HW3

Luke Logan, Gurunath Reddy

3/31/2020

1 Design

We ran our tests on a Chameleon Skylake node, which is equipped with an SSD. The SSD is a 240GB and has the model number MZ7KM480HMHQ. This device has a theoretical throughput of of 510 MB/s and 95000 IOPS (4K) [1].

We used the C programming language to construct MyDiskBenchmark in order to build programs close to the hardware. We used the POSIX interface in order to perform reads and writes (open, read, write, close) as well as for threading (pthreads). For write tests, MyDiskBenchmark opens files with the O_DIRECT, O_CREAT, and O_SYNC flags. The O_DIRECT flag bypasses OS caches when it performs reads/writes, and the O_SYNC flag makes it so that writes flush all data and metadata to the SSD before returning from the write system call (this guarantees I/O is synchronous). In other words, we make less use of caches, causing the full penalty of accessing the SSD to occur frequently every call to read/write. We also destroy all temporary files before each write benchmark (WS and WR) so that every write is performed on a new file.

When the benchmarks start, there are always at least two threads: the main thread and the I/O threads. When only a single file is being benchmarked, we send the file to a new thread in addition to the main thread. In order to get performance metrics, we use CLOCK_MONOTIC_RAW, which has a precision in nanoseconds. We chose this clock because it counts the time while the process is in blocked mode, unlike the clock() function. In order to get accurate results, we encompass the code that waits for I/O threads to terminate with the clocks. This has two concerns: (1) The I/O threads may start before the clock starts, and (2) We are including the costs of software (thread switching, random number generation, loop increments, etc) in our timing calculation. The problem with (1) is that the benchmark may report a throughput that is faster than the actual value. The problem with (2) is that the software overhead may be significant and will cause the benchmark to report a throughput that is slower than the actual value. In the future, it would be better to wrap every call to read/write with clocks OR to use something like ftrace, which augments kernel code with timers, in order to get accurate timing measurements.

We used bash scripting and python in order to automate testing. Before every write test, we clear the disk cache, but not for the reads. This was done under the assumption that the disk cache would have been filled after the write tests were performed and did not need to be flushed. We perform the read tests immediately after the write tests because the write tests will generate the temporary files needed for the read tests.

We call IOZone with the following parameters: -I -o -T. -I enables O_DIRECT, -o enables O_SYNC, and -T enables pthreads for multithreading. This is analagous to MyDiskBenchmark.

2 Tables

MyDiskBench Measured Throughput - MD MT
IOZone Measured Throughput - IOZone MT
Theoretical Throughput - TT
Efficiency - Eff

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
RR	48	4096.0	125317.26	81588.52	90000	139.24	90.65
RR	24	4096.0	103052.93	67136.64	90000	114.5	74.6
RR	12	4096.0	92923.51	59038.11	90000	103.25	65.6
RR	8	4096.0	71097.0	49121.73	90000	79.0	54.58
RR	4	4096.0	38577.01	29681.62	90000	42.86	32.98
RR	2	4096.0	19538.96	16024.4	90000	21.71	17.8
RR	1	4096.0	8468.37	8514.13	90000	9.41	9.46

Table 1: latency-read-rand-iops

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
WR	48	4096.0	3496.38	4409.22	90000	3.88	4.9
WR	24	4096.0	2471.54	2847.67	90000	2.75	3.16
WR	12	4096.0	1326.17	1441.63	90000	1.47	1.6
WR	8	4096.0	903.58	960.55	90000	1.0	1.07
WR	4	4096.0	470.49	483.12	90000	0.52	0.54
WR	2	4096.0	239.2	244.1	90000	0.27	0.27
WR	1	4096.0	120.18	126.32	90000	0.13	0.14

