Journal of Big Data

RESEARCH Open Access

# Gapprox: using Gallup approach for approximation in Big Data processing

Hossein Ahmadvand*, Maziar Goudarzi and Fouzhan Foroutan

*Correspondence: ahmadvand@ce.sharif.edu
Department of Computer Engineering, Sharif University of Technology, Tehran, Iran

**Abstract**

As Big Data processing often takes a long time and needs a lot of resources, sampling and approximate computing techniques may be used to generate a desired Quality of Result. On the other hand, due to not considering data variety, available sample-based approximation approaches suffer from poor accuracy. Data variety is one of the key features of Big Data which causes various parts of data to have different impact on the final result. To address this problem, we develop a data variety aware approximation approach called Gapprox. Our idea is to use a kind of cluster sampling to improve the accuracy of estimation. Our approach can decrease the amount of data to be processed to achieve the desired Quality of Result with acceptable error bound and confidence interval. We divide the input data into some blocks considering the intra/inter cluster variance. The size of the block and the sample size are determined in such a way that by processing small amount of input data, an acceptable confidence interval and error bound is achieved. We compared our work with two well-known state of the art. The experimental results show that our result surpasses the state of the art and improve processing time up to 17× compared to ApproxHadoop and 8× compared to Sapprox when the user can tolerate an error of 5% with 95% confidence.

**Keywords:** Data variety, Quality of Result, Approximation, Cluster sampling

## Introduction

Despite the huge number of researches in Big Data area, approximate computing in this area still remains a challenge. The approximation is used for reduction of resources such as time, cost or energy. Applications that analyze the input data, logs and queries to generate aggregated results or dashboards can benefit from approximation techniques in Big Data. In these applications, the output is much smaller than the input. This fact indicates that approximation can be used for increasing the processing performance for this kind of computation. Data skew causes reduction of performance in approximation. Data skew has many causes [1–3]. In this paper, we focused on the challenge that stems from variety. Data variety can be created by aggregating input data from multiple sources with different statistical distribution and uneven distribution of input data. This uneven distribution causes the data skew.

There are many approaches that have addressed the sample-based approximation. In case of lack of resources, approximate computing is a suitable approach for generating acceptable Quality of Result/Service [4–9]. The approximation can be used in

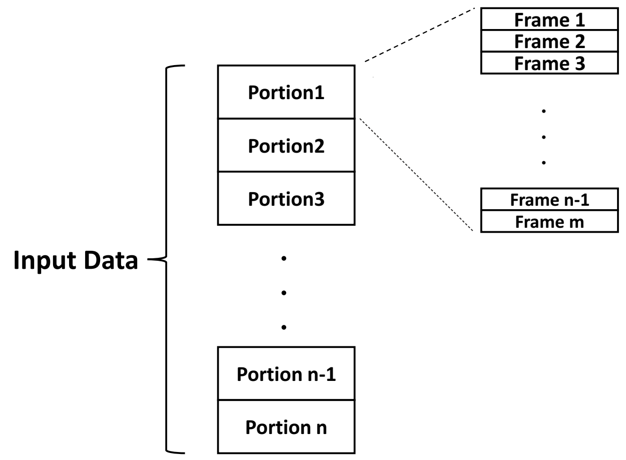

**Fig. 1** Blocks and frames design

wide range of applications such as image, audio or video processing, data analytics, Monte Carlo computation and machine learning processing [10–14].

Unfortunately, some of them do not consider data variety/skew. Some other process a large number of samples to achieve the desired error bound and confidence interval.

*ApproxHadoop* [15] is a well-known state of the art of our work. The error bound and confidence interval of its sampling approach is good when there is no variety in data. But in the real world and real datasets, a large volume of variety and skew exists. ApproxHadoop considers a large cluster size and produces very small samples. In the case of existing data variety in input data, this method may suffer from inefficient sampling and large variance.

We also have considered another well-known work for our evaluation. In *Sapprox* [16], the authors have used the cluster sampling with unequal probability to consider the distribution of input dataset. Implementation of this approach is not simple and may cause some inefficiency.

