CS 441
11/23/2019
Linden Stirk

Timing Study of the PageRank Algorithm Implemented with Clojure

Introduction

The purpose of this study is to explore the efficiency of multithreading with Clojure. A PageRank algorithm, which ranks the relative importance of webpages, is used to conduct the timing study. The algorithm runs for 1000 iterations and thus is a good candidate for multithreading. The elapsed time to complete the calculations is compared with the number of requested threads. Furthermore, the study also explores how the number of hash map partitions affects execution times.

Methods

The program starts by opening the *pages.txt* file containing information for dummy webpages, and reads in one line at a time, storing the data in a hash map. The key is the page ID, and the value is a vector of the outgoing links to other page ID's. This method produces an individual hash map for each page ID, resulting in 10,000 maps. These were combined with the custom *combine-maps* function. The next step is to find all the incoming links for each page ID. Since the data is random and not sorted, a brute force method is employed by which the entire data structure is searched for each page ID. This is by far the least efficient and most lengthy process of the entire program. In order to improve efficiency, the *some* function is used.

```
(defn search-map [targetID map] (for [[pageIndex outpages] map] (if (some #(= targetID %) (get map pageIndex)) pageIndex)))
```

If the query is found in the vector, instead of checking the value at each remaining index of the vector, the search immediately stops and moves onto the next page entry. Another hash map is created that stores the page ID and a count of the outgoing links on that page. I chose to use another data structure to store this information rather than recalculating the count for each entry every iteration of the loop. The third hash map stores the PageRank. Before the looping begins, the PageRank of all page ID's is initialized to 1. Inside the loop, the PageRank function is called and takes the *inpagesMap*, *outpagesCountMap*, and the current *pageRankMap*. The PageRank is calculated by the following formula where *d* is the damping factor (0.85), *PR(T)* is the current PageRank of page *T*, *C(T)* is the count of outgoing links on page *T*, and the summation is over all pages that have an outgoing link to page *A*:

$$PR(A) = (1 - d) + d \sum \frac{PR(T)}{C(T)}$$

Multithreading is incorporated into the *calc-page-rank* function. In the final code, the *modded-pmap* function, modified from the source code of the standard *pmap* function, takes the number of threads to use, *t,* and the sequence of hash maps, *mapPartitions*.

```
(def pageRankMap (combine-maps (modded-pmap #(calc-page-rank % outpagesCountMap pageRankMap) t mapPartitions)))
```

Before I determined a way to specify how many threads were called to run the PageRank loop, I approached the problem by dividing up the hash map into partitions corresponding to the number of threads that I wanted to use. The *mapPartitions* object holds a sequence of hash maps based on the number of partitions.

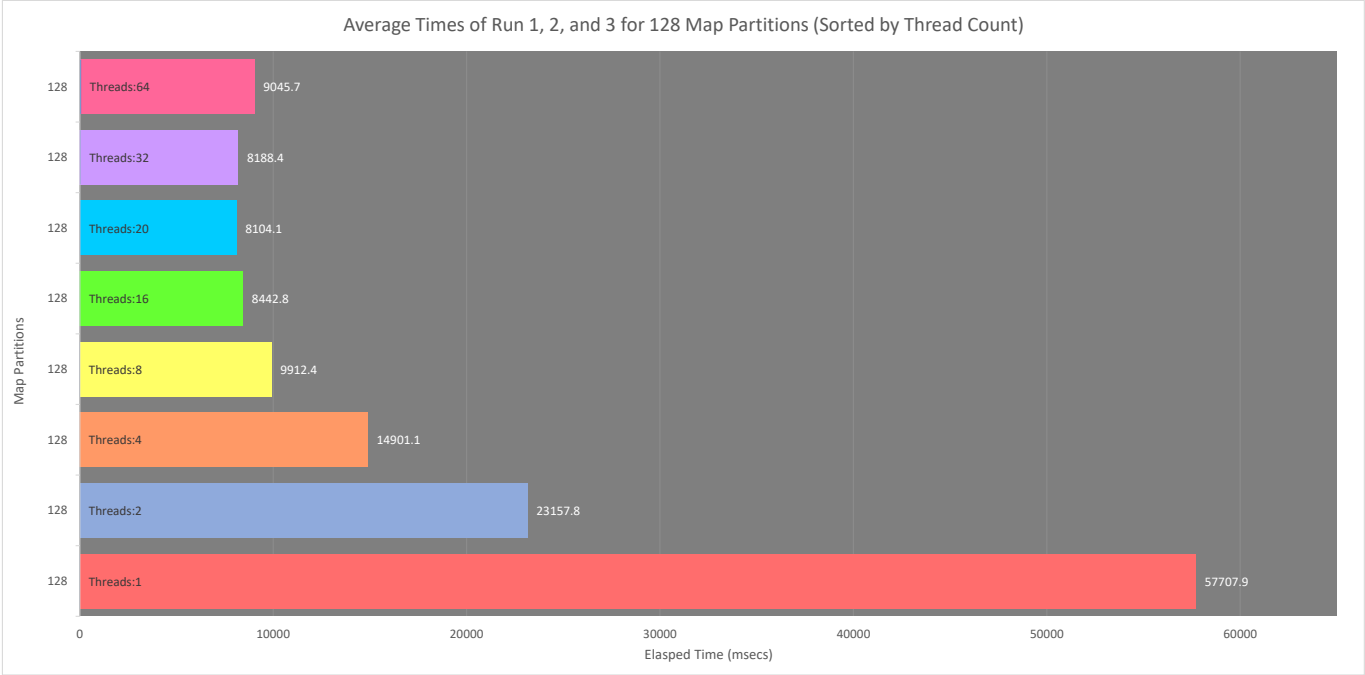
My idea was that *pmap*, a parallelized version *map*, which applies a function to each set of a collection, would assign each partition to a new thread. However, there is no way to guarantee that *pmap* creates the same number of threads as the partitions. Furthermore, *pmap* starts with two threads and proceeds to utilize available threads as needed. Preliminary tests suggested that the size of the map partition affected the execution time considerably. After modifying the source code for *pmap*, I did not know what number of partitions to choose in order to optimize the speed, so therefore I tested a range of map partitions from a single partition up to 10,000 partitions, or a partition for each entry in the hash map. One thing to note, if only one thread is used, the standard non-parallelized *map* function is used instead of *modded-pmap*. This is due to the way *modded-pmap* works, which requires two or more threads.

