RESEARCHERID
THOMSON REUTERS

[Tiwari* *et al.*, 6(9): September, 2017]
IC™ Value: 3.00

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A HADOOP BASED COLLABORATIVE FILTERING RECOMMENDER SYSTEM ACCELERATED ON GPU USING OPENCL

Jyotindra Tiwari [*1], Dr. Mahesh Pawar [*2], Dr. Anjana Pandey

[*1] School of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, India
[*2] Department of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, India
[*3] Department of Information Technology, Rajiv Gandhi Proudyogiki Vishwavidyalaya, India

DOI: 10.5281/zenodo.886896

ABSTRACT

Recommender systems are valuable tools to provide service recommendations to the users. The data available online is growing rapidly because online activity of customers has grown rapidly. This has raised big data analysis problem for recommender systems as consumers of service demands better recommendations from the service providers. To process and analyze this large scale data the traditional service recommenders systems suffer the problem of scalability and inefficiency. Most service recommender systems lack a level of personalization which means the recommendations generated are not personalized for users. So, a user may not get the recommendations which he likes. In this paper we present a faster and better collaborative filtering recommender system technique then pre existing techniques. The problem of scalability is addressed by using Hadoop framework with OpenCL.

KEYWORDS: Hadoop, Recommender System, Opencl

I. INTRODUCTION

In last decade a dramatic increase in variety in services provided by companies has been seen. Now a days companies provide vast number of services and products to meet the demands of a customer. In this way customers get more options to choose from but due to huge amount of data it makes harder for service provider companies to process such a huge amount of data. Recommender systems are used by companies to recommend users the product and services based upon the interest of users. In order to generate recommendations the likes, preferences and usage history of a user is considered. Practical implementations of Recommender System algorithms are done by various multi-national companies and tech giants these days and their domain ranges from hotels, online shopping, social networks, movies, news etc. A recommender system is used to recommend items to the users based upon their previous preferences of items.

GroupLens introduced the first known implementing project in the recommender system area. In the early days the main aim of was to apply and explore the automated collaborative filtering. The collaboriative filtering was first applied elsewhere in filtering the information in Usenet news[1]. Music recommender systems provide personalized music recommendations and Ringo agent was one of the first applications [2]. Another system which creates recommendation by using social filtering was proposed which is the automation of the word-of-mouth recommeder sytems. Amazon created BookMatcher system for book recommendations. Recommender systems face a huge amount of problems and certain areas of improvements. Recommendation systems are not personalized which means they do are not user specific. In our work we have proposed that a user preferences should be taken into account to generate recommendations.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

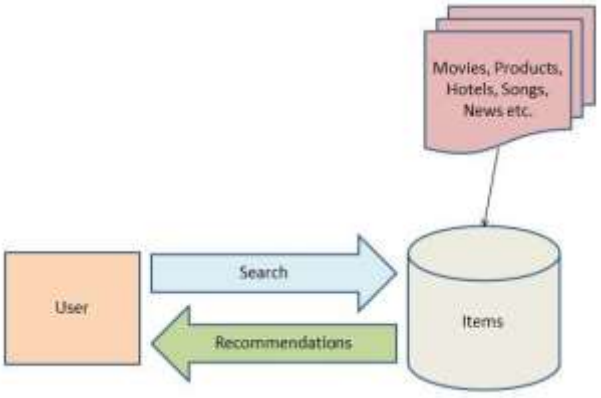

Figure 1: Basic Recommender System

This approach makes the recommender system more user specific and the recommendations thus generated match user interests. Another major challenge face by recommender system is the issue of scalability. As the recommender systems require processing of huge amount of data called Big Data. A huge amount of computation resources are required. The traditional approach has been to process the data on a cluster using a distributed processing framework such as Hadoop. In this paper we propose a fine grained parallel processing of Big Data. In this way the data processing is done in quick time and the results show a clear speedup when compared with pre existing techniques. Another issue is to enhance the performance of Recommender System i.e. to generate better recommendations.. A keyword set used is huge and which gives users a vast options to choose from. A recommender system may output a single item of a list of top ranked items for each user. There are several techniques to generate recommendations but two kinds of techniques are most used and these techniques are : Collaborative Filtering and Content-based recommendations[3].

1. Content Based Recommender Systems:

Content-based filtering recommends items based on a comparison between user profile data and content of items. Content-based filtering is also called Cognitive-filtering. The content of each items is the description of that item which are the terms occur in a document to describe that item. The same terms are used to represent user profile and built up by analyzing the content of items which have been seen by the user. The Recommendations are generated by matching users and items content.

2. Collaborative Filtering Based Recommender Systems:

Collaborative filtering generates recommendations by using the previous recommendations of other users. Collaborative filtering is also called as social filtering. The basic idea behind Collaborative filtering is that if a user likes a certain type of items in past then it may like similar items in future. A simple example of Collaborative filtering is that if users want to watch a movie then they will ask their friends who have similar taste. Because it is most likely that the users will like the movie of which matches the interest of a person who have same interest as the user. This information is used in the decision on which movie to see.

