ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 10(38), DOI: 10.17485/ijst/2017/v10i38/114003, October 2017*

Fast and Effective Root cause Analysis of Streaming Data using In-Memory Processing Techniques

S. Naveen Kumar[1] and S. Vijayaragavan[2]

[1]Department of Computer Science (Category-B), Bharathiar University, Coimbatore – 641046, Tamil Nadu, India; naveen.seerangan@gmail.com
[2]Department of Computer Science and Engineering, Paavai Engineering College, Namakkal – 637018, Tamil Nadu, India; shanvijay@outlook.com

Abstract

Objectives: Increased data generation mandates a highly scalable and powerful processing framework for root cause analysis. The objective is to identify such a framework by analyzing the existing processing architectures. **Methods/Analysis:** In-order to identify the best processing architecture for root-cause analysis, the existing architectures are divided in terms of sequential processing using python, CPU based parallelization, Hadoop MapReduce and Spark based parallel in-memory processing. Pre-processing the input text was identified to be the most process intensive component of any text based processing framework. Hence this module of the proposed root-cause analysis framework is implemented and is used for analysis. **Findings:** Performance is measured in terms of scalability, processing time, applicability, usability considering the streaming nature of data. Pre-processing module of the proposed framework is implemented in all of the considered processing architectures. Throttle points for each of the techniques is documented. It was identified that the scalability levels provided by sequential systems were not sufficient to handle the voluminous data. Considering the parallel approaches namely, CPU parallel, Hadoop MapReduce and Spark, it was identified that the CPU parallel approach exhibits effective performance until a certain level, after which the architecture fails. Hadoop and Spark based techniques exhibits high scalability levels, due to the underlying HDFS structure. However, their pros and cons in terms of other metrics indicate that the in-memory technique used by Sparkworks best both in terms of scalability and time complexity levels. Due to the dynamic nature of data under consideration, Spark architecture was identified to be the best for a root-cause analysis architecture. **Novelty/ Improvement:** A novel root-cause analysis framework incorporating pre-processing modules, aspect extraction and fuzzy based sentiment identification of aspects, rather than the conventional polarity analysis is proposed.

Keywords: Aspect Extraction, In-Memory Processing, Parallelization, Root Cause Analysis, Sentiment Analysis

1. Introduction

Increase in automation and reduction in communication costs has led to a huge increase in the data being generated. The advantage of such data is that it is information rich and hence can be mined to obtain very valuable knowledge. Problem arises during the process of mining such data. Increase in the amount of data generated was not proportional to the technological development. Data generation has shown huge growth, while technological enhancement has shown its growth with respect to Moore's Law.[1] This acts as a huge downside when considering the

*Author for correspondence

process of analyzing data. This gap in the technological development has been compensated for by introducing the concept of parallelization. The introduction of parallelism was also due to the inability to incorporate units of higher processing capabilities into a single chip due to power constraints.[2,3] Hence multiple processing units were integrated into a single chip and they were made to operate in parallel by sharing a single workload. Though this process has considerably reduced the response time, speed of the memory devices has become the next throttle point. Expected speedups are achieved if the data involved fits into the main memory. When the data size exceeds this limit, thrashing is performed by switching data from hard disk to main memory and vice versa. Since the access speed of secondary storage units are very low compared to main memory, the processing times are automatically increased.

This paper presents an in-memory solution for processing huge streaming data such that it exhibits low data access times hence exhibiting better speedup.

The architecture proposed in this paper comprises of several independent components. These components and their corresponding contributions available in literature are discussed in this section.

Aspect/ Opinion mining, being a mature domain, contains several contributions to its credit. A review on mining components from unstructured reviews is presented in the article.[4] This review analyzes computational techniques, models and algorithms available for identifying aspects from unstructured reviews. A method to enrich the knowledge bases using context based approaches is presented.[5] This method focuses mainly on large semantic knowledge bases. This method identifies ambiguous semantic terms and enriches them by incorporating them with domain specific information. Since several information retrieval techniques rely on knowledge from real time customer generated data, enriching the knowledge bases is of huge priority. Surveys on the impact of social media data on the sentiment analysis schemes is proposed.[6,7] The importance of sentiment contextualization in the area of opinion mining is emphasized.[8] Existing context-aware approaches uses language models [9], vector space modeling, linguistic patterns, rule based approaches or apply sentence- and discourse-based context shifters.[10-13] POS based contextual polarity analysis is presented in the article.[14] This method uses Principle Component Analysis (PCA) to select components from the source domain. Context knowledge based semantic lexicon enrichment is another huge area of research.[15] A subtopic based mining using aspects obtained from query is presented in the presented article.[16] This method mines aspects directly or indirectly from the query and hence obtains subtopics related to the current query. A sampling based sentiment mining approach for e-commerce applications was presented.[17] This technique relies upon vector models to create and effectively process Twitter streams. Emoticons are also considered as intrinsic parts of sentiment analysis due to the emergence of social media data. This as a major part of the analytics framework is discussed.[18] Another technique incorporating microblogs for sentiment analysis was proposed[19] Several applications of aspect mining include customer preference about tourism products, customer opinion mining on tourism, analyzing movie reviews, customer relationship management, analyzing online reviews etc.[20-24]