Based on the description presented, we have focused on the weakness of these two approaches and presented an approach to cover their weakness. Like Approxhadoop and Sapprox, we also have used cluster sampling for our goal. We select a suitable size as the block size in a way that inter-block variance decreased and intra-block variance increased. By obtaining a sample from each block, the error bound is decreased. By using this approach we can achieve the suitable Quality of Result by processing a small volume of the input dataset.

In our previous work [17], we have shown that data variety has a great impact on the performance of progressive processing. In this paper, we present a sample-based approximation method with high accuracy in the case of data skew/variety. For this goal, we present *Gapprox* to increase the accuracy of approximate processing. We have offered a kind of cluster sampling for increasing performance of sample-based approximation. As Fig. 1 shows, in our approach, we divide the input data into some blocks and then divide each block into some same sized frames. We select the block size in such a way that the inter block variance is increased and intra block variance is decreased. By this approach, the acceptable Quality of Result is achieved by

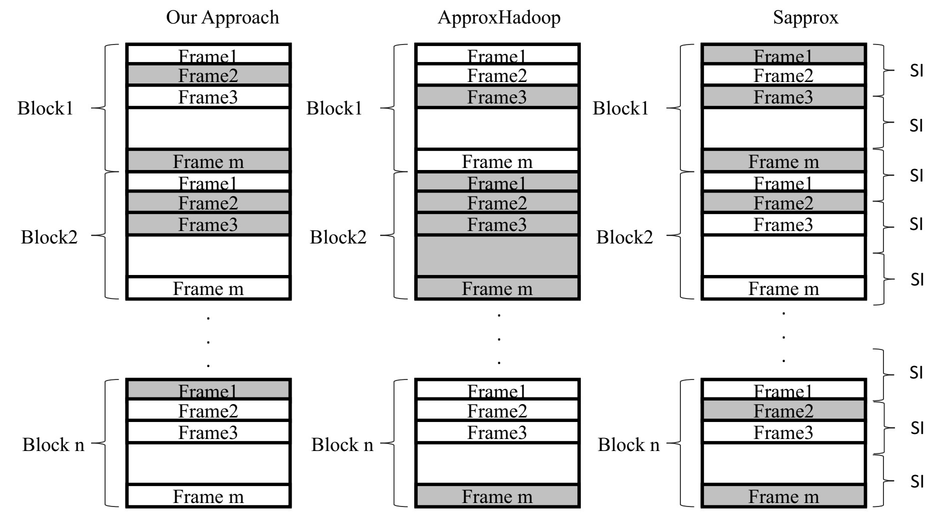

**Fig. 2** Our approach in comparison with other approaches

processing a very small amount of input data. Block and frame design are shown in Fig. 1.

We show the difference between approaches in Fig. 2. Approxhadoop selects some blocks and some frames in it as the sample. Sapprox uses sample interval for getting samples. Our approach divides the input data into some same size blocks and each block into some same size frames. In each block samples are selected among frames with certain error bound and confidence interval.

*Weaknesses of ApproxHadoop*

- The authors in ApproxHadoop have not considered the data variety/Skew. They have evaluated their approach by uniform datasets. In the case of existing an impressive amount of skew in data, the users get forced to process a large amount of input data to achieve acceptable Quality of Result.

*Weaknesses of Sapprox*

- In the beginning, the amount of sampled data to achieve acceptable error bound and the confidence interval is not definite. This weakness forces the user to process more data that the needed.
- Sapprox is very difficult for implementation in case of NoSQL datasets. In this kind of datasets determination of sampling interval is very hard for users. This fact causes inefficiency in case of NoSQL datasets.

*Advantages of our Approach*

- We have considered different skew conditions in Gapprox. In the case of uniform distribution, a negligible amount of input data must be processed to achieve the acceptable error bound and confidence interval. In case of uneven distribution, our approach is able to manage this issue by dividing input data into some blocks and using sampling with a guarantee. By this technique, we can tolerate the weakness of ApproxHadoop.