2.1. User–User Collaborative Filtering:

User-user Collaborative filtering is straightforward algorithmic implementation of collaborative filtering. In User-user CF the users whose past preferences behavior matches with current users are determined and using the rating of those users ratings of other items are predicted to determine the recommendations. Suppose for a user A if the recommendation have to be determined then we have to predict the ratings which user A would give to those items which have not been rated by user A. Collaborative Filtering will determine those users which have high similar interest with user A or in other words have a high agreement with A based upon the previous items which both have rated. These users are then used to calculate a weighted mean rating of the

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

unrated items depending upon the weights or the level of similarity between the other user and user A. This weighted mean approach gives priority to those users who have a greater match with the user A. User-user collaborative filtering is an effective technique but the drawbacks are that it suffers from scalability issues when the number of users grows.

2.2. Item–Item Collaborative Filtering :

Item-item collaborative filtering uses the set of items which the target user has rated and and determines similarity between the target item and set of items. Based on the similarity selects the top similar items. Prediction is computed by taking a weighted average on the target user's ratings on the most similar items. First the users which have rated the items are isolated then the similarity is calculated by applying similarity computation technique. Item-item collaborative filtering is one of the most widely deployed technique today. Rather than using similarities between user's rating behavior to predict preferences, item-item CF uses similarities between the rating patterns of items.

2.3. Hybrid Recommender Systems:

Both content-based filtering and collaborative filtering have there strengths and weaknesses. Three specific problems can be distinguished for content-based filtering: content description, over-specialization and subjective domain specialization. Content description addresses that generating useful description of the content can be very difficult. A content-based filtering system will not select items if the previous user behavior does not provide evidence for this. Additional techniques have to be added to give the system the capability to make suggestion outside the scope of what the user has already shown interest in. Content-based filtering techniques have difficulty in distinguishing between subjective information such as points of views and humor.

A collaborative filtering system doesn't have these shortcomings. Because there is no need for a description of the items being recommended, the system can deal with any kind of information. Furthermore, the system is able to recommend items to the user which may have a very different content from what the user has indicated .

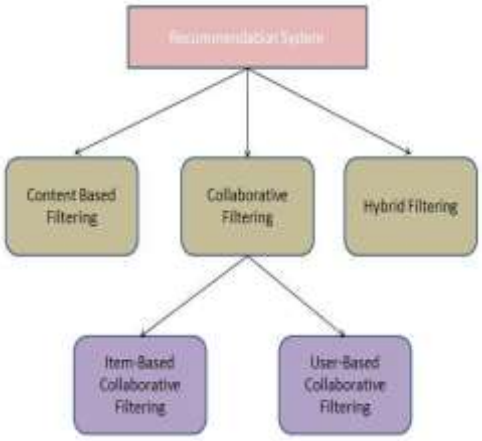

Figure 2: Categories of Recommendation Methods

Most of the recommender systems present recommendation list which is similar to the ratings of the services. They have not considered users different preferences, without meeting users' personalized requirements. [4] S Meng. At al. have addressed this problem and proposed a new recommender system methodology. They have used a novel approach for collaborative filtering. The preferences of a user are taken in the form of keywords. The implementation of the approach is done on Hadoop Mapreduce parallel computing framework.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

KASR METHODOLOGY

The keyword aware approach for recommender systems takes user preferences as keywords and generates recommendations according to that. There are two types of users : active user and previous user. A keyword candidate list is contains the list of all keywords the user determines the keywords from this list and user determines preferences by using the keywords from this list. An active user is a user who gives preferences as keywords or simply the user whose recommendations are determined. An active user can select keywords from keyword-candidate list which are preferences of that user and it represents the quality criteria which he/she is most concerned. The previous users are all other users whose keywords are determined by the reviews of the service given by each user. The preferences of a previous user are extracted from the reviews given by the user for the service.

Preprocess: This is the first step and in this step HTML tags and stop words are removed from the review set to keep only the relevant words to extract keywords in the next stage. The Porter Stemmer algorithm (keyword stripping) is used to remove the commoner morphological and inflexional endings from words in English[5]. For example if there are several words having same root word such as run, runner and running will be stemmed to same word run. The idea behind is that all the three words if comes in a review will be used in the same context which is run. Porter Stemmer comes from Information Retrieval Systems and it is mainly used for normalization of terms[5].

Keyword extraction: Each review is transformed into a keyword set which is a set of corresponding keywords in a review and the keywords are taken from the keyword-candidate list and domain thesaurus. If the review has a word in the from the domain thesaurus, then the corresponding keyword should be extracted into the preference keyword set of the user (APK). The preferences of a previous user (PPK) for a candidate service are extracted from his/her reviews for the service according to the keyword-candidate list and domain thesaurus. A review of the previous user will be formalized into the preference keyword set of him/her.

Measurement	General	Average	Important	very Important	Most Significant
Importance Degree	1	2	3	4	5

Table 1: Importance Degree of Keywords

Similarity computation: When the keywords are extracted from the reviews for previous users then the next step is to identify the reviews of previous users who have similar tastes to an active user. The neighborhood of active user is determined based on the similarity of their preferences with previous user. The reviews which are not related to active user will be filtered out by determining the intersection between active user keyword preference and previous user keyword preference set. If the intersection of active user keyword preference set and previous user keyword preference set is a null set or a void set then that particular previous user will be left out. Two similarity computation methods are used in keyword aware recommendation an approximate similarity computation method and an exact similarity computation method. When the weight of the keywords in the preference keyword set are not available approximate similarity method is used and when the weight is given exact similarity computation method is used. The preference weight vector is determined for Exact similarity computation by using TFIDF vectorization approach[6].