The *doseq* function allows the test to iterate through each number of threads and each number of partitions in one program execution. The first part initializes the PageRanks to 1, and partitions the hash map into *s* partitions, *mapSplits*, returning a sequence of hash maps, *mapPartitions*. After this, the loop runs for 999 more times, redefining the *pageRankMap* object during each iteration with updated values.

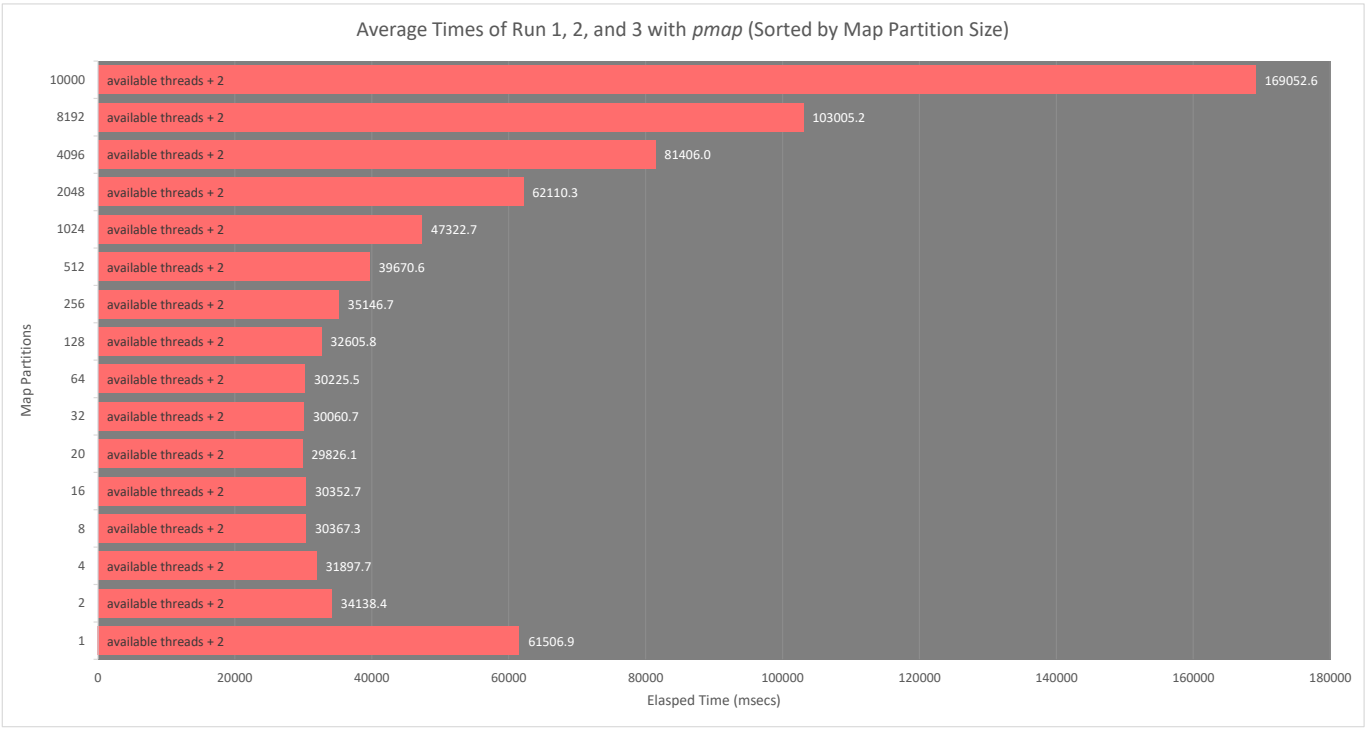
```
(defn main-body [myMap]
  (def inpagesMap (find-inpages myMap))
  (def outpagesCountMap (count-outpages myMap))
  (def threads [1 2 4 8 16 20 32 64])
  (def mapSplits [1 2 4 8 16 20 32 64 128 256 512 1024 2048 4096 8192 10000])

  (doseq [s mapSplits]
    (doseq [t threads]
      (save-times t s (with-out-str (time
                          (do
                           (def mapPartitions (map vec-to-map (split-map s inpagesMap))) ;; returns a sequence of hash maps
                           (if (= t 1)
                            (def pageRankMap (combine-maps (map #(calc-page-rank % outpagesCountMap (start-rank
                            myMap 1)) mapPartitions))) ;; initialize PageRanks to 1, standard map function for 1 thread
                            ;; else
                            (def pageRankMap (combine-maps (modded-pmap #(calc-page-rank %  outpagesCountMap (start-
                            rank myMap 1)) t mapPartitions))))) ;; initialize PageRanks to 1, modded pmap for > 1 thread
                           (save-results pageRankMap 1 false) ;; set to true if output of PageRanks is needed

                           (loop [i 1] ;; already run once above
                            (if (< i 1000)
                             (do
                              (if (= t 1)
                               (def pageRankMap (combine-maps (map #(calc-page-rank % outpagesCountMap pageRankMap)
                               mapPartitions))) ;; standard map function for 1 thread
                               ;; else
                               (def pageRankMap (combine-maps (modded-pmap #(calc-page-rank %  outpagesCountMap
                               pageRankMap) t mapPartitions)))) ;; modded pmap for > 1 thread
                              (recur (+ i 1)))
                             (save-results pageRankMap i false)))))))))) ;; set to true if output of PageRanks is needed
(shutdown-agents)) ;; releases threads
```

Findings

For the timing study, the multiple thread counts and map partitions are compared. As expected, an increase in the number of threads decreased the execution time of the PageRank algorithm up until the number of physical threads is exceeded. This study was conducted on a machine with an Intel i9 7900X with 20 logical processors, the reason why a thread count of 20 is included in the tests. Requesting more threads than are physically available generally leads to a slight increase in execution time.