Table 2: latency-write-rand-iops

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
RR	48	65536.0	598.16	394.62	510	117.29	77.38
RR	24	65536.0	584.0	405.47	510	114.51	79.5
RR	12	65536.0	540.8	405.0	510	106.04	79.41
RR	8	65536.0	513.11	392.47	510	100.61	76.95
RR	4	65536.0	467.61	355.39	510	91.69	69.68
RR	2	65536.0	318.54	330.85	510	62.46	64.87
RR	1	65536.0	210.0	234.32	510	41.18	45.95
RR	48	1048576.0	640.42	399.27	510	125.57	78.29
RR	24	1048576.0	637.83	396.48	510	125.06	77.74
RR	12	1048576.0	624.43	400.9	510	122.44	78.61
RR	8	1048576.0	627.87	410.32	510	123.11	80.46
RR	4	1048576.0	579.57	410.18	510	113.64	80.43
RR	2	1048576.0	468.5	384.21	510	91.86	75.33
RR	1	1048576.0	328.14	317.98	510	64.34	62.35
RR	48	16777216.0	573.32	372.48	510	112.42	73.04
RR	24	16777216.0	585.91	380.16	510	114.88	74.54
RR	12	16777216.0	582.2	385.2	510	114.16	75.53
RR	8	16777216.0	567.28	377.04	510	111.23	73.93
RR	4	16777216.0	551.15	390.16	510	108.07	76.5
RR	2	16777216.0	491.58	394.32	510	96.39	77.32
RR	1	16777216.0	391.99	379.04	510	76.86	74.32

Table 3: read-rand-mbps

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
RS	48	65536.0	605.69	397.19	510	118.76	77.88
RS	24	65536.0	597.19	417.04	510	117.1	81.77
RS	12	65536.0	564.0	419.15	510	110.59	82.19
RS	8	65536.0	550.41	395.91	510	107.92	77.63
RS	4	65536.0	526.84	356.95	510	103.3	69.99
RS	2	65536.0	401.94	343.74	510	78.81	67.4
RS	1	65536.0	279.83	379.32	510	54.87	74.38
RS	48	1048576.0	656.32	392.26	510	128.69	76.91
RS	24	1048576.0	656.71	391.11	510	128.77	76.69
RS	12	1048576.0	658.51	395.08	510	129.12	77.47
RS	8	1048576.0	674.02	410.18	510	132.16	80.43
RS	4	1048576.0	666.93	405.6	510	130.77	79.53
RS	2	1048576.0	592.88	381.75	510	116.25	74.85
RS	1	1048576.0	484.75	350.48	510	95.05	68.72
RS	48	16777216.0	632.43	405.36	510	124.01	79.48
RS	24	16777216.0	629.9	394.56	510	123.51	77.36
RS	12	16777216.0	615.11	383.52	510	120.61	75.2
RS	8	16777216.0	629.45	388.8	510	123.42	76.24
RS	4	16777216.0	625.23	391.6	510	122.59	76.78
RS	2	16777216.0	657.8	399.12	510	128.98	78.26
RS	1	16777216.0	598.68	381.36	510	117.39	74.78

Table 4: read-seq-mbps

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
WR	48	65536.0	73.8	104.07	510	14.47	20.41
WR	24	65536.0	67.71	88.92	510	13.28	17.43
WR	12	65536.0	58.91	64.32	510	11.55	12.61
WR	8	65536.0	49.04	52.11	510	9.61	10.22
WR	4	65536.0	28.42	29.29	510	5.57	5.74
WR	2	65536.0	14.95	15.12	510	2.93	2.96
WR	1	65536.0	7.75	7.53	510	1.52	1.48
WR	48	1048576.0	248.82	265.44	510	48.79	52.05
WR	24	1048576.0	235.42	317.38	510	46.16	62.23
WR	12	1048576.0	229.49	317.88	510	45.0	62.33
WR	8	1048576.0	219.35	302.36	510	43.01	59.29
WR	4	1048576.0	194.69	231.79	510	38.17	45.45
WR	2	1048576.0	114.67	150.95	510	22.48	29.6
WR	1	1048576.0	89.51	77.81	510	17.55	15.26
WR	48	16777216.0	306.03	395.2	510	60.01	77.49
WR	24	16777216.0	352.41	360.0	510	69.1	70.59
WR	12	16777216.0	359.91	325.68	510	70.57	63.86
WR	8	16777216.0	366.54	365.28	510	71.87	71.62
WR	4	16777216.0	316.86	320.24	510	62.13	62.79
WR	2	16777216.0	277.75	257.68	510	54.46	50.53
WR	1	16777216.0	276.53	253.28	510	54.22	49.66