- Our approach can achieve sampling targets by processing a low amount of data (less than 1%) in each condition of skew. Our approach can estimate the amount of data that must be processed before starting the process. Due to the techniques that we have used, our approach can achieve an acceptable Quality of Result by processing less data than Sapprox.
- Our approach can produce targeted Quality of Result in any condition of data skew by processing less than 1% of input data. But based on the experimental results, in case of uneven distribution, a large and undefined amount of data must be processed in ApproxHadoop.
- Our approach is applicable for SQL and NoSQL datasets and efficiently generate acceptable Quality of Result in case of both.
- We have used some techniques and statistical methods to help the user to implement the approach. We have used Simple Random Sample to determine the primary condition of data skew. We also divide data into some equal parts and present a simple approach to implement, in compare to Sapprox.

*We proposed three techniques in our approach*

1. Defining suitable size for the block. Block size should be defined in a way that the intra block variance is minimized.
2. Input data sampling. Only a subset of input data is processed. We use a sampling technique with a 95% confidence interval and 5% error bound.
3. User-defined approximation. In our work user can decide the level of approximation which the generated outcome will have.

We evaluate our work by using some well-known datasets. We also used the TPC Benchmark and Amazon review dataset for our evaluations [18, 19]. We compared our work with two well-known previous works as the state of the art: ApproxHadoop [15] and Sapprox [16].

*Contributions* In this paper, we have the following contributions:

1. We proposed a variety/skew aware approach for using an approximation in Big Data processing. That is practical and easy to use for users.
2. We use some statistical theories for sampling with an acceptable confidence interval and error bound in case of existing skew/variety in data.
3. We show how sampling theories can be used to estimate sample size and compute error bounds for approximations in MapReduce-like systems.
4. Via experimental results, we show that our approach can significantly reduce processing time compared to our state of the arts.

*Organization* The rest of the paper is organized as follows: "Related works" section presents an overview of the state of the arts and previous works including a classification of methods and tools. In "Background" section we introduced the background of our work. "Motivation" section presents the overall motivation of our work. "Methods" section describes the system design and experiments. The experimental result and discussion is presented in "Results and discussion" section, and finally, "Conclusions and future work" section extracts the main conclusions of the paper and proposes future work.

## Related works

*Approximation* Approximate computing is used in large-scale computing in case of lack of processing resources [4–9]. This kind of processing can be used for a wide range of application types including data analytics, machine learning, Monte Carlo computations, and image, audio or video processing [10–14]. The approximation can be used in case of the lack of resources such as time, cost or energy [9, 15, 20]. In [17] we also have considered data variety and have the benefit from it to develop efficient progressive processing. The acceptable quality of approximation results is achieved by using our approach.

The authors in [9] have presented a survey of techniques for approximate computing. They discussed strategies and approaches for approximation and monitoring output quality. They have considered different processing units (e.g., CPU, GPU and FPGA), processor components memory techniques and different framework for approximation computing. The authors in [20] have improved the performance of Big Data processing by using a kind of MapReduce that supports the early return from a job as it is being computed. ApproxHadoop [15] is well-known research in Big Data approximate computing. The authors have presented a framework for approximate computing by using sampling. Their framework has good performance in case of uniform distribution. But our analysis shows that ApproxHadoop does not have acceptable performance in case of existing data variety or input data with real distribution. The authors have paid attention to the weakness of Approxhadoop in [16]. They have considered data skew in their research. They present an approach with unequal probability sampling. Their solution has better performance in compare to ApproxHadoop. But this approach also has some weaknesses about NoSQL datasets and complexity of implementation. The authors have presented an approximate computing technique in [21] by learning how an approximation can be applied in code regions. They have used Neural Network techniques to learn how to reduce energy consumption by replacing the code regions.

We also have presented an approximation framework for Big Data processing. Our approach is more efficient than ApproxHadoop and more flexible than Sapprox. The authors in ApproxHadoop do not consider the data skew/variety. The plan that presented in Sapprox is hard to implement for NoSQL datasets and achieved to the acceptable Quality of Result by processing more data than Gapprox.

*Sample-based approximation approaches* Approaches and researches in approximate query processing are presented in [22]. BlinkDB [10] offers a solution to select samples for each query column set and uses samples to answer online queries in the distributed file system. Offline sampling can use preprocessing and a priori knowledge of dataset. Traditional sampling approach like [10, 23–25], that present uniform-based-sampling are unreliable in case of data skew. These methods select more tuples for big groups and enough tuples in rare groups. But in the case of data variety, the size of the sample group and the values in each sample group may be highly skewed.