With 128 map partitions, 20 threads, corresponding to the number of physical threads available, performs the best at 7.12 times faster than a single thread, with an average time of just 8.10 seconds. By contrast, it takes a single thread 57.70 seconds to execute. Requesting 32 threads has a minor impact on execution time. Requesting 64 threads, however, is about 0.93 times the speed of using 16 threads, or about 600 msecs slower. It is interesting that 2 threads are about 2.5 times faster than using a single thread, but 4 threads are only about 1.55 times faster than 2 threads. The execution time does not scale linearly with the thread count. The complete table of Run 1, 2, and 3 are available on page 7.



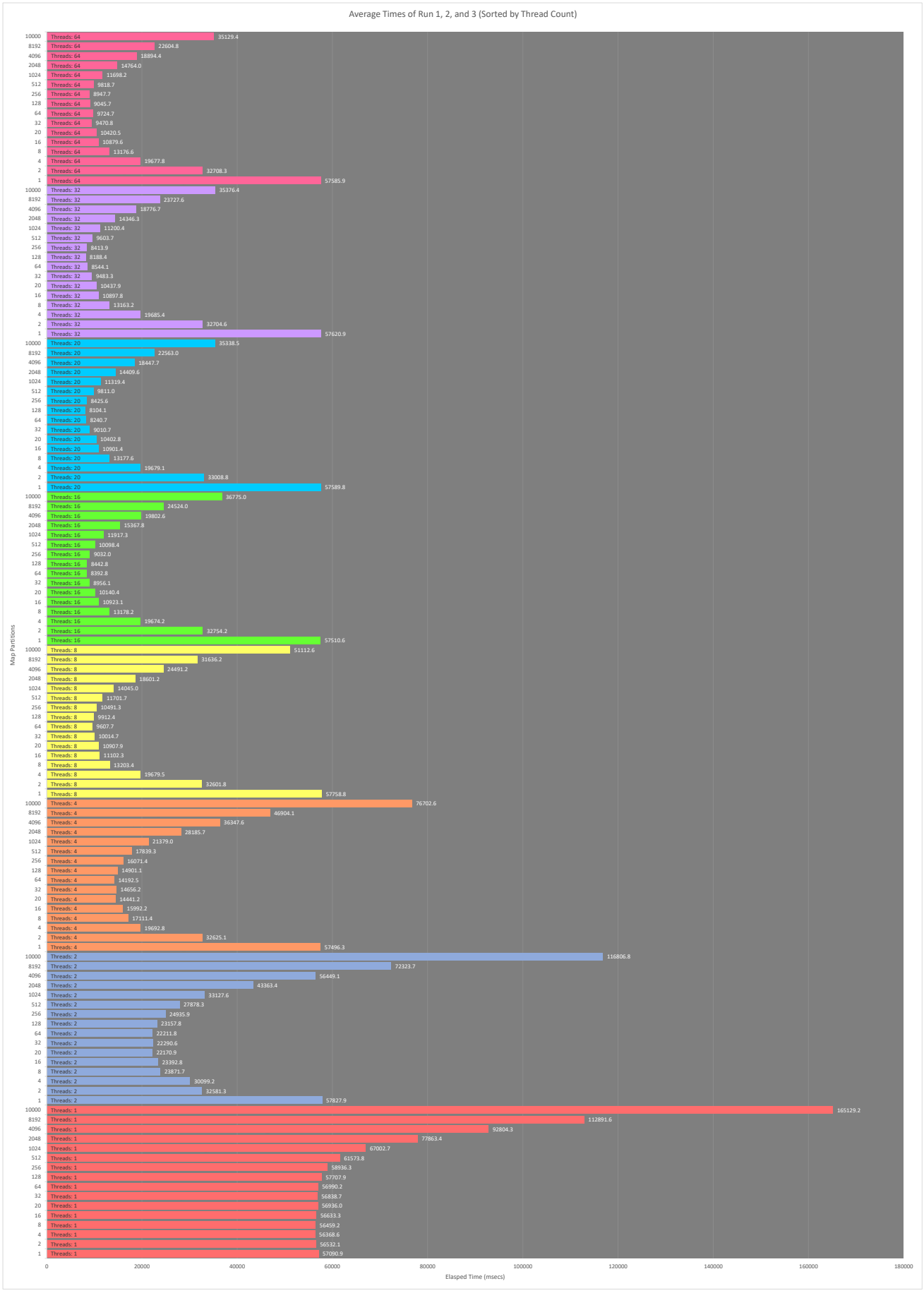
A further comparison of modded-pmap to pmap shows a substantial increase in the efficiency of modded-pmap. Although the number of threads pmap used was not recorded, with 128 map partitions, pmap is 0.77 times the speed of modded-pmap using only 2 threads.