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Approximate similarity computation— It is a frequently used method for computing approximate similarity. In approximate similarity computation Jaccard coefficient is applied in the approximate similarity computation to calculate the diversity and similarity of sample sets.

$$sim_{ASC}(APK, PPK_j) = \frac{|APK \cap PPK_j|}{|APK \cup PPK_j|} \quad (1)$$

Exact similarity computation— It is a cosine based approach applied in exact similarity computation which uses cosine similarity formula to calculate similarity. It is very similar to the Vector Space Model in information retrieval[7][8].

$$sim_{ESC}(APK, PPK) = \cos(W_{AP}, W_{PP}) = \frac{W_{AP} \cdot W_{PP}}{\|W_{AP}\| \times \|W_{PP}\|} \quad (2)$$

Calculate personalized ratings and generate recommendations: When the similarity between active user and previous user is calculated the further filtering will be conducted which will filter out those previous users which are very different from active user. For a give threshold, if similarity between active user and previous user is less than that threshold then previous user keyword set will be filtered out. So for a given theta,

If, sim(APK,PPK) < θ (3)

PPK is retained otherwise PPK is filtered out. The thresholds given in two similarity computation methods are different, which are both empirical values.

Once the set of most similar users are found, the personalized ratings of each candidate service for the active user can be calculated. After calculating the personalized ratings for all the services for the active user, sorting of ratings is done. By considering some top highest ratings, the personalized recommendation list is provided to the user. Here, we use a weighted average approach to calculate the personalized rating pr of a service for the active user. The formula to calculate personalized rating by Weighted Average Approach is

$$pr = \bar{r} + k \sum_{PPK_j \in \hat{R}} sim(APK, PPK_j) \times (r_j - \bar{r}) \quad (4)$$

$$k = 1 / \sum_{PPK_j \in \hat{R}} sim(APK, PPK_j) \quad (5)$$

KASR ON HADOOP

KASR is implemented on Hadoop environment to handle big data generated by recommendation systems in order to improve the scalability and efficiency. Figure 3 shows the flow chart of computation of KASR on Hadoop environment. The review file is first split into blocks. These blocks are placed on Mapper in the form of Key/Value pairs. In each Mapper, there are various algorithms running after removal of stop words, HTML tags, etc. from the review file, porterStemmer(), keywordExtractor(), similarityComputation(), personalizedRating().

porterStemmer(): This algorithm reduces the number of unique terms from the input dataset.

keywordExtractor() : This algorithm is used to extract the keywords from the active user and reviews given by previous users from the review file.

similarityComputation() : This algorithm implements the similarity computation between active user's and previous user's preferences.

personalizedRating() : This algorithm is used to calculate the personalized rating of for each services based on the similarity of active user's and previous user's preferences.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

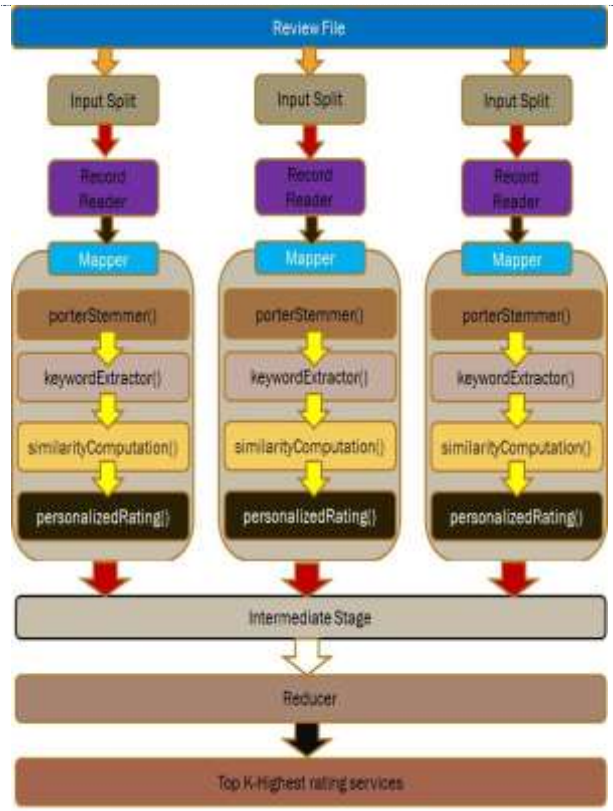

Figure 3: Flow chart of computation of KASR on Hadoop

Mapper produces output by using these above algorithms in the form of Key/Value pairs. These Key/Value pairs are given to Reducer after shuffling and sorting. Output of the Reducer is sorted and based on top K highest ratings, the recommendation list is generated.
All content should be written in English and should be in 1 column.

- Page type will be A4 with normal margin, word spacing should be 1.
- No space will be added before or after paragraph.
- This section should be typed in character size 10pt Times New Roman, Justified.