Agarwal et al. [26], Pol and Jermaine [27], Zeng et al. [28], Zeng et al. [29] used closed-form estimation and Bootstrap method for providing high confidence interval. They repaired the results by using resampling. The authors in [30] with the knowledge of data distributions use error-bounded stratified sampling and can reduce the sample size. The authors in [31] offer a solution for approximate computing for more kinds of data besides relational data. Researchers in [32] use online uniform HDFS samples and by using Bootstrap method incrementally evaluate the accuracy. This method can be used efficiently for complex queries. But the researchers unlike us did not consider the data variety and skew. Authors in *ApproxHadoop* [15] consider the datasets with uniform distributions. But real datasets are not uniformly distributed. Random sampling in this kind of datasets generate biased sampling and does not work very well. The authors in *Sapprox* [16] have noted the weakness of ApproxHadoop [15] and tried to offer a way to overcome it. They have used cluster sampling with unequal probability theory. It is very hard for users to define suitable *Sampling Interval* for the various condition of skew. Our approach can achieve acceptable Quality of Result with processing fewer amount data than Sapprox. A cluster-based sampling approach is presented in [33]. This approach combines three kinds of sampling: uniform sampler, distinct sampler, and universal sampler. This method uses Horvitz–Thompson (HT) to estimate the exact answer and uses the central-limit theorem to compute the confidence interval. It also uses "dominance transitivity" before using the estimator. The authors in [34] have applied sampling to MapReduce facilitate analysis of various film details, review comments and users profile.

We have also presented a sample based framework for Big Data processing that can be used in case of data skew/variety. Previous approaches that are presented in this section do not consider generating acceptable Quality of Result in any condition of data skew and any types of dataset. In some approaches, we must have some information about input data distribution. In some other, the authors consider uniform distribution. There is no sensitivity about NoSQL datasets. So, Gapprox is generally different from previous approaches.

*Distributed approximation systems* The authors in [10] present an approximate query processing system that provides an efficient strategy to get a sample from the dataset. Sapprox [16] presents a more flexible approach than BlinkDB and efficient than ApproxHadoop [15]. We also present an approach that is more efficient than ApproxHadoop and more flexible than Sapprox. Quickr [33] is also designed presented ad-hoc queries on big-data clusters like BlinkDB [10], ApproxHadoop and Sapprox. Quickr [33] does not need to discover all dataset to achieve a certain confidence interval. It has been used by Microsoft's search engine, i.e., Bing. Microsoft also develop and use a framework for progressive analytics "Now!" and use it for Windows Azure [35]. SnappyData [36] is a Spark-based system for delivering interactive analytics. FluoDB [37] is another Spark-based approach to support general (online analytical processing) OLAP queries. XDB [38] is another system that can support online aggregation. This approach support complex queries including join operator. We also have presented an approach that can be used in any condition of data skew. Our approach can be used in case of uniform or uneven distribution. None of the above solutions can perform acceptable Quality of Result in any condition of skew and input data types. So, our approach is different from them.

*Variation* We present a sample-based approximation in this paper. Our approach improved the Quality of Result by processing a small amount of input data. The authors in [39] have shown how approximation techniques can be used to address network and processing variations. They have used an approximation to reduce computation requirement without a reduction in Quality of Result. The authors in [40] investigated the problem of imbalanced sub-datasets and inefficient sampling on sub-datasets over a Hadoop cluster. They have mentioned that uneven sub-datasets distribution leads to lower-performance in parallel data analysis. They have offered a framework to balance workload among computational nodes. The authors in [41] consider the data skew to reduce the finishing time and computational resources. They have modeled different degrees of skew in the input data and provided experimental results under different conditions. The authors in [42] have presented a framework to increase data-aware scheduling for jobs with combinatorial choices. They have considered intermediate data transfer for locality and balancing. The authors in [43] have presented a survey of techniques for approximate computing. They have discuss different approaches to find program portions that are suitable for approximation and monitor Quality of Results. The authors in [44] have presented a framework to mitigate the skew in Big Data processing applications. They have mitigated the skew of two sources: (1) Skew due to an uneven distribution of input data, and (2) the skew caused by some subset of data taking longer to process than other. The authors in [45] have surveyed Big Data processing platforms. They have divided the Big Data processing platforms into 2 main categories: (1) Horizontal Scaling that also called Scale out and (2) Vertical scaling that also called Scale-up. Using more distributed servers is considered in Scale-out strategy. In Scale up approach, current processing element such as CPU and memory is replaced by more powerful ones. We have also presented an approach to scale up for big data processing in [17] considering the data variety.