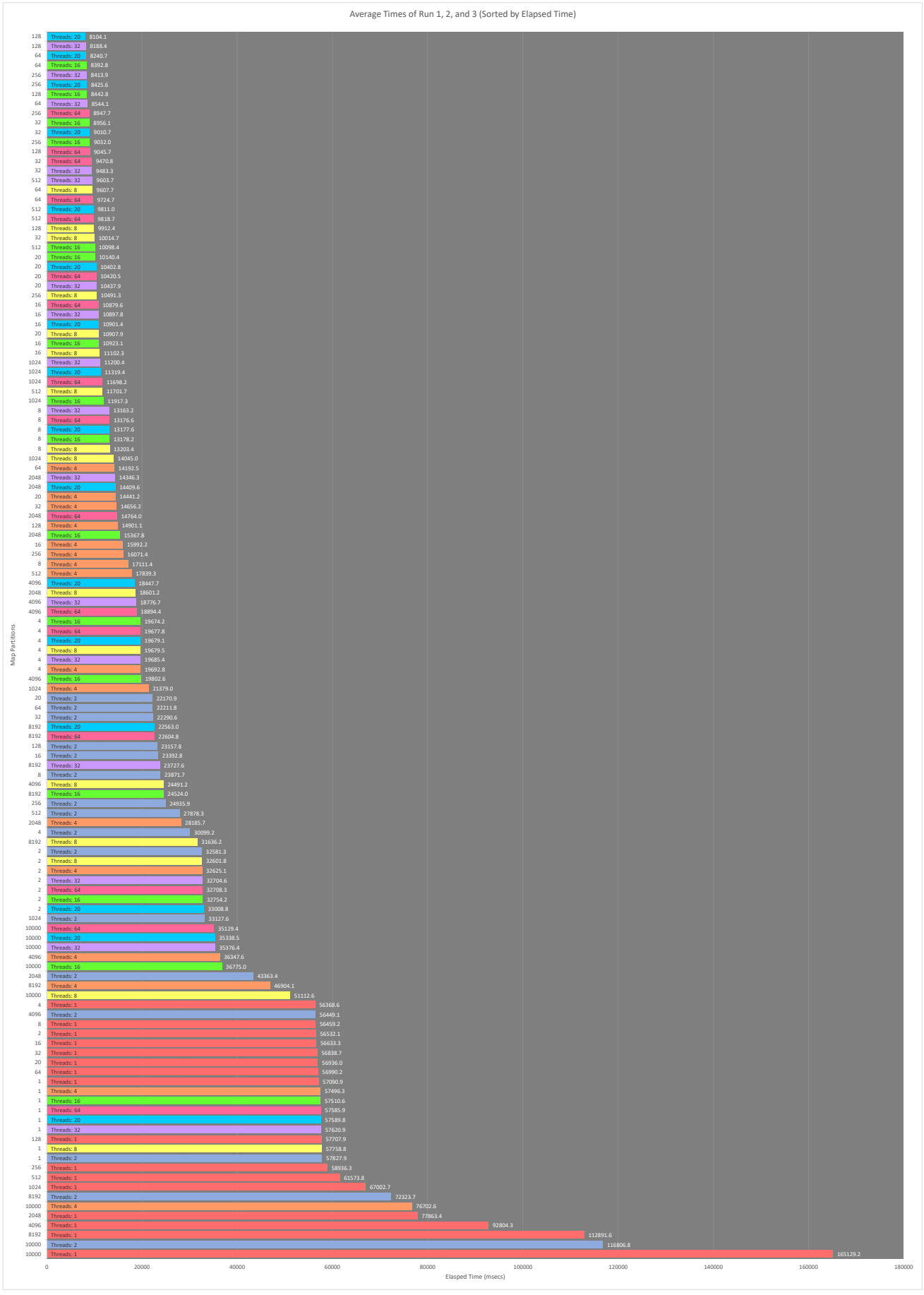
Perhaps more interesting is the effect of the number of map partitions that are used. The charts for the average times of Run 1, 2, and 3 sorted by thread count and execution time are available on pages 5 and 6, respectively.

The optimal number of map partitions appears to 64 or 128, with the fastest time reached by 20 threads and 128 map partitions at 8104.1 msecs. Using 20 threads, this is 7.11 times faster than passing the entire map in a single partition, and 4.36 times faster than passing 10,000 partitions, or each entry as an individual map. The slowest by far is using a single thread with 10,000 partitions, which is 20.38 times slower than the fastest time. The number of threads using a single map partition has no apparent effect on the execution time. This is because the map is not divided up amongst the threads. Likewise, passing 1 thread multiple map partitions has a neutral effect on execution time up to 128 map partitions, or 64 partitions for 2, 4, and 8 threads, respectively. Excluding the 1 thread count, as the number of map partitions increase, the execution time is reduced considerably up to 64 or 128 map partitions, depending on the number of threads. From 256 to 10,000 map partitions, the execution times slows considerably.

Conclusion

The results show a clear relationship between the number of threads and map partitions and execution time. As the number of threads increases up to the number of physical threads, the execution time decreases. As soon as the number of physical threads is reached, a slight increase in execution time is observed. Similarly, increasing the number of map partitions past the 64 or 128 thresholds, depending on the thread count, execution time decreases. This is where the data varies per thread count, and there is no specific multiplier relating the number of available threads to the number of map partitions. However, the results demonstrate the importance of multithreading. Furthermore, they indicate that partitioning data into smaller blocks can optimize execution time considerably. However, it is important that a balance is found between threads and partitions.