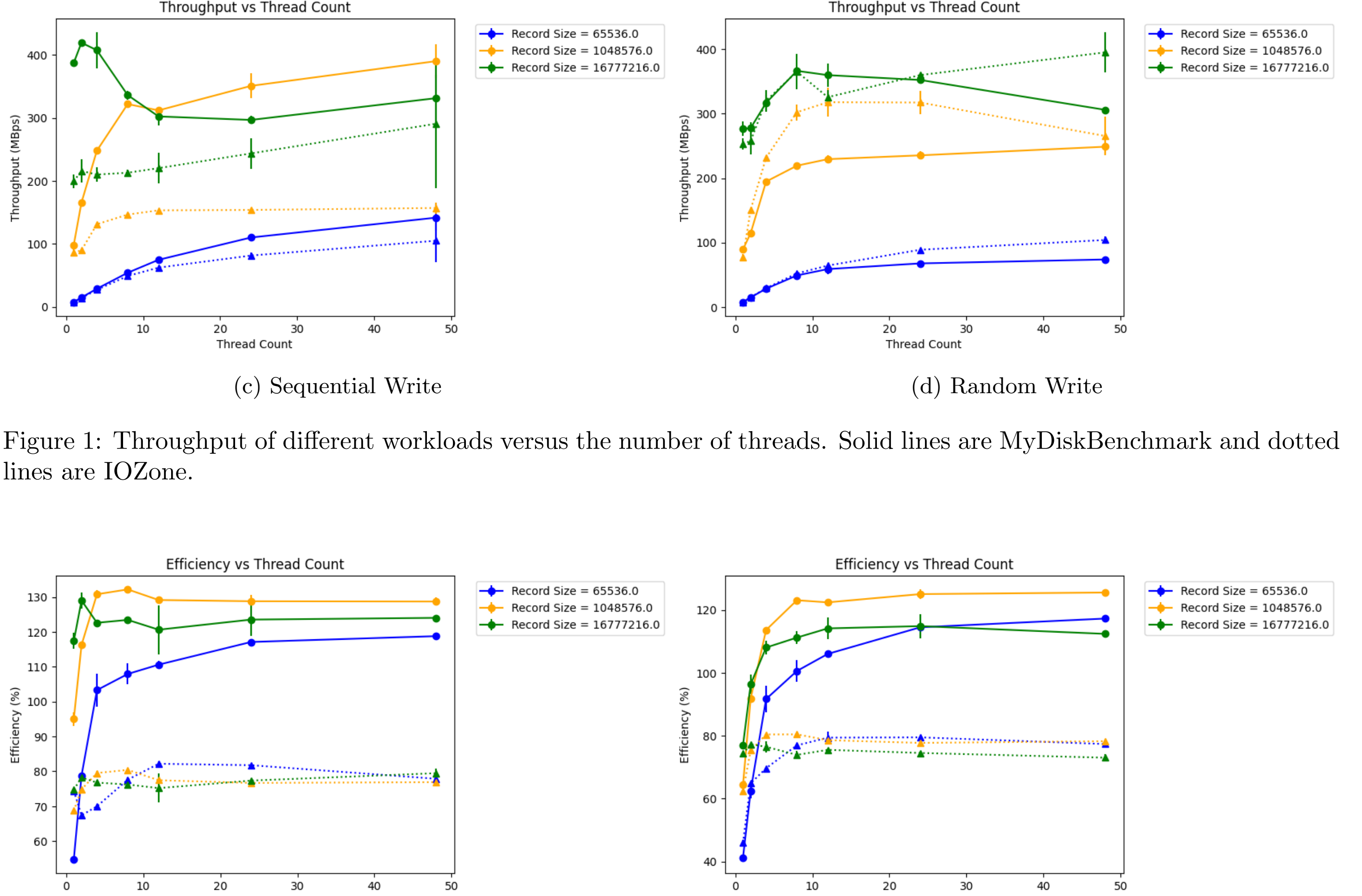
Table 5: write-rand-mbps

Workload	Concurrency	Record Size	MD MT (MB/sec)	IOZone MT (MB/sec)	TT (MB/sec)	MyDiskBench Eff (%)	IOZone Eff (%)
WS	48	65536.0	141.62	105.02	510	27.77	20.59
WS	24	65536.0	110.13	81.5	510	21.6	15.98
WS	12	65536.0	74.72	62.33	510	14.65	12.22
WS	8	65536.0	54.21	49.02	510	10.63	9.61
WS	4	65536.0	28.81	27.23	510	5.65	5.34
WS	2	65536.0	14.91	14.21	510	2.92	2.79
WS	1	65536.0	7.74	7.39	510	1.52	1.45
WS	48	1048576.0	389.97	156.95	510	76.46	30.78
WS	24	1048576.0	350.7	153.88	510	68.77	30.17
WS	12	1048576.0	311.82	153.24	510	61.14	30.05
WS	8	1048576.0	321.84	146.5	510	63.11	28.73
WS	4	1048576.0	247.85	131.2	510	48.6	25.73
WS	2	1048576.0	165.41	89.74	510	32.43	17.6
WS	1	1048576.0	98.21	86.55	510	19.26	16.97
WS	48	16777216.0	331.21	290.64	510	64.94	56.99
WS	24	16777216.0	296.68	243.52	510	58.17	47.75
WS	12	16777216.0	302.23	220.16	510	59.26	43.17
WS	8	16777216.0	336.08	212.88	510	65.9	41.74
WS	4	16777216.0	407.3	210.0	510	79.86	41.18
WS	2	16777216.0	419.27	215.68	510	82.21	42.29
WS	1	16777216.0	387.11	199.52	510	75.9	39.12