2. Proposed Architecture

Root Cause Analysis (RCA) from text is chosen as the field of operation, as it requires faster processing of huge amount of data. Root Cause Analysis is the process of identifying the root cause of the problem, whose removal would prevent an undesirable event from occurring. RCA from text specific to a domain can reveal several aspects of the domain and their polarity as perceived by the users/ customers. Sentiment mining plays a vital role in this process in segregating the text based on their polarity.

A fast and effective RCA technique that can be used on huge streaming data is presented in Figure 1. The initial data preprocessing includes tokenization, stemming and normalization. This is followed by aspect extraction, which extracts single aspects, followed by multiple aspects. Aspect sentiments are identified and polarity based segregation divides the data into positive and negative classes. Root causes are then identified using Significant Term Aggregation (STA). The root causes identified are then ranked and displayed to the user based on the threshold.

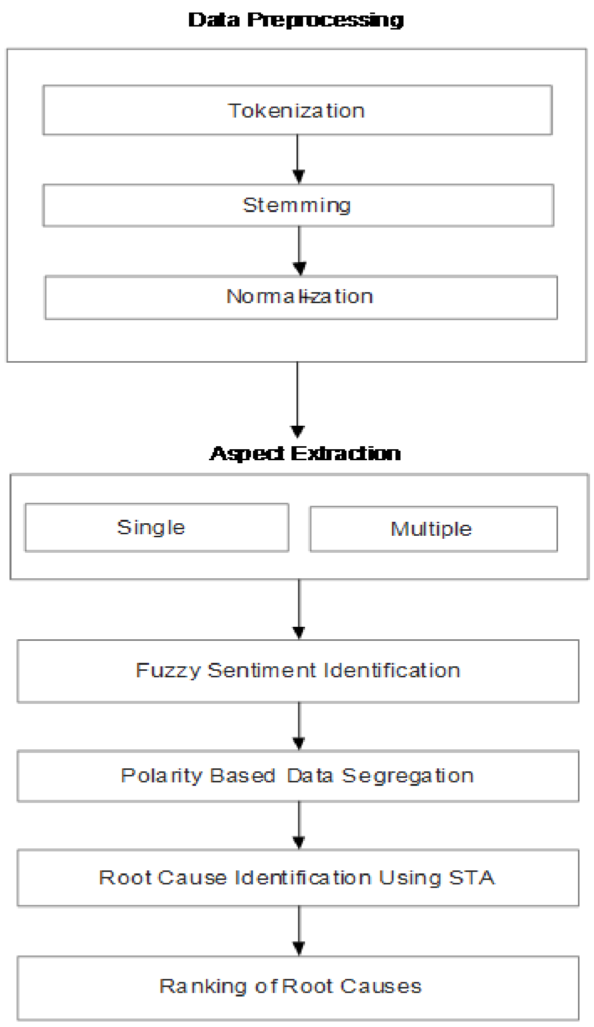


Figure 1. Fast and Effective Root Cause Analysis of Huge Streaming Data using In-Memory Processing Techniques.

2.1 Data Preprocessing

Data Preprocessing involves using several techniques to identify the major components of the text.[25] The data preprocessing framework in our architecture involves three major components; tokenization, stemming and normalization. Tokenization deals with identifying the independent components of the text. Stemming eliminates affixes (prefix and suffix) from a word to obtain the seed word. Current approach utilizes the Porter Stemmer 2, a popular version of the algorithm proposed.[26] Normalization converts the text to canonical format so that uniformity can be achieved in the corpus. The process involves a large data corpus, as tokenizer is directly applied on the raw data. Hence a huge amount of processing is required.