II. PROPOSED METHODOLOGY

The analysis of big data for the recommendation system is very useful. It gives better understanding to the system by analyzing the big data from its statistics and getting the huge amount of information about the customers, reviewers and services from these data. So, this motivates the recommendation systems to analyze the big data generated by these systems on a very large scale to get the valuable information which can help in understanding them about their customers, reviewers and services. It helps in making a better and more accurate system from the information it gets from the analysis of this data.

Moreover, in most of the recommendation systems, the ratings of services and the recommendation list provided to the users are same. They do not consider users different preferences and hence provides the same recommendations to the different users with their different choices and preferences. Different users may have different preferences of services. Irrespective of their preferences, most of the recommendation systems provide the same recommendation list to the users.

These drawbacks are removed by using an approach Keyword-Aware Service Recommendation System (KASR), which provides the recommendation list to the users according to their own preferences. It calculates the personalized ratings and produces the recommendation list to the users by extracting the keywords

from the reviews given by previous users. This KASR approach is implemented on Hadoop framework which gives the system a good technique to analyze the big data generated by these recommendation systems.

The implementation of KASR on Hadoop environment has some drawbacks. First problem is, it doesn't utilize the resources properly. Now days, almost every computer system comes with GPUs and some cores of CPU. Hadoop framework doesn't utilize the GPUs and every cores of CPU which may lead to its poor computation performance and the efficiency may not be up to the mark. Another problem with KASR implementation on Hadoop framework is its energy consumption. It consumes high energy. Large size of cluster consumes high energy in terms of cost of cooling and powering of the hardware together. Several Works have been done to accelerate Mapreduce parallel programs using heterogeneous platforms[9][10][11].

Motivated by these observations, in this paper, the drawbacks of existing KASR system implemented on Hadoop is removed by implementing KASR on HADOOPCL. HADOOPCL[12] provides high level distributed and heterogeneous programming models which provide the system two level of parallelism: Inter-node parallelism and Intra-node parallelism. Because of its heterogeneous property, it utilizes the resources by utilizing the GPUs and cores of CPU. In heterogeneous model, power-efficient processors are used which reduces the consumption of energy.

KASR ON HADOOPCL

Implementation of KASR on Hadoop improves scalability and efficiency. However, it doesn't utilize resources of the system. Now days, every computer system comes with multiple cores of CPU, GPUs, APUs, FPGAs etc. The efficiency of the system improves if we utilize these resources properly. Hadoop doesn't utilize these resources which may lead to poor computational performance. Another problem with Hadoop is it consumes high energy.

Considering these problems in mind, this KASR method is implemented on HadoopCL as it was proposed in Tiwari J et al. [13]. HadoopCL uses OpenCL to utilize the resources like cores of CPUs, GPUs, APUs, FPGAs, etc. Moreover, a significant amount of energy is used just to keep a system alive. In idle state a system consumes 70% of the energy . This significant amount of energy is wasted by the system if they are in idle state. Most of the energy is used inefficiently because improper utilization of resources such as CPU, Memory, I/O etc. To reduce the consumption of energy because of idle state of systems and poor resource utilization, the KASR method is implemented on heterogeneous environment HadoopCL which provides proper utilization of resources. Researchers of Peng et al. (2013) has shown that the measurement of energy consumption can be done by measuring the usage of the machine. So there is direct relation between Load on a system and energy consumption.

Figure 4 explains the Architecture of HadoopCL cluster. In HadoopCL cluster, at each node a task runner is running which is responsible for the management of generated child JVMs. Each JVM executes Mapper and Reducer in heterogeneous environment isolated form Hadoop environment. Each JVM is assigned a single HDFS block in the form of Key/Value pair. Each child JVM can execute either Map or Reduce by the use of two java threads: Main Java Thread and Child Java Thread.

Main Java Thread takes input from HDFS blocks, copy these blocks to devices, start executing these blocks using Mapper and Reducer kernel, retrieves the output from the device and sending the produced output to the dedicated communication thread.

Child Java Thread takes the output from the devices and produces the output to the HDFS.

APARAPI tool is used to convert user-written java byte code generated after compilation to OpenCL kernels. These OpenCL kernels are generated for both the map and reduce method.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

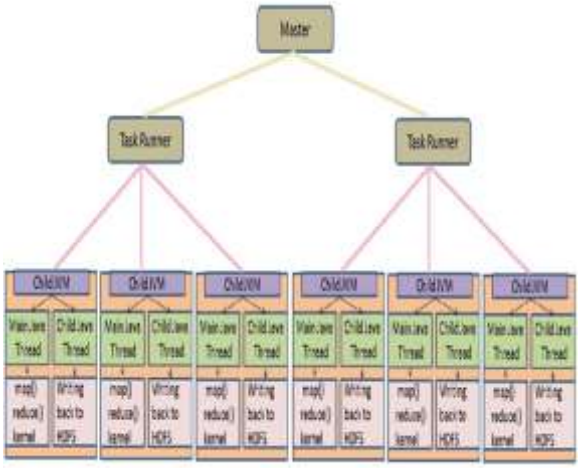

Figure 4 : Architecture of HadoopCL cluster

An example to understand working of HadoopCL is done through the dataset of review file of Hotels. The dataset consists of number of hotels in which enormous number of reviews is given by the users. Each hotel review file is having number of reviews given by various users.