None of the above researches have not yet presented a framework to increase the accuracy and reduce the amount of data to be processed to achieve acceptable Quality of Result. ApproxHadoop and Sapprox are two works that can be compared with our approaches. We show our work's advantage in the case of efficiency and flexibility in compared with them.

## Background

In this section, we present some tools, topics and statistical methods that will be noted in this paper.

*Spark* Apache Spark is an open-source cluster-computing framework [46]. Originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since then. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.

*RDD* RDDs are fault-tolerant, parallel data structures that let users explicitly persist intermediate results in memory, control their partitioning to optimize data placement, and manipulate them using a rich set of operators [47–49].

We use a command in Scala language for getting suitable size sample from RDDs.

*Multi-stage sampling* We leverage multi-stage sampling [50] to compute error bounds for approximate MapReduce applications that compute aggregations (e.g., counting accesses to Web pages from a log file). The set of supported aggregation functions includes sum, count, average, and ratio. For simplicity, we next discuss two-stage sampling. Depending on the computation, it may be necessary to use additional sampling stages as discussed at the end of the section.

Suppose we have $T$ units in $N$ clusters. Each cluster contains $M_i$ units. Suppose further that each unit $j$ in cluster $i$ has an associated value $V_{ij}$, and we want to compute the sum of these values.

To compute an approximate sum, we can create a sample by randomly choosing $n$ clusters, and then randomly choosing $m_i$ units from each chosen cluster $i$. Two-stage sampling then allows us to estimate the sum from this sample as:

$$\tau = \frac{N}{n}\sum_{i=1}^{n}\left(\frac{M_i}{m_i}\sum_{j=1}^{m_i} v_{ij}\right) \pm \in \tag{1}$$

$$\in = t_{n-1,1-\alpha/2}\sqrt{Var(\tau)} \tag{2}$$

$$Var(\tau) = N(N-n)\frac{S_u^2}{n} + \frac{N}{n}\sum_{i=1}^{n} M_i(M_i - m_i)\frac{S_i^2}{nm_i} \tag{3}$$

where $(s_u^2)$ is the inter-cluster variance (computed using the sum and average of the values associated with units from each cluster in the sample), $(s_i^2)$ is the intra-cluster variance for cluster $i$, and $t_{n-1,1-\alpha/2}$ is the value of the Student t-distribution with $n-1$ degrees of freedom at the desired confidence $1-\alpha$.

*Cochran sampling* The Cochran formula allows you to calculate an ideal sample size given a desired level of precision, desired confidence level, and the estimated proportion of the attribute presented in the population [51]. Cochran's formula is considered especially appropriate in situations with large populations. A sample of any given size provides more information about a smaller population than a larger one, so there's a 'correction' through which the number given by Cochran's formula can be reduced if the whole population is relatively small.

*Aggregation Applications* Many companies are used Aggregation Applications widely in transaction operations, Business calculations and Analytical computations. An aggregate function is a function that computes multiple values grouped together and forms final value. In this paper, we present an approach to increase the accuracy of approximation in aggregation applications. This kind of applications can benefit from approximation. As a real and concrete example, the popularity of individual Web pages is important for the Web site operators. They use logs of their Websites for this information. In this kind of computation, the relative popularity is more important than the exact access.