2.2 Aspect Extraction

Aspect extraction is the process of identifying objects of importance in a text. A text may contain one or several aspects of importance. These aspects determine the polarity and the context of the text. Aspect extraction is carried out by initially identifying the entities contained in the text. An entity is a noun around which the text is built upon. Aspects are sub-categories of entities. Domain ontology is one of the major requirements for aspect extraction. The ontology serves as the rule base in identifying the aspects. Usually aspects are identified in the form of 1-gram words. Since the aspects combined with several other phrases has the probability of inversing the polarity of the text, this approach uses both 1-gram and n-gram extraction of aspects. The n-gram retrieval is not constant. Instead, the neighbors are analyzed and are selected heuristically based on several criterion. An n-gram aspect is always preferred over a (n-1)-gram aspect. An added advantage is that n-gram based aspect extraction automatically eliminates several commonly occurring structures in the text automatically.

2.3 Fuzzy Sentiment Identification and Polarity based Segregation

The extracted aspects are passed to the sentiment identification phase. Sentiment identification is the process of identifying the polarity of the aspect under analysis. Since the current approach extracts several aspects from the text, every aspect is passed through this phase to obtain its corresponding sentiment. An aspect may not necessarily be positive or negative. A single aspect may contain a level of both positive and negative polarities associated with it. Hence the proposed approach operates on fuzzy semantic levels rather than defined polarities. In case of a 1-gram aspect, both the positive and the negative magnitudes of the aspect are considered, while in case of n-gram aspects, a single polarity is chosen based on the surrounding text and its magnitude corresponding to the given aspect is chosen. This phase is followed by the segregation of aspects based on their polarity. In case of aspects with both polarity levels, the aspect is duplicated and is copied to both the sections. The end of this phase builds a polarity based repository containing list of aspects exhibiting features corresponding to that category. The unstructured data analyzed until this phase is then converted to structured data of the form

$$\{\text{entity}, \text{aspect}, \text{context}, \text{sentiment}, \text{opinion holder}, \text{time}\} \quad (1)$$

Where *entity* is the object under consideration, *context* is used to identify the context in which the review is given (weightage could be provided based on context*), sentiment* can be Positive, Negative or Neutral, *opinion holder* is the one who expressed the opinion (could be a combination of fields), better if the gender, location, age, occupation etc. could be identified and added to this component and *time* represents the timestamp when the opinion is expressed. This structured data maps the customer with the aspects and their corresponding sentiments and is called the customer aspect mapping table.

2.4 Root Cause Identification using STA

Identification of root causes is performed by identifying aspects with high impacts. This is performed by performing Significant Term Aggregation (STA). Identifying the significant terms requires identification of their Term Frequency (TF) and Inverted Document Frequency (IDF). Term Frequency/ Inverted Document Frequency (TF-IDF) is a statistic that determines the importance of a word in a huge collection of documents. This is a comparative system, which identifies the importance of a word in the domain based repository and the importance of the word in the global repository. This helps identify the significance of the word with respect to the current operating domain.

$$tfidf(t,d,D) = tf(t,d) \times idf(t,D) \quad (2)$$

Term Frequency (TF) and the Inverted Document Frequency (IDF) are calculated using the below formulae.[27, 28]

$$tf(t,d) = \frac{f(t,d)}{count(w,d)} \quad (3)$$

where *f(t,d)* refers to the number of times the word *t* is present in the document *d* and *count(w,d)* refers to the number of words in the document *d.*