Figure 5 explains the flow chart of execution of single hotel's review on a single node of Hadoop cluster. It shows that, a single review (review 1) is when given to the Mapper, at Mapper, heterogeneous environment will come into picture. The authors or re viewers of the hotel is given as input to the work items of GPUs. At each work item, the various algorithms run such as: porterStemmer(), keywordExtractor(), similarityComputation(), personalizedRating() etc. So this brings the parallelism at thread level.

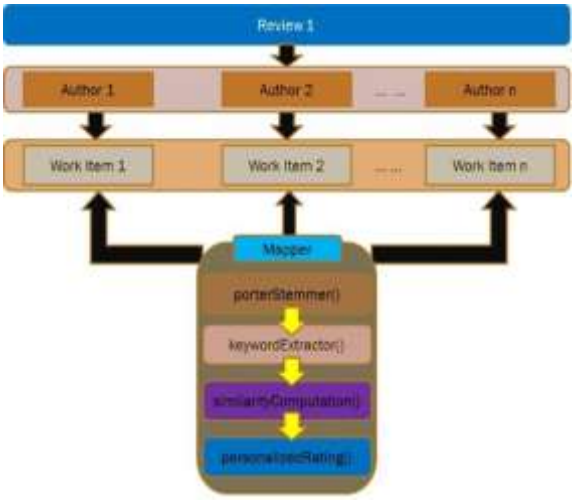

Figure 5 : Flow Chart of execution of Review of a Hotel on single node

III. EXPERIMENT AND RESULT

The machine used for the processing having the specification described in Table: 4.1 as follows:

Resources	Descriptions
Processor	INTEL i7 processor with 3.4GHz clock rate

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

RAM	8 GB/machine
Graphics Card(GPU)	2 GB GPU
Operating System	Linux/Ubuntu 14.04
Hard Disk	1 TB
Topology	Connected by gigabit Ethernet cable

Table: 2 Configuration of cluster's machines

The above description shown in Table 2 explains the configuration used on each machine in a cluster. Every single machine is installed with INTEL i7 processor with 3.4 GHZ clock rate, 8 GB RAM, 2 GB GPU, 1 TB hard disk etc.

DATASET USED

Technically, our experiments are conducted in a Hadoop as well as HadoopCL platform. To evaluate the performance of KASR on both Hadoop and HadoopCL, the experiment has been performed on three types of dataset from hotel recommendation system. These datasets varies in sizes. Table 3 explains the sizes of the datasets, frameworks to handle these datasets and the environment used in order to handle these datasets. These datasets consists of hotel reviews given by the previous users.

EXPERIMENTAL RESULT

In this chapter the experimental results are described, analyzed and compared based on their execution time. The experiments are conducted on three frameworks; KASE_ESC on Java, Hadoop and HadoopCL. Then comparison of results is done based on execution time taken by KASE_ESC on Java, Hadoop and HadoopCL framework on different number of nodes. The experimental results are obtained by performing 20-20 iterations of execution of KASR on 11.5 GB of dataset, 1.06 GB of dataset and 241 MB of datasets on both frameworks; Hadoop and HadoopCL on a cluster with 2 nodes, 4 nodes and 6 nodes. Then the average of these 20 iterations is taken and final results are obtained and used for comparing the execution time and calculating the overall speedup obtained by HadoopCL over KASE_ESC on Java and Hadoop with different number of nodes on different sizes of datasets.

Frameworks	Descriptions	Size of the Datasets
KASE_ESC on Java	Java framework executes these datasets in sequential manner. Increase in the size of data decreases the performance of Java.	11.5 GB
		1.06 GB
		241 MB
Hadoop	Hadoop framework executes these datasets in distributed manner. Increase in the size of data increases the	11.5 GB
		1.06 GB

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

	performance of Hadoop.	241 MB
HadoopCL	HadoopCL framework executes these datasets in distributed and heterogeneous manner. Increase in the size of data increases the performance of HadoopCL.	11.5 GB
		1.06 GB
		241 MB

Table: 3 Datasets Description

Table 4 explains the base paper implementation of KASR using KASE_ESC on Java and Hadoop cluster of 2, 4 and 6 nodes. The execution time is calculated on three different sizes of datasets. Then a comparison graph as explained in Figure: 6 is made on logarithmic scale which is a comparative analysis of execution time between KASR using KASE_ESC on Java and KASR on Hadoop. The overall speedups gained by Hadoop over KASE_ESC on Java on mentioned datasets are 3.881793138, 14.67046201 and 42.28880319 on 2 nodes, 6.058237248, 19.94344396 and 67.29138621 on 4 nodes and 7.190906982, 21.45029614, 79.78028811 on 6 nodes of Hadoop cluster.