## Motivation

*Motivation* We consider Data Variety in our work and show the difference of results when different amount of skew exists. We have used bootstrapping method to generate 12 GB data without Data Variety and with uniform distribution [52]. We use this uniform input data for the first experiment. We also have used an input with 12 GB size for our second experiment that is collected from three sources. We consider IMDB,

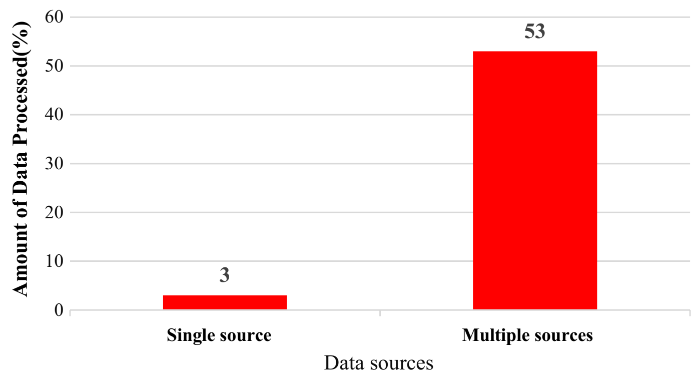

**Fig. 3** Amount of data processed to meet sampling requirements

Gutenberg, and Quotes and get 4 GB from each of them [53–55]. We show the result of this experiment in Fig. 3. The vertical axis represents the "*amount of data processed* "to meet the sampling requirements, and the horizontal axis represents the *"Approaches"*.

ApproxHadoop is a well-known state of the art. We study the functionality of ApproxHadoop in two conditions: existing and not existing data variety. As discussed before multiple resources increase data variety and uneven distribution. Figure 3 shows the amount of data processed to meet the acceptable confidence interval (95%) and error bound (5%). In case of not existing data variety, ApproxHadoop has acceptable performance and by processing less than 1% of input data the sampling requirement is achieved. But ApproxHadoop has achieved expected Quality of Result by processing a large amount of input data as sample. In this case, more than a half of the input data must be processed. The authors in ApproxHadoop have assumed that input data is uniformly distributed. Some reasons such as aggregating data from multiple sources may cause an impressive amount of variety in most real datasets. We have considered this fact in our research.

Our approach is a kind of multistage sampling method. Multistage designs are used in many practical cases [56]. For example in *Gallup method,* first approximately 300 election districts used in sampling. At the second stage, they select 5 households per district. We employ the Gallup idea in our work and consider Data Variety in our two-stage sampling.

## Methods

Based on the motivation presented in "Motivation" section, in the case of Data Variety, previous approaches do not have acceptable performance. So, we offer our approach and discuss our system design in this section. As Fig. 4 shows, we divide the input data into some same sized blocks and then divide each block into some same sized frames. As discussed in "Results and discussion" section, we used multi-stage sampling in this paper. We consider variance of blocks as inter-block variance ($S_u$) and variance of frames as intra-block variance ($S_i$).

In our approach, based on formula 3, there are 4 approaches to reduce sample size for achieving the acceptable result:

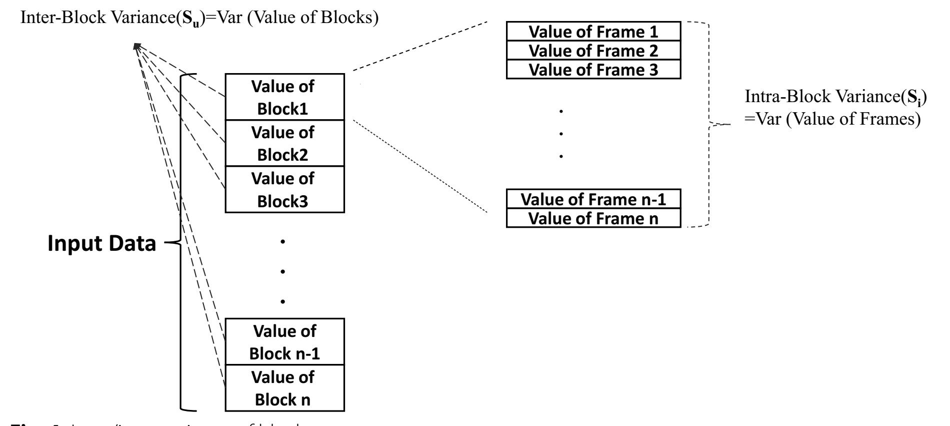

**Fig. 4** Inter/intra variance of blocks

1. Minimizing $S_u$: For minimizing $S_u$, we should consider bigger size as the Block Size. This decision causes a bigger value for $S_i$.
2. Minimizing $S_i$: For minimizing $S_i$, we should consider lower size as the Block Size, this decision causes bigger value for $S_u$.
3. Processing all blocks (n = N): In case of selecting all blocks, the first term of Eq. (3) will be zero.
4. Processing all frames: In case of selecting all frames in selected Blocks, the second term of Eq. (3) will be zero.

*Decision 1* If we select 1 and try to minimize the $S_u$, the Block Size would have big value and it increases $S_i$. So, based on formula 3, for minimizing the data variance we should try to select all frames in selected Blocks for processing. Based on the big volume of Blocks in this approach, processing all frames have great overheads.

*Decision 2* If we select 2 and try to minimize the $S_i$ the Block Size would have a small value and increase $S_u$. So, based on formula 3, for minimizing data variance we should try to select all Blocks for processing. Based on the low volume of frames in our approach, processing all samples have low overheads.

### Determining Frame Size

The frame size should be determined small enough that intraframe has unique characteristics and behavior. Also, the Frame Size should be big enough to be a good representation of Block. In this paper, we consider 1 KB as the frame size.

### Determining Sample Size

*Sample Size* We use Cochran Sampling technique for our work. Based on this approach if we process limited samples, we achieve acceptable error bound and confidence interval. Based on the Frame Size we can consider each frame as one unique sample.

*Cochran Sampling* For populations that are large, Cochran yields a representative sample for proportions.

**Table 1 Our notations in our algorithm**

| Symbol | Description |
|---|---|
| $BS$ | Block Size |
| $UBS$ | Upper bound of Block Size |
| $LBS$ | Lower bound of Block Size |
| $PT$ | Processing time |
| $PPT$ | Preferred processing time |
| $PT(B_i)$ | Processing time of i-th block |
| $DV$ | Data variety |
| $IDS$ | Input data size |
| $PEB$ | Preferred error bound |
| $EB$ | Error bound |
| $PD$ | Processed data |
| $SBSA$ | Suitable block size for approximation |
| $EVar$ | Expected variance |
| $Var_{srs}$ | Variance of simple random sample |

$$n_0 = \frac{Z^2 pq}{e^2} \tag{4}$$

Which is valid where $n_0$ is the sample size, $Z^2$ is the abscissa of the normal curve that cuts off an area at the tails ($1 - \alpha$ equals the desired confidence level, e.g., 95%), e is the desired level of precision, $p$ is the estimated proportion of an attribute that is present in the population, and $q$ is $1 - p$. The value for $Z$ is found in statistical tables which contain the area under the normal curve.

## Determining Block Size

*Problem definition* As discussed in previous sections data skew reduces the efficiency of approximation. We must design an approach to reduce the impact of skew on approximation efficiency. We must design an approach that can present acceptable performance in any condition of data skew and in case of SQL and NoSQL data sets. For this goal, we present 2 algorithms to solve the problem. Algorithm 1 presents our approach for approximation in case of data skew to minimize the processing time. Algorithm 2 presents our approach for determining the Suitable block size for approximation.

*Notation* We use some notation for the presentation of our algorithm. We show this notation in Table 1.

*Problem statement* Based on Eqs. 1 to 3, error bound and confidence interval depend on inter and intra block variance. For reduction of the error bound the block size must be determined in a way that the amount of sample is reduced.

*Problem formulation* The objective function, to be minimized, is the total processing time of blocks and the constraint is the error bound. *PT*, *EB*, and *PEB* presents Processing Time, Error Bound and Preferred Error Bound. Thus, the problem can be formulated as below:

$$f(x) = Min \sum_{i}^{n} PT(B_i) \tag{5}$$

Subject to:

$$EB < PEB \tag{6}$$

Equation (5) is the objective function which should be minimized. Equation (6) is the constraint of our work. Based on the Spark and RDD limitation these constraints exist in our work. Default RDD size in Spark is 64 MB. We can merge RDDs for generating a bigger RDD.