$$idf(t,D) = \log\log\frac{N}{|\{d \in D : t \in d\}|} \quad (4)$$

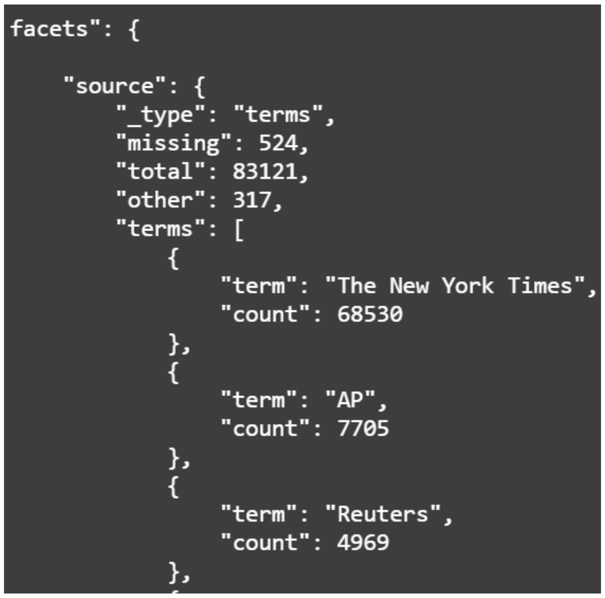
where, N is the total number of documents in the corpus, and is the number of documents that contains word *t*. If the term is not in the corpus, then it will lead to a divide-by-zero error, hence it is also common to adjust the denominator toTerms with high significance corresponds to terms with high TF-IDF values. This process is carried out individually on repositories containing positive and negative aspects. The results are ranked in decreasing order according to the TF-IDF scores. A threshold is maintained and values falling below the threshold are pruned to obtain the final set of root causes independently in each polarity domain.

The identified root causes are mapped with the customer aspect mapping table to obtain the customer/ customers concerned about the particular aspect and the polarity of their review. The rank of concerned customer/ customers is analyzed and the final product based recommendations are obtained.

2.5 Dataset Analysis

The datasets that are considered for the current study includes Twitter API [29], Google API and the New York Times API.[29-31] Major properties of such API is that they do not provide the datasets as a whole. Instead, the user is required to query the database to obtain a set of results according to the query. The number of results returned via the APIs are constrained (even for registered users). Hence the users are either required to wait for the required amount of data and then begin the processing, or the users need to perform their processing on-the-go with the data block available in hand. One major drawback is that this data cannot be used as-such. It is noisy and contains several components other than the actual data.

A snapshot of the data returned from New York Times API is presented in Figure 2. The data is in JSON format representing several components including the actual data required by the application. Elimination of undesired components become mandatory. This process is then followed by in-memory processing of data to obtain the root causes.



```
facets": {

    "source": {
        "_type": "terms",
        "missing": 524,
        "total": 83121,
        "other": 317,
        "terms": [
            {
                "term": "The New York Times",
                "count": 68530
            },
            {
                "term": "AP",
                "count": 7705
            },
            {
                "term": "Reuters",
                "count": 4969
            },
```

Figure 2. A Sample Snapshot of API Data.

3. Solutions under Consideration

The following solutions were considered for performing root cause analysis on huge streaming data.

3.1 Map Reduce Paradigm (Sequential) Processing Techniques

Map Reduce is a parallel problem solving method that divides the process into two basic phases; Mapper phase and the Reducer phase. The mapper phase usually performs the processing and the reducer phase aggregates the data presented by the mapper phase to provide the final results. Though this is a parallel problem solving architecture, it can operate effectively even in a sequential working scenario. Scalability of this system is constrained by the main memory, hence the scaling level of this architecture is the level of the main memory involved in the processing system. Though the property of utilizing the main memory storage is a constraint in terms of scaling, it provides an advantage to this approach by speeding up the process to provide faster results. Hence the applicability of this approach is constrained in the areas involving less data and more processing. Our approach implements the Map Reduce paradigm implemented in sequential context in Python and efficiency obtained in this approach is discussed in section 4.

3.2 Parallel Processing Techniques

Parallel processing is one of the areas on the raise due to the increase in the parallel processing architectures and improved parallel processing capabilities. This section describes two of the most basic parallel processing architectures available and their advantages over other architectures.

3.2.1 CPU based Parallelization

CPUs form the basic component of any computing system. Due to the increase in the processing capabilities of individual CPUs and the introduction of multi-core CPUs has paved way for effective parallelization using them. Major advantage of this approach is that it is closely coupled with the operating system, hence provides high efficiencies. The major downside of this approach is its inability to process huge amounts of data. This also constraints the scalability of the architecture to a large extent.

3.2.2 Hadoop

In order to effectively utilize huge data and bring about effective parallel processing, HDFS was introduced. Hadoop architecture works on this file system, thereby utilizing a large memory and performing parallel processing in it. Data stored in HDFS is treated as a group of bytes, rather than its structured form. This makes HDFS suitable for Big Data, where Volume, Velocity and Variety of data play a major role. Hence this architecture is highly scalable and can accommodate data of any size. The tradeoff comes from the fact that the accommodation of large data requires appropriate indexing techniques, so any processing requires a constant data retrieval time. Hence this makes the approach suitable only for very large amounts of data, while operations on small data indicates unnecessary data retrieval latencies.