Average Times of Run 1, 2, and 3 (Sorted by Thread Count)
Map Partitions
Elasped Time (msecs)
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
10000 Threads: 64 35129.4
8192 Threads: 64 22604.8
4096 Threads: 64 18894.4
2048 Threads: 64 14764.0
1024 Threads: 64 11698.2
512 Threads: 64 9818.7
256 Threads: 64 8947.7
128 Threads: 64 9045.7
64 Threads: 64 9724.7
32 Threads: 64 9470.8
20 Threads: 64 10420.5
16 Threads: 64 10879.6
8 Threads: 64 13176.6
4 Threads: 64 19677.8
2 Threads: 64 32708.3
1 Threads: 64 57585.9
10000 Threads: 32 35376.4
8192 Threads: 32 23727.6
4096 Threads: 32 18776.7
2048 Threads: 32 14346.3
1024 Threads: 32 11200.4
512 Threads: 32 9603.7
256 Threads: 32 8413.9
128 Threads: 32 8188.4
64 Threads: 32 8544.1
32 Threads: 32 9483.3
20 Threads: 32 10437.9
16 Threads: 32 10897.8
8 Threads: 32 13163.2
4 Threads: 32 19685.4
2 Threads: 32 32704.6
1 Threads: 32 57620.9
10000 Threads: 20 35338.5
8192 Threads: 20 22563.0
4096 Threads: 20 18447.7
2048 Threads: 20 14409.6
1024 Threads: 20 11319.4
512 Threads: 20 9811.0
256 Threads: 20 8425.6
128 Threads: 20 8104.1
64 Threads: 20 8240.7
32 Threads: 20 9010.7
20 Threads: 20 10402.8
16 Threads: 20 10901.4
8 Threads: 20 13177.6
4 Threads: 20 19679.1
2 Threads: 20 33008.8
1 Threads: 20 57589.8
10000 Threads: 16 36775.0
8192 Threads: 16 24524.0
4096 Threads: 16 19802.6
2048 Threads: 16 15367.8
1024 Threads: 16 11917.3
512 Threads: 16 10098.4
256 Threads: 16 9032.0
128 Threads: 16 8442.8
64 Threads: 16 8392.8
32 Threads: 16 8956.1
20 Threads: 16 10140.4
16 Threads: 16 10923.1
8 Threads: 16 13178.2
4 Threads: 16 19674.2
2 Threads: 16 32754.2
1 Threads: 16 57510.6
10000 Threads: 8 51112.6
8192 Threads: 8 31636.2
4096 Threads: 8 24491.2
2048 Threads: 8 18601.2
1024 Threads: 8 14045.0
512 Threads: 8 11701.7
256 Threads: 8 10491.3
128 Threads: 8 9912.4
64 Threads: 8 9607.7
32 Threads: 8 10014.7
20 Threads: 8 10907.9
16 Threads: 8 11102.3
8 Threads: 8 13203.4
4 Threads: 8 19679.5
2 Threads: 8 32601.8
1 Threads: 8 57758.8
10000 Threads: 4 76702.6
8192 Threads: 4 46904.1
4096 Threads: 4 36347.6
2048 Threads: 4 28185.7
1024 Threads: 4 21379.0
512 Threads: 4 17839.3
256 Threads: 4 16071.4
128 Threads: 4 14901.1
64 Threads: 4 14192.5
32 Threads: 4 14656.2
20 Threads: 4 14441.2
16 Threads: 4 15992.2
8 Threads: 4 17111.4
4 Threads: 4 19692.8
2 Threads: 4 32625.1
1 Threads: 4 57496.3
10000 Threads: 2 116806.8
8192 Threads: 2 72323.7
4096 Threads: 2 56449.1
2048 Threads: 2 43363.4
1024 Threads: 2 33127.6
512 Threads: 2 27878.3
256 Threads: 2 24935.9
128 Threads: 2 23157.8
64 Threads: 2 22211.8
32 Threads: 2 22290.6
20 Threads: 2 22170.9
16 Threads: 2 23392.8
8 Threads: 2 23871.7
4 Threads: 2 30099.2
2 Threads: 2 32581.3
1 Threads: 2 57827.9
10000 Threads: 1 165129.2
8192 Threads: 1 112891.6
4096 Threads: 1 92804.3
2048 Threads: 1 77863.4
1024 Threads: 1 67002.7
512 Threads: 1 61573.8
256 Threads: 1 58936.3
128 Threads: 1 57707.9
64 Threads: 1 56990.2
32 Threads: 1 56838.7
20 Threads: 1 56936.0
16 Threads: 1 56633.3
8 Threads: 1 56459.2
4 Threads: 1 56368.6
2 Threads: 1 56532.1
1 Threads: 1 57090.9