Table 5 explains the comparison between execution time taken by KASR on Hadoop and KASR on HadoopCL with different number of nodes on 11.5 GB dataset. On executing this dataset on different nodes, it can be seen that the execution time taken by HadoopCL is much less than Hadoop. Moreover, by increasing the number of nodes, the execution time taken by HadoopCL decreases faster in proportion than execution time decrease in Hadoop. A graph, Figure: 6, of comparison between execution time taken by KASR on Hadoop and HadoopCL are made which executes on the dataset of size 11.5 GB. This dataset is executed on 2 nodes, 4 nodes and 6 nodes of cluster of Hadoop as well as HadoopCL. The overall speedups gained by HadoopCL over Hadoop on 11.5 GB dataset are 1.657325467, 1.36275701 and 1.714829032. So the overall speedup of HadoopCL over Hadoop is 1.7x when KASR is performed on Hadoop and HadoopCL on 11.5 GB data. If the number of nodes is increased and the size of dataset is increased the speedup will also increase.

	Execution Time taken by Frameworks (in milliseconds)			
	KASE_ESC on Java	Hadoop		
Size of the Datasets	Single Node	2 Nodes Cluster	4 Nodes Cluster	6 Nodes Cluster
11.50 GB	3849021	91017	57199	48245
1.06 GB	134105	9134	6719	6247
241 MB	31227	7986	5117	4311

Table: 4 Comparison between KASR using KASE_ESC on Java and Hadoop

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

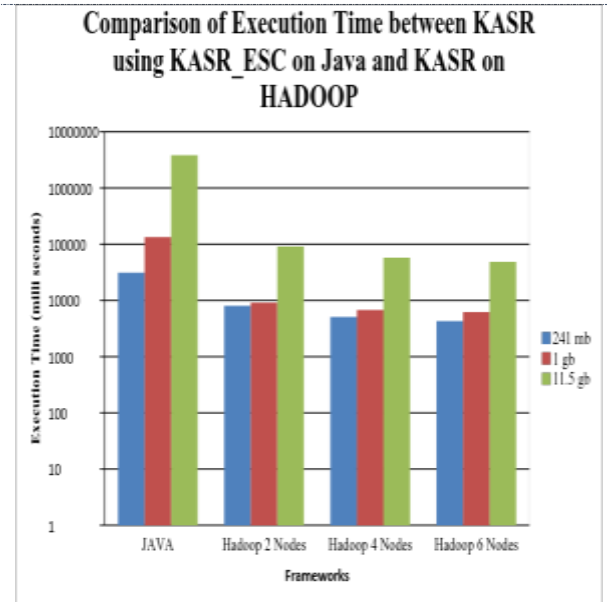

Figure 6 : Comparison graph between KASR on Java and Hadoop

Number of Nodes in a Cluster	Execution Time taken by Frameworks (in milliseconds)	
	Hadoop	HadoopCL
2 Nodes	91017	54918
4 Nodes	57199	41973
6 Nodes	48245	28134

Table: 5 Comparison between Hadoop and HadoopCL on 11.5 GB dataset

Table 6 explains the comparison between execution time taken by KASR on Hadoop and KASR on HadoopCL with different number of nodes on 1.06 GB of dataset. A graph, Figure: 7, of comparison between execution time taken by KASR on Hadoop and HadoopCL are made which executes on the dataset of size 1.06 GB. This dataset is executed on 2 nodes, 4 nodes and 6 nodes of cluster of Hadoop as well as HadoopCL. The overall speedups gained by HadoopCL over Hadoop on 1.06 GB dataset are 1.351783336, 1.20563431 and 1.24516643.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

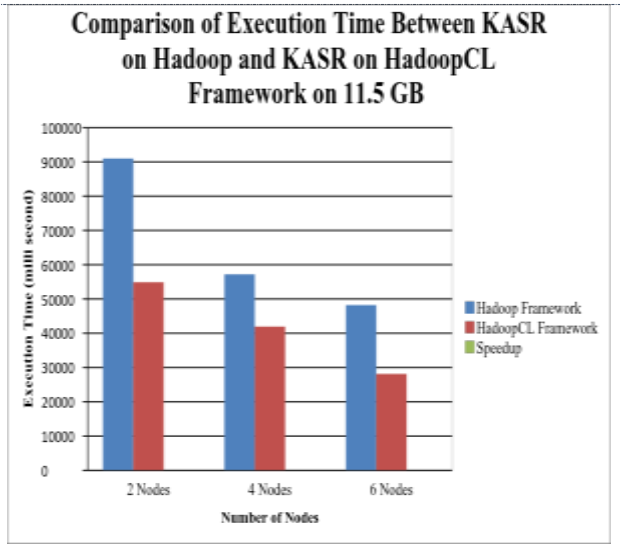

Figure 7 : Comparison graph between Hadoop and HadoopCL on 11.5 GB dataset

Number of Nodes in a Cluster	Execution Time taken by Frameworks (in milliseconds)	
	Hadoop	HadoopCL
2 Nodes	9134	6757
4 Nodes	6719	5573
6 Nodes	6247	5017