3.2.3 Spark

Efficiency provided by Hadoop architectures are not sufficient in recent working environments due to the huge time requirements. HDFS operates on the basis of secondary memory. Even though operations are performed in parallel, they are not suitable for real time faster processing environments. Spark utilizes in-memory based operation, thereby utilizing the main memory of the systems to the maximum extent in-order to reduce the processing time.

4. Results and Discussion

The preprocessing phase was implemented and experiments were conducted to observe the efficiencies exhibited by the techniques discussed in Section 4. Implementations were carried using C#.NET for analyzing CPU based parallelization, Python based MapReduce implementation for analyzing the Map Reduce Paradigm implemented in sequential context using Python, Python based Hadoop implementation for Hadoop and PySpark for analyzing its efficiency on in-memory processing. Experiments were conducted on the book dataset obtained from the presented article.[32] Figures 3-9 are constructed by applying a single dataset on all the approaches and identifying their processing capabilities.

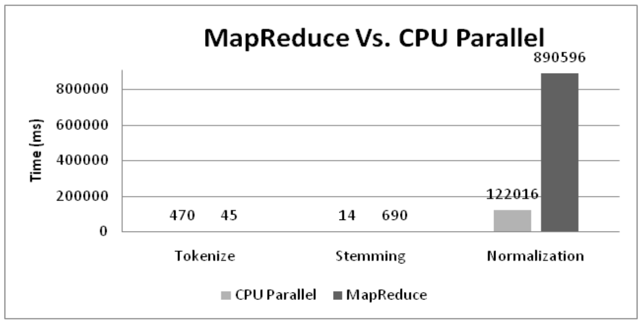


Figure 3. MapReduce Vs. CPU Parallel.

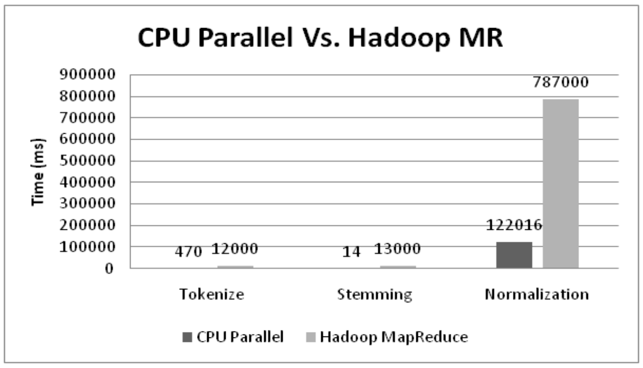


Figure 4. CPU Parallel Vs. Hadoop MR.

A comparison between C# Parallel implementation and all the other methods under discussion are presented in Figure 3-5. It could be observed that irrespective of the technique, CPU parallel implementations exhibit faster processing rates. This occurs due to the flexibility available for the CPU parallel implementations to directly utilize the main memory.

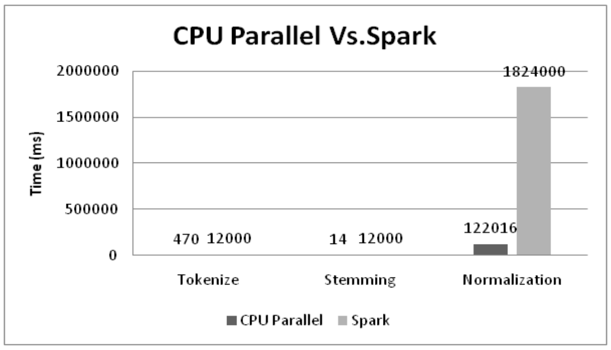


Figure 5. CPU Parallel Vs. Spark.

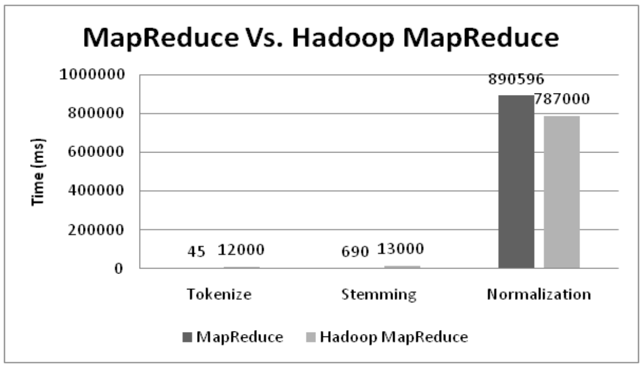


Figure 6. MapReduce Vs. Hadoop MapReduce.

