System setup specification

Hardware Specs	16g ram, 150g Flash Storage, 2.2 Ghz, Intel core i7
Index Cluster Setup	Distributed (Solr Cloud). Version 6.2.1
Spark Cluster	Spark Version 2.0.0, Scala Version 2.11.8, Java 1.8.0_31, Standalone Cluster. 1 Master 12g, 2 Workers 8g

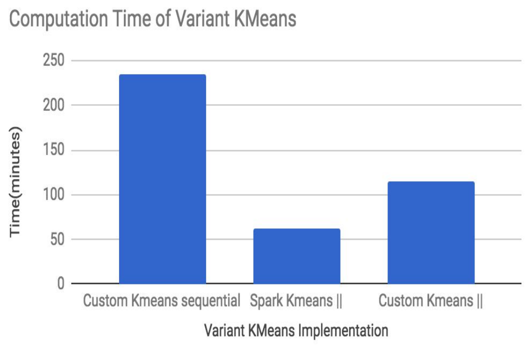
Experimental Analysis

For experiment analysis, metrics considered are in terms efficiency and correctness of clustering algorithm and run time analysis. Cluster quality analysis by applying clustering related evaluation metrics and selective search effectiveness measure is still work in progress.

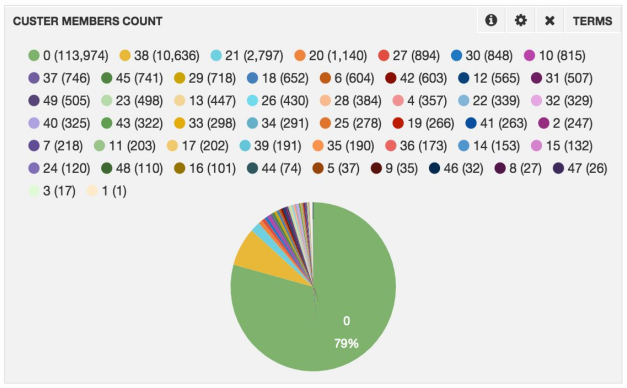
Table 1: Comparison of Custom k-means sequence & parallel and Spark k-means parallel Clueweb09 dataset for specific number of records : 143,861

K=50, Iterations = 25, Features = 25000, Total unique terms = 859583

Records count	Custom Kmeans sequential	Spark Kmeans ||	Custom Kmeans ||
143,861	3 hours 55 min	1 hour 2 min	1 hour 55 min



Cluster allocation, Chart 1 Spark Means || vs



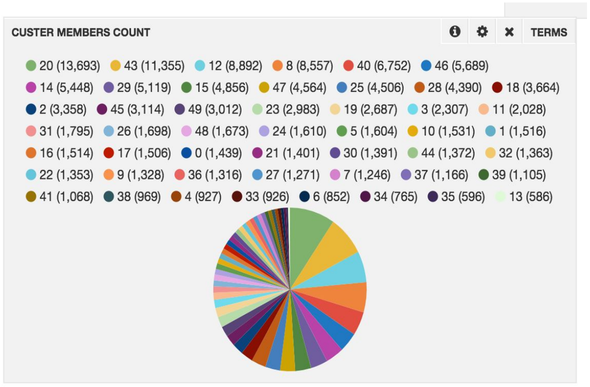


Chart 2 Custom Kmean ||

Table 2

Comparing Spark k-means parallel with Custom k-means parallel

Clueweb09 dataset for specific number of records : 143,861

K=50, Iterations = 10, Features = 25000, Total unique terms = 859583

Records (N) 143861	Warc File Streaming	Text analysis, TF DF, IDF with TF-IDF weights calculation	Vectorize, collect vectors for KMean	KMean	Solr Indexin g	Total
Spark k-means || MLLib	37 seconds	7 minutes, 25 seconds	3 minutes, 48 seconds.	1 hour, 4 minutes, 8 seconds	4 minutes, 51 seconds	1 hour 21 minutes
Custom k-means ||	35 seconds	7 minutes, 45 seconds	3 minutes, 48 seconds	1 hours, 39 minutes, 36 seconds	5 minutes, 47 seconds	1 hour 56 minutes

Table 3

Custom k-means parallel computation time for Clueweb09 dataset.