Average Times of Run 1, 2, and 3 (Sorted by Elapsed Time)
Map Partitions
Elasped Time (msecs)
0
20000
40000
60000
80000
100000
120000
140000
160000
180000
128 Threads: 20 8104.1
128 Threads: 32 8188.4
64 Threads: 20 8240.7
64 Threads: 16 8392.8
256 Threads: 32 8413.9
256 Threads: 20 8425.6
128 Threads: 16 8442.8
64 Threads: 32 8544.1
256 Threads: 64 8947.7
32 Threads: 16 8956.1
32 Threads: 20 9010.7
256 Threads: 16 9032.0
128 Threads: 64 9045.7
32 Threads: 64 9470.8
32 Threads: 32 9483.3
512 Threads: 32 9603.7
64 Threads: 8 9607.7
64 Threads: 64 9724.7
512 Threads: 20 9811.0
512 Threads: 64 9818.7
128 Threads: 8 9912.4
32 Threads: 8 10014.7
512 Threads: 16 10098.4
20 Threads: 16 10140.4
20 Threads: 20 10402.8
20 Threads: 64 10420.5
20 Threads: 32 10437.9
256 Threads: 8 10491.3
16 Threads: 64 10879.6
16 Threads: 32 10897.8
16 Threads: 20 10901.4
20 Threads: 8 10907.9
16 Threads: 16 10923.1
16 Threads: 8 11102.3
1024 Threads: 32 11200.4
1024 Threads: 20 11319.4
1024 Threads: 64 11698.2
512 Threads: 8 11701.7
1024 Threads: 16 11917.3
8 Threads: 32 13163.2
8 Threads: 64 13176.6
8 Threads: 20 13177.6
8 Threads: 16 13178.2
8 Threads: 8 13203.4
1024 Threads: 8 14045.0
64 Threads: 4 14192.5
2048 Threads: 32 14346.3
2048 Threads: 20 14409.6
20 Threads: 4 14441.2
32 Threads: 4 14656.2
2048 Threads: 64 14764.0
128 Threads: 4 14901.1
2048 Threads: 16 15367.8
16 Threads: 4 15992.2
256 Threads: 4 16071.4
8 Threads: 4 17111.4
512 Threads: 4 17839.3
4096 Threads: 20 18447.7
2048 Threads: 8 18601.2
4096 Threads: 32 18776.7
4096 Threads: 64 18894.4
4 Threads: 16 19674.2
4 Threads: 64 19677.8
4 Threads: 20 19679.1
4 Threads: 8 19679.5
4 Threads: 32 19685.4
4 Threads: 4 19692.8
4096 Threads: 16 19802.6
1024 Threads: 4 21379.0
20 Threads: 2 22170.9
64 Threads: 2 22211.8
32 Threads: 2 22290.6
8192 Threads: 20 22563.0
8192 Threads: 64 22604.8
128 Threads: 2 23157.8
16 Threads: 2 23392.8
8192 Threads: 32 23727.6
8 Threads: 2 23871.7
4096 Threads: 8 24491.2
8192 Threads: 16 24524.0
256 Threads: 2 24935.9
512 Threads: 2 27878.3
2048 Threads: 4 28185.7
4 Threads: 2 30099.2
8192 Threads: 8 31636.2
2 Threads: 2 32581.3
2 Threads: 8 32601.8
2 Threads: 4 32625.1
2 Threads: 32 32704.6
2 Threads: 64 32708.3
2 Threads: 16 32754.2
2 Threads: 20 33008.8
1024 Threads: 2 33127.6
10000 Threads: 64 35129.4
10000 Threads: 20 35338.5
10000 Threads: 32 35376.4
4096 Threads: 4 36347.6
10000 Threads: 16 36775.0
2048 Threads: 2 43363.4
8192 Threads: 4 46904.1
10000 Threads: 8 51112.6
4 Threads: 1 56368.6
4096 Threads: 2 56449.1
8 Threads: 1 56459.2
2 Threads: 1 56532.1
16 Threads: 1 56633.3
32 Threads: 1 56838.7
20 Threads: 1 56936.0
64 Threads: 1 56990.2
1 Threads: 1 57090.9
1 Threads: 4 57496.3
1 Threads: 16 57510.6
1 Threads: 64 57585.9
1 Threads: 20 57589.8
1 Threads: 32 57620.9
128 Threads: 1 57707.9
1 Threads: 8 57758.8
1 Threads: 2 57827.9
256 Threads: 1 58936.3
512 Threads: 1 61573.8
1024 Threads: 1 67002.7
8192 Threads: 2 72323.7
10000 Threads: 4 76702.6
2048 Threads: 1 77863.4
4096 Threads: 1 92804.3
8192 Threads: 1 112891.6
10000 Threads: 2 116806.8
10000 Threads: 1 165129.2

Run 1 - Start: Fri Nov 22 23:54:15 CST 2019		
Threads	Map Partitions	Time (msecs)
1	1	56818.5413
2	1	57876.0984
4	1	57761.8118
8	1	57906.6139
16	1	57400.9371
20	1	57787.4894
32	1	57614.0248
64	1	57766.3975
1	2	56539.8151
2	2	32564.4376
4	2	32587.2957
8	2	32602.4588
16	2	33038.6025
20	2	33680.0089
32	2	32774.9134
64	2	32712.332
1	4	56577.2214
2	4	30042.3978
4	4	19572.6529
8	4	19549.2045
16	4	19583.2531
20	4	19593.512
32	4	19567.9733
64	4	19577.8179
1	8	56582.4736
2	8	23864.0983
4	8	17035.1532
8	8	13158.8581
16	8	13126.7624
20	8	13106.3963
32	8	13105.336
64	8	13124.8046
1	16	56660.0755
2	16	23386.0112
4	16	15943.2312
8	16	11111.9694
16	16	10904.1376
20	16	10879.6714
32	16	10863.1482
64	16	10873.5953
1	20	57265.5827
2	20	22020.5098
4	20	14410.2507
8	20	10936.771
16	20	10064.6874
20	20	10369.7542
32	20	10383.565
64	20	10385.2663
1	32	56855.4227
2	32	21789.2544
4	32	14499.9166
8	32	10107.1431
16	32	8968.9209
20	32	8991.8654
32	32	9439.6042
64	32	9467.997
1	64	57440.2913
2	64	22134.7644
4	64	14180.5418
8	64	9683.9229
16	64	8404.1106
20	64	8298.2011
32	64	8542.985
64	64	9763.1933
1	128	58148.682
2	128	23045.9168
4	128	14830.2167
8	128	10004.8428
16	128	8498.4267
20	128	8165.4794
32	128	8294.1454
64	128	9016.7698
1	256	59147.569
2	256	24888.4444
4	256	16000.8508
8	256	10419.3111
16	256	8973.059
20	256	8313.9797
32	256	8346.6238
64	256	8872.9334
1	512	61281.6498
2	512	27745.4776
4	512	17721.5363
8	512	11538.5272
16	512	9927.7059
20	512	9666.9485
32	512	9493.2956
64	512	9640.0598
1	1024	66653.0724
2	1024	32450.8202
4	1024	20943.1016
8	1024	13684.7077
16	1024	11655.1354
20	1024	11033.0116
32	1024	10958.1237
64	1024	11502.0297
1	2048	76985.2053
2	2048	43027.6207
4	2048	27055.3153
8	2048	17820.5334
16	2048	14856.5959
20	2048	13844.1012
32	2048	13956.1684
64	2048	14541.7633
1	4096	91176.6958
2	4096	55119.7662
4	4096	35109.9173
8	4096	23377.6848
16	4096	18897.9297
20	4096	17623.2914
32	4096	17958.1562
64	4096	18398.4612
1	8192	109056.3129
2	8192	70555.6859
4	8192	44205.9219
8	8192	29521.2929
16	8192	23122.9672
20	8192	21404.0925
32	8192	22254.8898
64	8192	21569.1901
1	10000	159196.7236
2	10000	112951.1719
4	10000	72081.1693
8	10000	47184.2852
16	10000	34215.663
20	10000	32881.9294
32	10000	32991.1631
64	10000	32752.6377
Run 1 - End: Sat Nov 23 00:56:49 CST 2019		