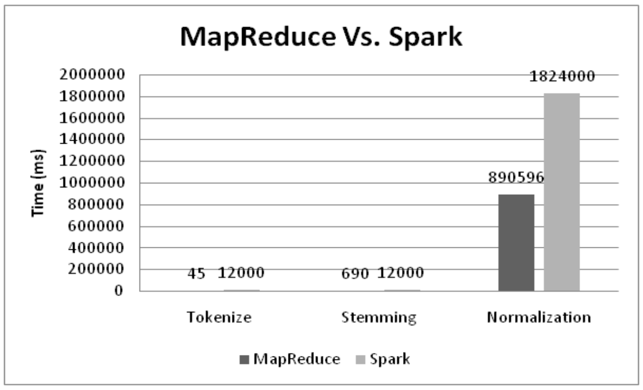


Figure 7. MapReduce Vs. Spark.

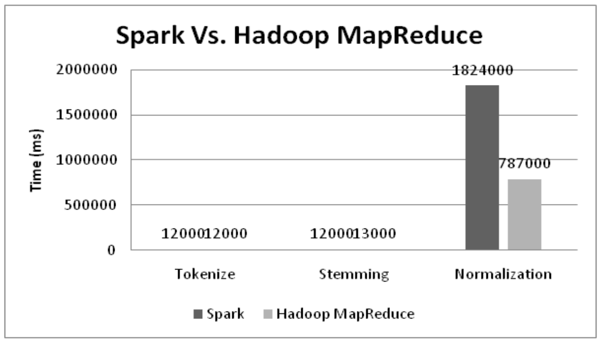


Figure 8. Spark Vs. Hadoop MapReduce.

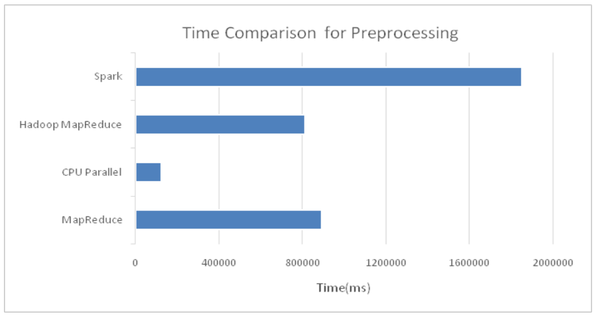


Figure 9. Time Comparison for Preprocessing.

A comparison between the Map Reduce paradigm in sequential context and Spark based implementation is presented in Figure 7. It can be observed that the Spark based implementation exhibits higher time requirements when compared with the implementation of Map Reduce paradigm in sequential context.

A comparison between Hadoop Map Reduce and Spark based implementation of the pre-processing phase is shown in Figure 8. It could be observed that the tokenization and the stemming phases show similar processing times, while the normalization phase shows that Hadoop based implementations exhibit faster processing when compared to Spark based implementations. The quantified results for time comparison is presented in Table 1.

Table 1. Time Comparison on Various Platforms

	Tokenization	Stemming	Normalization
MapReduce	45	690	890596
CPU Parallel	470	14	122016
Hadoop MapReduce	12000	13000	787000
Spark	12000	12000	1824000

The aggregated time taken for the preprocessing is shown in Figure 9. It could be observed that the time taken for the entire preprocessing phase is the least in CPU based parallelization techniques and increases in the order of Hadoop based Map-Reduce, Map Reduce paradigm in sequential context and finally Spark.

In order to identify the threshold points to identify the performance increase in in-memory techniques, the data set size was varied gradually until the throttle point of 400 books, where the CPU based implementations exhibited very huge time requirements. This observation correlated with the fact that CPU based parallel implementation depend largely on the main memory. On exceeding this level, CPU based implementations fail to operate.

Time comparison between parallel variants of the preprocessing algorithm is presented in Figure 10. It could be observed that CPU based parallel implementations exhibited linear increase with respect to the data size varying from 1 to 100. At the throttle point of 400, the algorithm achieved its maximum limit and hence exhibited and exponential time increase. The Hadoop Map Reduce algorithm also showed similar increase, but it remained scalable due to the huge memory support provided by HDFS. The Spark based algorithm, though exhibited high time requirements, and was more robust to increase in data, in terms of processing time. Even with huge increase in the size of data, the time requirements did not raise much, hence providing a stable platform for processing. The aggregated time for preprocessing is shown in Table 2.