Number of records : 201,002

K=50, Iterations = 10, Features = 10000, Total unique terms = 859583

Records (N) 201,002	Warc File Streaming	Text analysis, TF DF, IDF with TF-IDF weights calculation	Vectorize, collect vectors for KMean	KMean	Solr Indexin g	Total

Custom k-means ||	56 seconds	10 minutes, 25 seconds	4 minutes, 58 seconds.	2 hours, 17 minutes, 8 seconds	8 minutes, 02 seconds	2 hours 42 minutes

Table 4
Custom k-means parallel total computation time for various dataset size
Clueweb09 dataset.

Total Docs	Total Time	KMean Inputs
644	14 seconds	F=5000, Clusters=5, Iterations = 10
3423	31 seconds	F=5000, Clusters=5, Iterations = 10
9143	1.3 minutes	F=8000, Clusters=10, Iterations = 10
32922	11 minutes	F=10000, Clusters=10, Iterations = 10
52319	16 minutes	F=10000, Clusters=10, Iterations = 10
81990	35 minutes	F=10000, Clusters=10, Iterations = 10
143861	1.82 hours	F=10000, Clusters=10, Iterations = 10
232204	2.2 hours	F=10000, Clusters=10, Iterations = 10

Chart 3
Total Time taken for a complete clustering job by specified number of documents

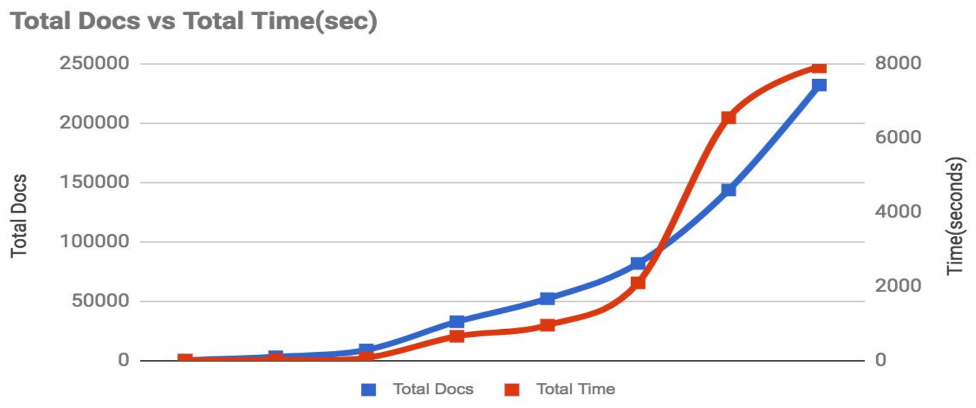


Table 5

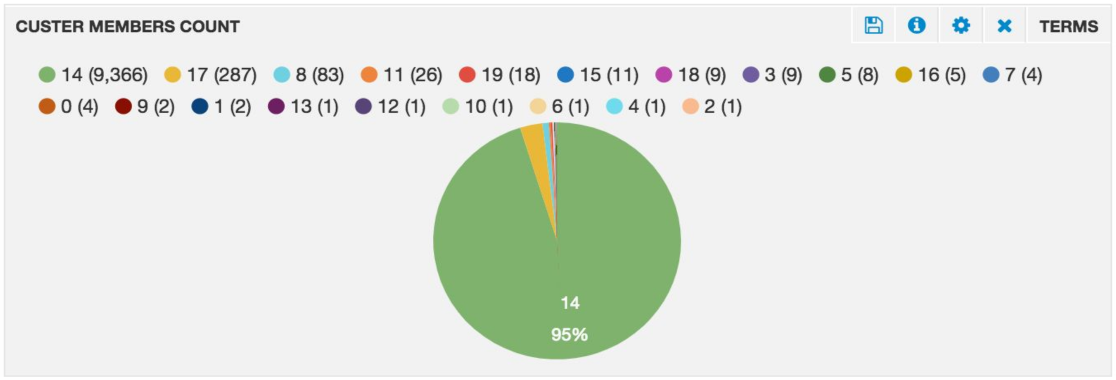
Comparing Spark k-means parallel with Custom k-means parallel

20Newsgroup Dataset, total number of records : 9840

K=20, Iterations = 10, Features = 15000, Total unique terms = 100455

20Newsgroup	Text analysis and TF, DF, IDF computation	KMean Clustering	Solr Index	Total Time
Spark k-means||	4 min 20 sec	25 sec	5 sec	4.8 min
Custom k-means||	4 min 12 sec	1 min 40 sec	9 sec	5.7 min