Table 6: write-seq-mbps

3 Results

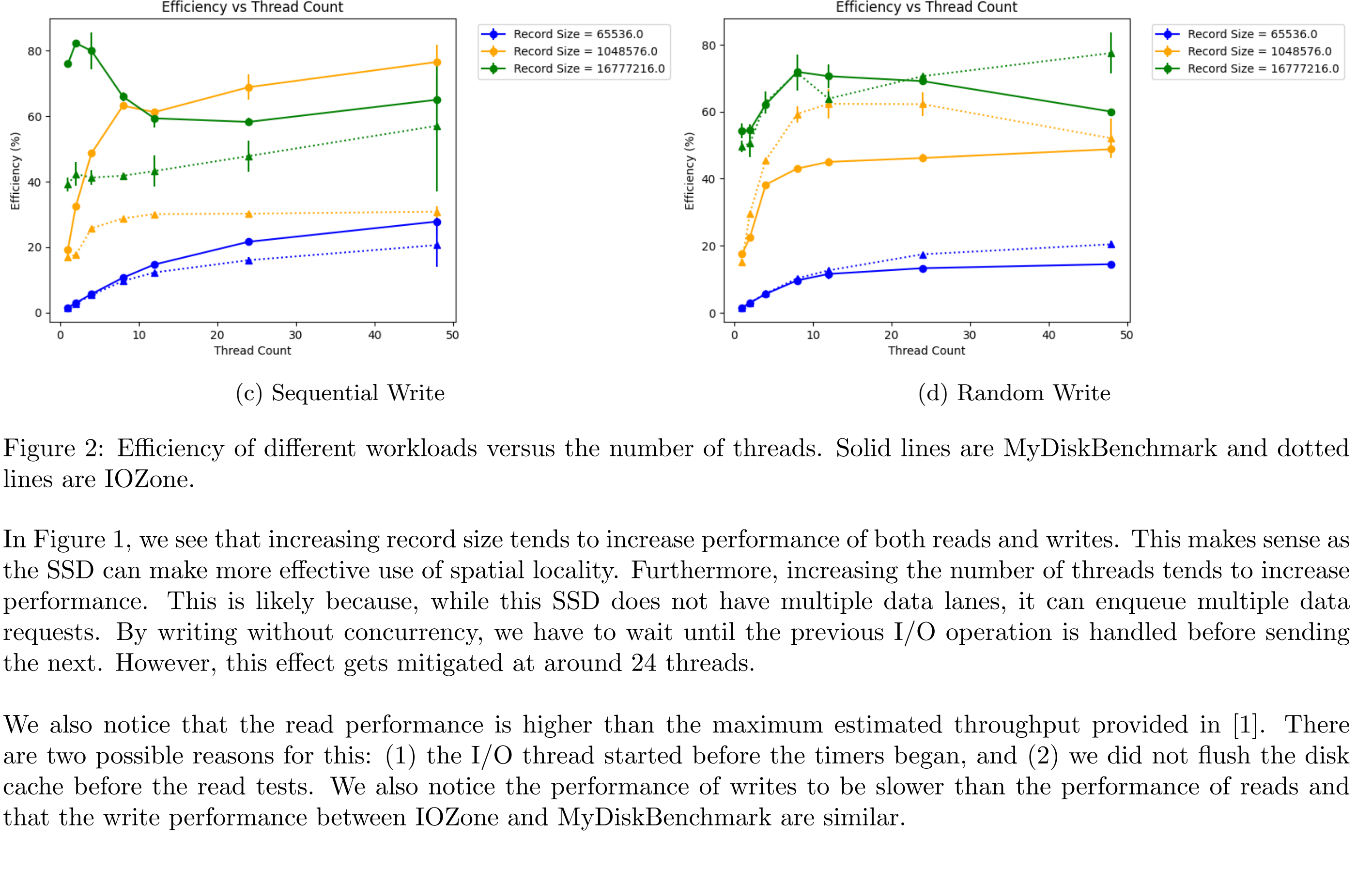
3.1 Throughput Tests



(a) Sequential Read (b) Random Read

(c) Sequential Write (d) Random Write

Figure 1: Throughput of different workloads versus the number of threads. Solid lines are MyDiskBenchmark and dotted lines are IOZone.