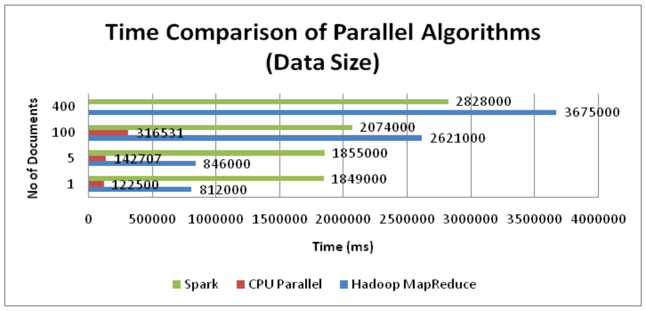


Figure 10. Time Comparison of Parallel Algorithms (Data Size).

Table 2. Aggregated Time Taken for Pre-Processing

Technique	Time (ms)
Python	1057975
MapReduce	891331
CPU Parallel	122500
Hadoop MapReduce	812000
Spark	1848000

Time variance exhibited by the parallel variants of the pre-processing algorithm is presented in Figure 11. It could be observed that Hadoop based implementation exhibits very high variance, followed by CPU based parallel implementation. Though it has moderate variance, the throttle point for the CPU based implementation remains low due to memory constraints. The Spark based implementation exhibits very low variance. Even this method exhibited high processing times with smaller data, it was observed that the increase in time is very low with respect to the increase in the data size. Since the data for Spark can be obtained from HDFS, it is also more robust to increase in data. This scalability factor makes in-memory based processing techniques best candidates for processing huge streaming data.

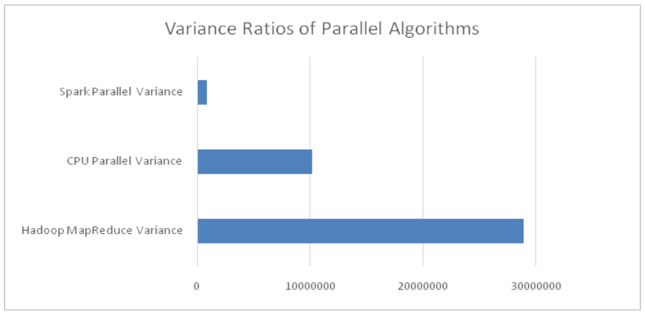


Figure 11. Variance Ratios of Parallel Algorithms.

5. Conclusion

This paper presents analysis of techniques that can be used on huge streaming data to obtain a fast and scalable system. Root cause analysis using aspect mining techniques on huge data is considered as the domain of operation. CPU based parallelization, Map Reduce Paradigm in sequential context, Hadoop and in-memory processing using Spark were used for analysis. It was observed from the results that Map Reduce paradigm in sequential context and CPU based techniques are constrained by data size, while Hadoop based techniques exhibit linear increase in processing times. Spark based techniques exhibiter higher processing times when compared with other techniques in the initial stages. As the data size is increased, it was observed that the variance exhibited by in-memory based Spark is much lower when compared to the other techniques and after a throttle point, it was observed that Spark based technique exhibited more efficient processing speeds when compared to other techniques.