Run 2 - Start: Sat Nov 23 06:18:55 CST 2019		
Threads	Map Partitions	Time (msecs)
1	1	58820.9254
2	1	59779.9896
4	1	58998.49
8	1	59104.2722
16	1	59067.3694
20	1	59017.1392
32	1	59289.6294
64	1	59065.5071
1	2	57954.9206
2	2	33492.7758
4	2	33559.498
8	2	33499.6007
16	2	33494.5977
20	2	33612.4929
32	2	33514.986
64	2	33515.1019
1	4	57601.8101
2	4	30932.1541
4	4	20207.9263
8	4	20201.7799
16	4	20206.6901
20	4	20185.0979
32	4	20198.7227
64	4	20183.3612
1	8	57655.2835
2	8	24587.8128
4	8	17666.0527
8	8	13685.7497
16	8	13661.026
20	8	13665.6448
32	8	13648.6812
64	8	13657.184
1	16	58133.8761
2	16	24160.4878
4	16	16508.3458
8	16	11416.6143
16	16	11157.9685
20	16	11117.6464
32	16	11140.3457
64	16	11095.0222
1	20	58300.462
2	20	22933.1204
4	20	14903.6479
8	20	11244.7865
16	20	10343.569
20	20	10621.1157
32	20	10638.3152
64	20	10662.4257
1	32	58219.077
2	32	23147.5009
4	32	15235.7498
8	32	10338.6542
16	32	9148.8389
20	32	9183.8922
32	32	9687.2878
64	32	9673.8981
1	64	58238.7745
2	64	22957.2472
4	64	14629.0185
8	64	9938.2156
16	64	8583.0887
20	64	8435.0388
32	64	8714.0242
64	64	9913.9242
1	128	58955.1111
2	128	24303.3227
4	128	15398.4631
8	128	10250.7821
16	128	8661.3893
20	128	8356.1951
32	128	8376.07
64	128	9249.5684
1	256	60338.6726
2	256	26240.7823
4	256	16748.6211
8	256	11002.5768
16	256	9382.3739
20	256	8754.7072
32	256	8714.0787
64	256	9224.0834
1	512	63468.1704
2	512	28934.2652
4	512	18609.8317
8	512	12396.2177
16	512	10557.8922
20	512	10200.6215
32	512	10046.2925
64	512	10276.3793
1	1024	69811.8413
2	1024	35143.0147
4	1024	22632.6796
8	1024	15168.4662
16	1024	12625.3202
20	1024	12036.4907
32	1024	11904.1375
64	1024	12457.626
1	2048	81945.172
2	2048	45396.3752
4	2048	30616.5205
8	2048	20684.2454
16	2048	16576.8542
20	2048	15714.5762
32	2048	15539.6366
64	2048	15837.4628
1	4096	98517.7121
2	4096	60519.9011
4	4096	39474.1804
8	4096	27413.3709
16	4096	21883.2226
20	4096	20488.2274
32	4096	20646.9641
64	4096	20391.92
1	8192	119638.7332
2	8192	77023.8394
4	8192	50521.5757
8	8192	35752.7258
16	8192	27724.2716
20	8192	25541.5311
32	8192	26222.0819
64	8192	24945.5508
1	10000	180485.1256
2	10000	125368.1214
4	10000	86520.1671
8	10000	60331.5691
16	10000	42727.0161
20	10000	40742.2458
32	10000	41125.874
64	10000	40175.4263
Run 2 - End: Sat Nov 23 07:26:01 CST 2019		