Table: 6 Comparison between Hadoop and HadoopCL on 1.06 GB dataset

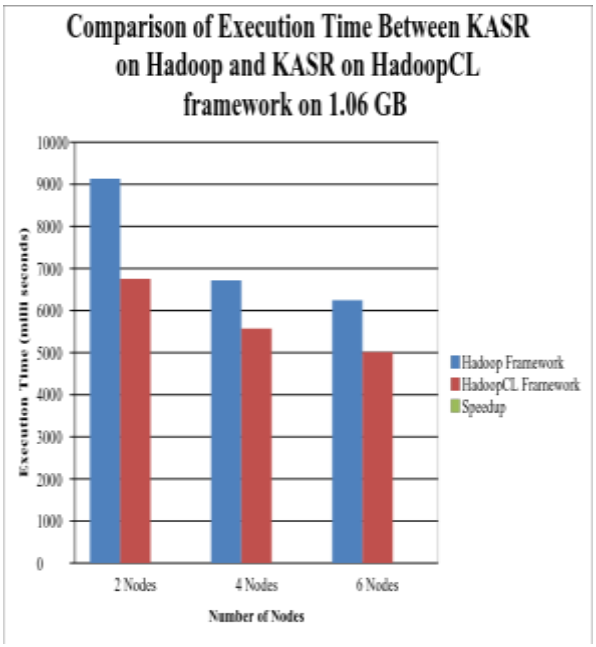

Figure 8 : Comparison graph between Hadoop and HadoopCL on 1.06 GB dataset

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

Number of Nodes in a Cluster	Execution Time taken by Frameworks (in milliseconds)	
	Hadoop	HadoopCL
2 Nodes	7986	5679
4 Nodes	5117	4998
6 Nodes	4311	4049

Table: 7 Comparison between Hadoop and HadoopCL on 241 MB dataset

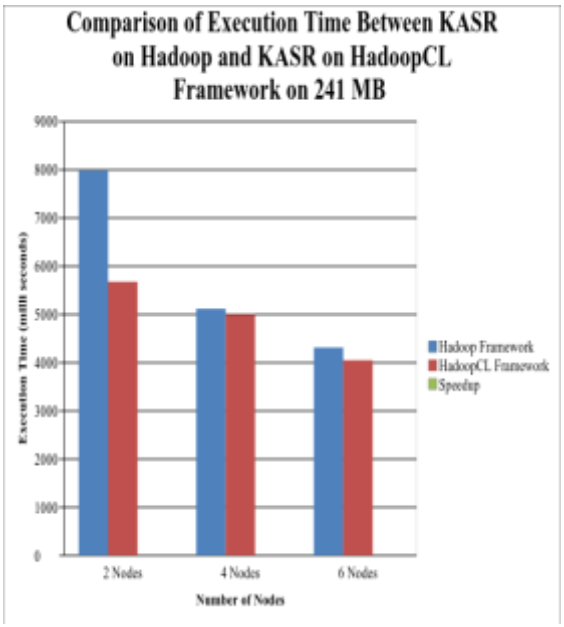

Figure 9: Comparison graph between Hadoop and HadoopCL on 241 MB dataset

A graph, Figure: 9, of comparison between execution time taken by KASR on Hadoop and HadoopCL are made which executes on the dataset of size 241 MB. This dataset is executed on 2 nodes, 4 nodes and 6 nodes of cluster of Hadoop as well as HadoopCL. The overall speedups gained by HadoopCL over Hadoop on 241 MB dataset are 1.406233492, 1.023809524 and 1.064707335.

Finally Table: 8 explain the speedup of the Keyword-Aware Hotel Recommendation System using HadoopCL over Hadoop at 2 nodes, 4 nodes and 6 nodes of cluster at 241 MB, 1.06 GB and 11.5GB of datasets. Figure: 10 shows the graph which gives the comparative analysis of speedups achieved by using HadoopCL over Hadoop at different datasets using different nodes of cluster.

Different Sizes of The Datasets	Speedup on 2 nodes of cluster	Speedup on 4 nodes of cluster	Speedup on 6 nodes of cluster
241 MB	1.406233492	1.023809524	1.064707335
1.06 GB	1.351783336	1.205634308	1.245166434
11.5 GB	1.657325467	1.36275701	1.714829032

Table: 8 Comparative analyses of speedups achieved by HadoopCL over Hadoop

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

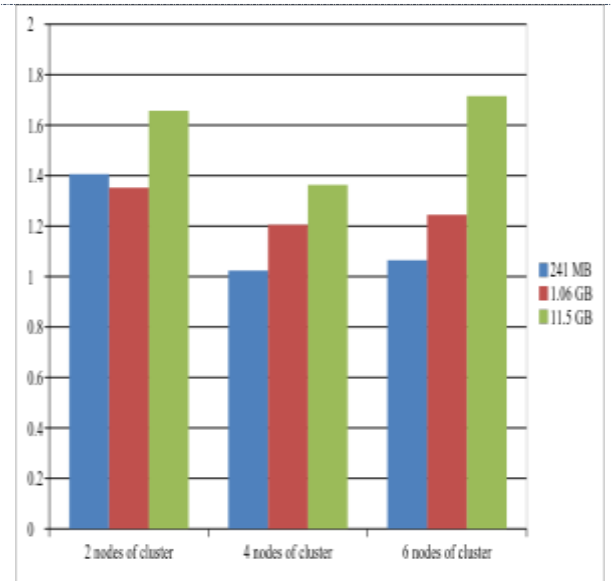

Figure 10 : Comparative graph of speedups achieved by HadoopCL over Hadoop

Thus for a larger dataset, the result provided by HadoopCL is more accurate and faster than Hadoop. For huge datasets and more number of nodes in a cluster the results produced are more accurate, efficient and faster system. So, the overall performance of the system improves.

