complement this, the worst configuration is the same from WR, where there are 8 processes, reading randomly in a record size of 16MB, the largest record size. This also makes sense because there are 8 processes running all at once and it's reading in the largest possible record size, meaning it takes longer to read in that section of the 1GB file first than it would for a smaller record size.

2 Disk Latency (OPs/s)

To benchmark latency, there was a set record size of 4KB and only the number of processes changed.

2a. WR - Write Random

Workload	Concurrency	Record Size	FileIO Measured Throughput (OPs/sec)	IOZone Measured Throughput (OPs/sec)
WR	1	4KB	4407.504810	25960.12
WR	2	4KB	5142.291835	36409.85
WR	4	4KB	6138.205534	30455.6
WR	8	4KB	6527.115576	19596.54

Best: Concurrency - 1
Worst: Concurrency - 8
Writing randomly means I'm jumping to a random place in the 1GB file in order to write over a record size number of data and then jumping to another random place in the file to write over next. It makes sense to me that the best configuration for WR is when there's 1 process, because that means there's one process going through the 1GB file and writing over in a set record size and then moving on to the next random place to write over. To complement this, the worst configuration is when there are 8 processes, which also makes sense because there are 8 processes running all at once, meaning it takes longer to write over that section of the 1GB file first than it would for a smaller record size.

2b. RR - Read Random

Workload	Concurrency	Record Size	FileIO Measured Throughput (OPs/sec)	IOZone Measured Throughput (OPs/sec)
RR	1	4KB	12667.780046	45691.72
RR	2	4KB	17153.290785	50245.36
RR	4	4KB	21098.611283	29233.96
RR	8	4KB	10516.154210	22842.7

Best: Concurrency - 8

Worst: Concurrency - 4

Reading randomly means I'm jumping to a random place in the 1GB file in order to read in a record size number of data and then jumping to another random place in the file to read over next. Ideally, the best and worst configurations would be the same as WR. However, the best configuration for RR is when there are 8 processes. This configuration could make sense because that means there are 8 processes going through the 1GB file and reading in the smallest increment size and then moving on to the next random place to write over. I'm assuming that my I/O system reads faster than it can write over a file. It's reading in much faster because there are 8 processes doing it all at once. However, the worst configuration is when there are 4 processes, the middle of the numbers given. This could just be a flub in my I/O system from reading in a lot of data previously, but it is worse than 8 processes because 4 processes mean there are less active children trying to read in the whole 1 in split parts.