Run 3 - Start: Sat Nov 23 11:39:44 CST 2019		
Threads	Map Partitions	Time (msecs)
1	1	55633.2712
2	1	55827.5663
4	1	55728.7273
8	1	56265.6465
16	1	56063.5713
20	1	55964.8566
32	1	55958.9497
64	1	55925.8527
1	2	55101.4196
2	2	31686.6996
4	2	31728.4353
8	2	31703.3642
16	2	31729.3806
20	2	31733.7892
32	2	31824.0299
64	2	31897.538
1	4	54926.6555
2	4	29323.0203
4	4	19297.7076
8	4	19287.625
16	4	19232.8029
20	4	19258.5986
32	4	19289.619
64	4	19272.2981
1	8	55139.8512
2	8	23163.1725
4	8	16632.9019
8	8	12765.6436
16	8	12746.6819
20	8	12760.82
32	8	12735.5101
64	8	12747.7493
1	16	55106.0565
2	16	22631.9866
4	16	15524.8803
8	16	10778.4164
16	16	10707.1147
20	16	10706.8261
32	16	10690.0304
64	16	10670.0897
1	20	55241.9837
2	20	21559.1272
4	20	14009.7643
8	20	10542.0553
16	20	10012.834
20	20	10217.4065
32	20	10291.7488
64	20	10213.9121
1	32	55441.5059
2	32	21935.0077
4	32	14232.977
8	32	9598.2322
16	32	8750.5607
20	32	8856.4142
32	32	9323.0881
64	32	9270.3641
1	64	55291.5453
2	64	21543.4456
4	64	13767.8691
8	64	9200.8966
16	64	8191.1938
20	64	7988.7658
32	64	8375.2633
64	64	9497.0515
1	128	56019.8911
2	128	22124.1996
4	128	14474.5534
8	128	9481.4921
16	128	8168.6874
20	128	7790.4929
32	128	7894.9946
64	128	8870.8631
1	256	57322.5793
2	256	23678.4109
4	256	15464.7888
8	256	10052.0302
16	256	8740.4694
20	256	8208.1828
32	256	8181.1365
64	256	8745.9451
1	512	59971.5328
2	512	26955.1629
4	512	17186.406
8	512	11170.3456
16	512	9809.7163
20	512	9565.4546
32	512	9271.4707
64	512	9539.6507
1	1024	64543.275
2	1024	31788.872
4	1024	20561.1291
8	1024	13281.8867
16	1024	11471.3139
20	1024	10888.7674
32	1024	10739.0137
64	1024	11134.8207
1	2048	74659.7809
2	2048	41666.3046
4	2048	26885.2857
8	2048	17298.9669
16	2048	14669.8489
20	2048	13670.2456
32	2048	13543.1586
64	2048	13912.8909
1	4096	88718.3901
2	4096	53707.5717
4	4096	34458.852
8	4096	22682.4207
16	4096	18626.7714
20	4096	17231.7253
32	4096	17724.9504
64	4096	17892.9119
1	8192	109979.7421
2	8192	69391.4719
4	8192	45984.6915
8	8192	29634.4394
16	8192	22724.6553
20	8192	20743.5225
32	8192	22705.9591
64	8192	21299.708
1	10000	155705.6035
2	10000	112101.0743
4	10000	71506.4248
8	10000	45821.8701
16	10000	33382.4291
20	10000	32391.1901
32	10000	32012.1016
64	10000	32460.0146
Run 3 - End: Sat Nov 23 12:40:51 CST 2019		

Averages of Runs 1,2 and 3		
Threads	Map Partitions	Time (msecs)
1	1	57090.91263
2	1	57827.88477
4	1	57496.34303
8	1	57758.8442
16	1	57510.62593
20	1	57589.8284
32	1	57620.86797
64	1	57585.9191
1	2	56532.05177
2	2	32581.30433
4	2	32625.07633
8	2	32601.8079
16	2	32754.1936
20	2	33008.76367
32	2	32704.6431
64	2	32708.32397
1	4	56368.56233
2	4	30099.19073
4	4	19692.76227
8	4	19679.53647
16	4	19674.2487
20	4	19679.0695
32	4	19685.43833
64	4	19677.82573
1	8	56459.20277
2	8	23871.69453
4	8	17111.36927
8	8	13203.41713
16	8	13178.15677
20	8	13177.62037
32	8	13163.17577
64	8	13176.5793
1	16	56633.33603
2	16	23392.82853
4	16	15992.15243
8	16	11102.33337
16	16	10923.0736
20	16	10901.3813
32	16	10897.84143
64	16	10879.56907
1	20	56936.00947
2	20	22170.91913
4	20	14441.22097
8	20	10907.87093
16	20	10140.36347
20	20	10402.7588
32	20	10437.87633
64	20	10420.5347
1	32	56838.66853
2	32	22290.58767
4	32	14656.21447
8	32	10014.6765
16	32	8956.106833
20	32	9010.723933
32	32	9483.3267
64	32	9470.753067
1	64	56990.2037
2	64	22211.81907
4	64	14192.47647
8	64	9607.678367
16	64	8392.7977
20	64	8240.668567
32	64	8544.090833
64	64	9724.723
1	128	57707.89473
2	128	23157.81303
4	128	14901.07773
8	128	9912.372333
16	128	8442.834467
20	128	8104.0558
32	128	8188.403333
64	128	9045.733767
1	256	58936.27363
2	256	24935.8792
4	256	16071.42023
8	256	10491.30603
16	256	9031.967433
20	256	8425.623233
32	256	8413.946333
64	256	8947.653967
1	512	61573.78433
2	512	27878.3019
4	512	17839.258
8	512	11701.69683
16	512	10098.43813
20	512	9811.0082
32	512	9603.686267
64	512	9818.6966
1	1024	67002.72957
2	1024	33127.56897
4	1024	21378.9701
8	1024	14045.0202
16	1024	11917.2565
20	1024	11319.42323
32	1024	11200.42497
64	1024	11698.1588
1	2048	77863.38607
2	2048	43363.4335
4	2048	28185.70717
8	2048	18601.24857
16	2048	15367.76633
20	2048	14409.641
32	2048	14346.3212
64	2048	14764.039
1	4096	92804.266
2	4096	56449.07967
4	4096	36347.6499
8	4096	24491.1588
16	4096	19802.64123
20	4096	18447.74803
32	4096	18776.69023
64	4096	18894.43103
1	8192	112891.5961
2	8192	72323.66573
4	8192	46904.06303
8	8192	31636.1527
16	8192	24523.9647
20	8192	22563.0487
32	8192	23727.6436
64	8192	22604.8163
1	10000	165129.1509
2	10000	116806.7892
4	10000	76702.58707
8	10000	51112.5748
16	10000	36775.03607
20	10000	35338.4551
32	10000	35376.37957
64	10000	35129.35953