The results and discussion may be combined into a common section or obtainable separately. They may also be broken into subsets with short, revealing captions.

IV. CONCLUSION

A user-based Collaborative Filtering algorithm is used to produce appropriate recommendation list of hotels based on user's preferences. KASR implementation on Hadoop environment results in improvement in scalability and efficiency to handle big data but it doesn't utilize resources properly as it doesn't utilize the cores of the CPU and GPUs which leads to poor computational performance. HadoopCL, which is distributed as well as heterogeneous in nature, provides inter-node parallelism as well as intra-node parallelism to utilize the resources properly in terms of cores of CPU and GPUs. The experimental results show that overall speedup and performance is improved by using HadoopCL over Hadoop. It also shows that, the more number of nodes in a cluster and the bigger size of datasets once executed by HadoopCL provides more efficient system and provides better performance with overall speedup from 1.3x to 1.7x in comparison to Hadoop at 6 nodes of cluster. Since the load on a system is reduced by providing parallelism at thread level, the energy consumption is reduced. Future work specific to HadoopCL would include extensions to the APARAPI runtime and translator which may decrease code changes required for porting to HadoopCL and improves performance. Also Future work specific to KASR would include how to distinguish the positive and negative preferences of the users from their reviews to make the predictions more accurate.

ISSN: 2277-9655
Impact Factor: 4.116
CODEN: IJESS7

V. REFERENCES

[1] Resnick, P., Iacovou, N., Suchak, M., Bergstrom, P., Riedl, J.: GroupLens: an open architecture for collaborative filtering of netnews. In: Proceedings of the 1994 ACM conference on Computer supported cooperative work (CSCW '94), pp. 175–186. ACM, New York (1994)

[2] Shardanand, U., Maes, P.: Social information filtering: algorithms for automating "word of mouth". In: Katz, I.R., Mack, R., Marks, L., Rosson, M.B., Nielsen, J. (eds.) Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '95), pp. 210–217. ACM Press/Addison-Wesley Publishing Co., New York (1995)

[3] A. Tuzhilin Et al. "Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-art and Possible Extensions," IEEE Transactions on Knowledge and Data Engineering, Vol. 17, No 6 pp. 734-749, 2005

[4] S Meng. Et al . "KASR: A Keyword-Aware Service Recommendation Method on MapReduce for Big Data Applications" IEEE Transactions on Parallel and Distributed Systems (Volume: 25, Issue: 12: Dec. 2014)

[5] B. Issac and W. J. Jap, "Implementing spam detection using bayessian and porter stemmer keyword stripping approaches," TENCON 2009-2009 IEEE Region 10 Conference, pp. 1-5, 2009

[6] A. Chu et al. "A comparison of two methods for determining the weights of belonging to fuzzy sets", Journal of Optimization Theory and Applications. Vol 27, No 4. Pp. 531-538, 1979.

[7]] P. Castells, M. Fernandez, and D. Vallet, "An Adaptation of the Vector-Space Model for Ontology-Based Information Retrieval," IEEE Transactions on Knowledge and Data

[8] Y. Zhu, Y. Hu, "Enhancing search performance on Gnutella-like P2P systems, "IEEE Transactions on Parallel and Distributed Systems. Vol. 17, No. 12, pp. 1482-1495, 2006.

[9] Dumitrel Loghin, Lavanya Ramapantulu, Yong Meng Teo "An Approach for Energy Efficient Execution of Hybrid Parallel Programs" in 2015 IEEE International Parallel and Distributed Processing Symposium

[10] SungYe Kim, Jeremy Bottleson, Jingyi Jin, Preeti Bindu " Power Efficient MapReduce Workload Acceleration Using Integrated-GPU", in IEEE First International Conference on Big Data Computing Service and Applications (Big Data Service),2015 pp. 162 - 169.

[11] Motahar Reza, Aman Sinha, Rajkumar Nag, Prasant Mohanty "CUDA-enabled Hadoop cluster for Sparse Matrix Vector Multiplication" in 2015 IEEE 2nd International Conference on Recent Trends in Information Systems (ReTIS)

[12]] Max Grossman, Mauricio Breternitz, Vivek Sarkar "HadoopCL: MapReduce on Distributed Heterogeneous Platforms through Seamless Integration of Hadoop and OpenCL" in 2013 IEEE 27th International Symposium on Parallel & Distributed Processing Workshops and PhD Forum

[13] Tiwari J, Pawar M, Pandey A, An Automated Complex Word Identification from Text: A Survey. Orient J. Comp. Sci. and Technology;10(3)

CITE AN ARTICLE

Tiwari , J., Pawar, M., Dr, & Pandey, A., Dr. (2017). A HADOOP BASED COLLABORATIVE FILTERING RECOMMENDER SYSTEM ACCELERATED ON GPU USING OPENCL. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*, *6*(9), 195-209. Retrieved September 5, 2017.