(a) Sequential Read (b) Random Read

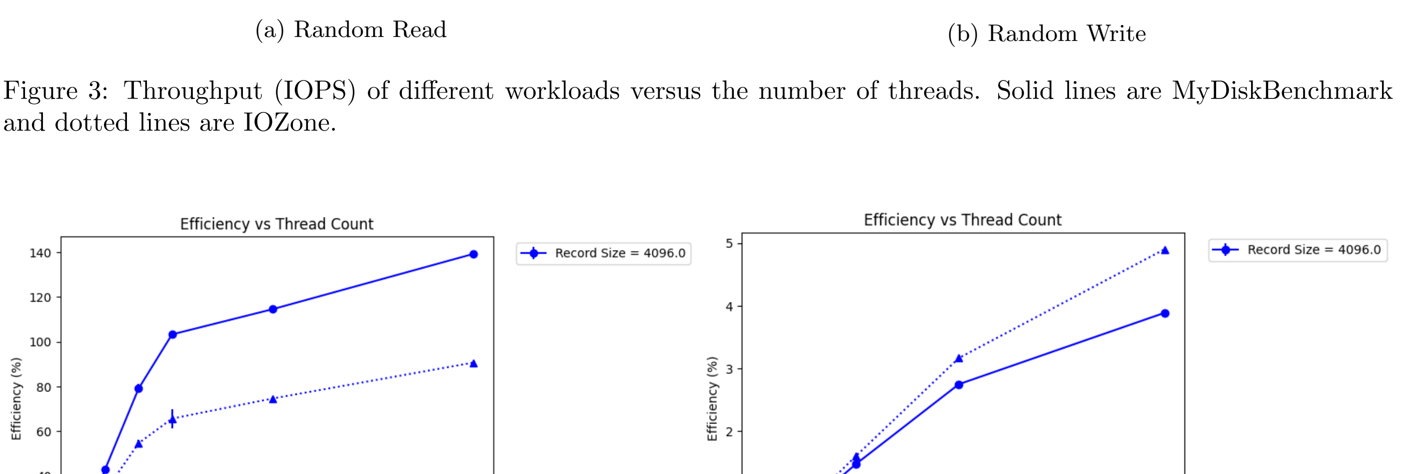
(c) Sequential Write (d) Random Write

Figure 2: Efficiency of different workloads versus the number of threads. Solid lines are MyDiskBenchmark and dotted lines are IOZone.