6. References

1. Moore GE. Cramming more components onto integrated circuits. IEEE solid-state circuits society newsletter. 2006; 20(3):33–5.
2. McMenamin A. The end of Dennard scaling; 2013 April 15.
3. Bohr M. A 30 year retrospective on Dennard's MOSFET scaling paper. IEEE Solid-State Circuits Society Newsletter. 2007; 12(1):11-3. Crossref
4. Khan K, Baharudin B, Khan A, Ullah A. Mining opinion components from unstructured reviews: A review. Journal of King Saud University-Computer and Information Sciences. 2013; 26(3):258–75. Crossref
5. Weichselbraun A, Gindl S, Scharl A. Enriching semantic knowledge bases for opinion mining in big data applications.Knowledge-based systems. 2014; 69:78–85. Crossref PMid:25431524 PMCid:PMC4235782
6. Ghaleb OAM, Vijendran AS. Survey and Analysis of Recent Sentiment Analysis Schemes Relating to Social Media. Indian Journal of Science and Technology. 2016 Nov 10; 9(41):1–16.
7. Ragini JR, Anand PR. Sentiment Analysis: A Comprehensive Overview and the State of Art Research Challenges. Indian Journal of Science and Technology. 2016 Dec 29; 8(1):1–7.
8. Gangemi A, Presutti V, Recupero DR. Frame-based detection of opinion holders and topics: a model and a tool. Computational Intelligence Magazine. IEEE. 2014; 9(1):20–30. Crossref
9. Lau RYK, Lai CL, Li Y. Leveraging the web context for context-sensitive opinion mining. In Computer Science and Information Technology. 2nd IEEE International Conference; 2009. p. 467–71. Crossref
10. Das A, Gambäck B. Sentimantics: conceptual spaces for lexical sentiment polarity representation with contextuality. In Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis. Association for Computational Linguistics; 2012 July. p. 38–46. PMid:22690049 PMCid:PMC3361776
11. Wu Y, Wen M. Disambiguating dynamic sentiment ambiguous adjectives. In Proceedings of the 23rd International Conference on Computational Linguistics. Association for Computational Linguistics; 2010 Aug. p. 1191–9. PMCid:PMC2872725
12. Ding X, Liu B, Yu PS. A holistic lexicon-based approach to opinion mining. In Proceedings of the International Conference on Web Search and Data Mining. ACM; 2008 Feb. p. 231–40. Crossref
13. Wilson T, Wiebe J, Hoffmann P. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. Computational linguistics. 2009; 35(3):399–433. Crossref
14. Xia R,Zong C, Hu X, Cambria E. Feature ensemble plus sample selection: domain adaptation for sentiment classification. Intelligent Systems. IEEE. 2013; 28(3):10–8. Crossref
15. Lu Y, Castellanos M, Dayal U, Zhai C. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In Proceedings of the 20th international

conference on World wide web. ACM; 2011 Mar. p. 347–56. Crossref

16. Wang CJ, Lin YW, Tsai MF, Chen HH. Mining subtopics from different aspects for diversifying search results. Information retrieval. 2013; 16(4):452-483. Crossref
17. Vinodhini G, Chandrasekaran RM. A sampling based sentiment mining approach for e-commerce applications. Information Processing & Management. 2017; 53(1):223–36. Crossref
18. Nirmal VJ, Amalarethinam DG. Emoticon based Sentiment Analysis using Parallel Analytics on Hadoop. Indian Journal of Science and Technology. 2016 Sep 17; 9(33):1–7. Crossref
19. Wu F, Song Y, Huang Y. Microblog sentiment classification with heterogeneous sentiment knowledge. Information Sciences. 2016 Dec 10; 373:149–64. Crossref
20. Marrese-Taylor E, Velásquez JD, Bravo-Marquez F, Matsuo Y. Identifying customer preferences about tourism products using an aspect-based opinion mining approach. Procedia Computer Science. 2013 Jan 1; 22:182–91. Crossref
21. Bucur C. Using Opinion Mining Techniques in Tourism. Procedia Economics and Finance. 2015; 23:1666–73. Crossref
22. Basari AS, Hussin B, Ananta IG, Zeniarja J. Opinion mining of movie review using hybrid method of support vector machine and particle swarm optimization. Procedia Engineering. 2013 Jan 1; 53:453–62. Crossref
23. TuzhilinA. Customer relationship management and Web mining: the next frontier. Data Mining and Knowledge Discovery. 2012; 24(3):584–612. Crossref
24. Li Y, Qin Z, Xu W, Guo J. A holistic model of mining product aspects and associated sentiments from online reviews. Multimedia Tools and Applications. 2015 Dec 1; 74(23):10177–94. Crossref
25. George A, Nirmal VJ. Parallelization techniques in Pre-Processing Phase for Sentiment Analysis in Big Data: A Comparative Analysis. International Journal of Applied Engineering Research. 2015; 10(16):37270–7.
26. Porter MF. An algorithm for suffix stripping. Program.1980; 14(3):130–7. Crossref
27. Baeza-Yates R, Ribeiro-Neto B. Modern information retrieval. New York: ACM press; 1999 May 15.
28. Salton G. Automatic text processing: The transformation analysis and retrieval of Reading: Addison-Wesley; 1989.
29. Twitter developer documentation. Available from: Crossref
30. Google APIs Explorer. Available from: Crossref Date Accessed: 5/1/2017
31. The NewYark times developer network. Available from: Crossref Date Accessed: 5/1/2017
32. Free ebooks-Project Gutenberg. Available from: Crossref Date Accessed: 5/1/2017