Cluster allocation of Spark k-means for 20Newsgroup



Cluster allocation of Custom k-means for 20Newsgroup

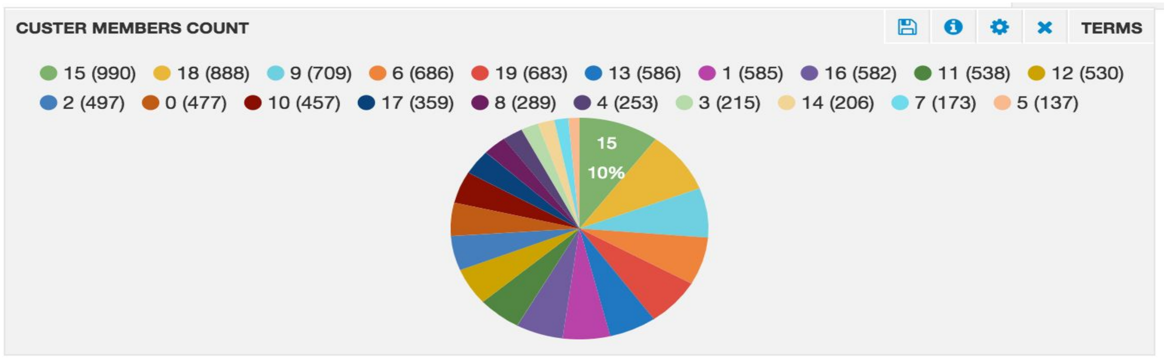


Table 6
Projection Phase. Part of Clueweb09 dataset.

Records count	Custom k-means ||
135,512	36 minutes

Results and Discussion

Statistics presented in Tables from 1 to 6 are experimental results of complete project execution in terms of memory and efficiency. Due to the limitation of hardware resources, KMean model was trained with limited dataset of size 210k records that consumed 16 gb of physical memory. Further, utilizing the same model, project phase was executed in batches and applied for over million records. On an average, for records of size over 150k, time spent during projection phase was over 36 minutes, as shown in Table 6.

At first, project utilized the spark's in-built HashingTF-IDF and k-means parallel library for vectorization and clustering respectively. Although this setup expend most minimal computation time of less than 30 minutes, resulting clusters were skewed, as seen in Chart 1. In the next setup, HashingTF-IDF was replaced with custom developed TF-IDF model that extended spark libraries, and tests were run with this new vectorizing model and Spark's k-means parallel. Results of this setup were as shown in Table 2, Row 2. Even with this setup, resulting clusters were skewed. Hence, a custom k-means was implemented, and initially it was sequential kmean implementation. Outcome of cluster quality was as desired for test with new vectorizing model and custom k-means, cluster results was as seen in chart 2, however, computation was unreasonable when compared to spark's k-means parallel. In further tests, custom k-means parallel was implemented and results of all these variants of k-means were tested out, computation time of each test is as displayed in Table 1. Vectorization model, file stream and indexing procedure were same for all the tests, except for k-means implementation. As it can been seen that, for dataset consisting over 150k records, custom k-means parallel expend additional of 40 minutes when compared to spark k-means parallel, nevertheless, the effectiveness of results are better for custom k-means. Table 4 with chart 3 presents time utilized by custom k-means for various sizes of dataset.

Table 5 presents comparison of spark k-means parallel over custom k-means parallel, for another type of dataset, 20 Newsgroups, it also exhibits resulting clusters allocation in pie chart 4 & 5.

Conclusions

Parallel k-means reduces computation time over more than 2x when compared to sequential k-means with indistinguishable cluster quality. Although spark k-means|| uses lesser time when compared to custom k-means||, cluster quality of spark k-means are not as desired. Spark k-means|| utilizes slightly different approach for merging records distributed over nodes and for centroid computation at each iteration, it also uses Euclidean distance for computing similarity between two records, those could be one of the potential reasons for skew cluster results. HashingTF-IDF model for vectorization is efficient, however, it results for potential collision of similar terms assigned to same index, and hence affecting cluster quality. Even though the vocabulary size of datasets experimented with, were 30x larger than dictionary terms used to train KMean model, resulting quality of clusters were desirably symmetric. Current work was focused on improving cluster quality and efficiency, implementation for selective search is yet to be done, further, search cost analysis and effectiveness of results are yet to be experimented and examined. Current work focused on organizing collection into topic shards with better cluster quality. Future work focus will be on deciding which index shards to search from the given set of shards, which is a type of resource selection problem[1, section 2], and also to conduct in-depth study of the tradeoff between search cost and search accuracy in a sharded index environment.