In Figure 1, we see that increasing record size tends to increase performance of both reads and writes. This makes sense as the SSD can make more effective use of spatial locality. Furthermore, increasing the number of threads tends to increase performance. This is likely because, while this SSD does not have multiple data lanes, it can enqueue multiple data requests. By writing without concurrency, we have to wait until the previous I/O operation is handled before sending the next. However, this effect gets mitigated at around 24 threads.

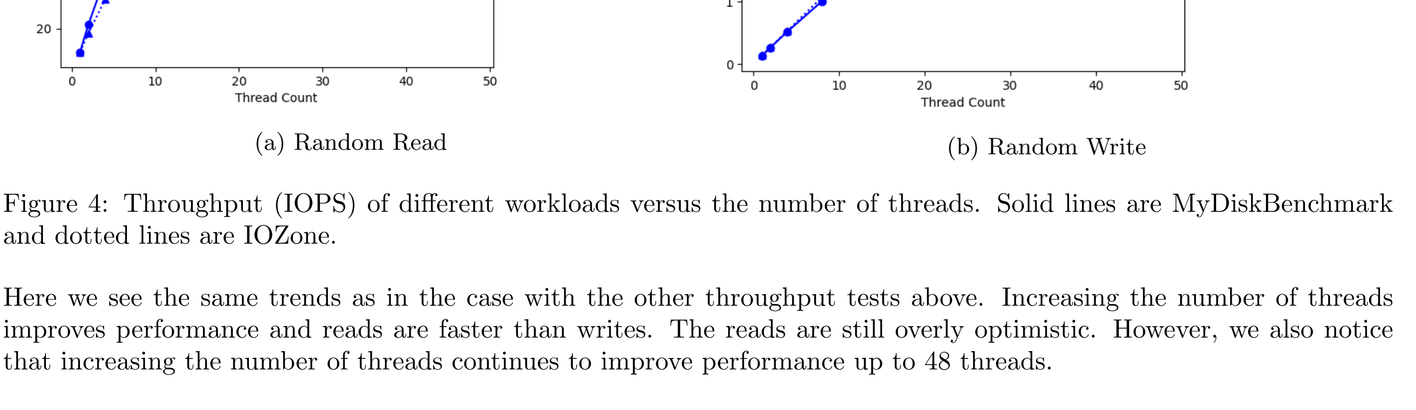
We also notice that the read performance is higher than the maximum estimated throughput provided in [1]. There are two possible reasons for this: (1) the I/O thread started before the timers began, and (2) we did not flush the disk cache before the read tests. We also notice the performance of writes to be slower than the performance of reads and that the write performance between IOZone and MyDiskBenchmark are similar.

3.2 Latency Tests



(a) Random Read (b) Random Write

Figure 3: Throughput (IOPS) of different workloads versus the number of threads. Solid lines are MyDiskBenchmark and dotted lines are IOZone.



(a) Random Read (b) Random Write

Figure 4: Throughput (IOPS) of different workloads versus the number of threads. Solid lines are MyDiskBenchmark and dotted lines are IOZone.

Here we see the same trends as in the case with the other throughput tests above. Increasing the number of threads improves performance and reads are faster than writes. The reads are still overly optimistic. However, we also notice that increasing the number of threads continues to improve performance up to 48 threads.

References

[1] Samsung, "2.5" sata 6gbps sm863a," 2016. [Online]. Available: http://www.farnell.com/datasheets/2213317.pdf