Algorithm1. Deision1(skew-aware approximation)

```
1: Input: DV, IDS
2: output: BS, PT
3:  select(SBSA for BS)                                   // Algorithm2
4:  estimate $S_u$, $S_i$, PD,PT and EB
5:  while (!min(PT) & PEB < EB)                           // minimize PT and take PEB
6:        Select all blocks                               // Multistage Sampling
7:      Random Sampling for intra /blocks(frames)         //Multistage Sampling
8:     Merge partial results and generate final results   // we used Scala commands
9: end while
```

Lines 1–2 of the algorithm is initializing the variables. Note that *DV*, *IDS*, *BS* and *PT* show the Data variety, Input Data Size, Block Size and Processing Time. Line 3 presents algorithm 2 to select *SBSA*. Line 4 estimate the $S_u$, $S_i$, *PD*, *PT*, and *EB*.

The loop in lines 5 to 9 shows the multistage sampling from which the Preferred Error Bound and minimum Processing Time is achieved.

*Complexity Analysis* The algorithm time complexity is of O (1). Because the algorithm selects all blocks and sample intra them based on the Cochran method.

Figure 5 shows our approach in general (Algorithm 1). Samples derived from primary RDDs form new RDDs and intermediate RDDs join each other and make the final RDD.

*Simple Random Sample* We have used Simple Random Sample to determine the skew condition. Using a cluster sample generally requires either a larger sample size than a SRS or using a wider confidence interval. To determine segment size, we have used the design effect formula [57]. The design effect is used to determine how much larger sample size or confidence interval need to be. Based on Eq. 7 the design effect increases as cluster sizes increases, and as interclass correlation increases.

$$DEFF = 1 + \rho(n - 1) \tag{7}$$

*DEFF* is design effect, $\rho$ is the intraclass correlation, $n$ is the average size of the cluster.

Based on Eq. 8 the estimated variance decreases when the variance of Simple Random Sample decreases and the *DEFF* decreases.

$$\widehat{Var}\left(\hat{\tau}\right) = \widehat{Var_{srs}}\left(\hat{\tau}\right) \times DEFF \tag{8}$$

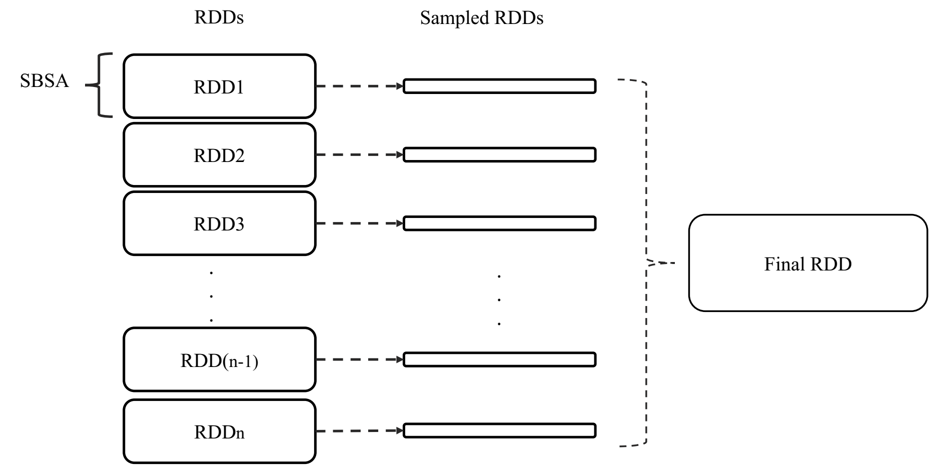

**Fig. 5** Algorithm 1 overview

We present algorithm 2 to determine the *Suitable Block Size for Approximation.* We have used *DEFF* to estimate the amount of data skew. To overcome this issue, we divide the Block Size and increase the amount of samples by this approach.

```
Algorithm2. Determine Suitable Block Size for Approximation
1: Input: DV, IDS
2: output: BS
3:While(!SBSA)
4:   select(All data as BS)
5: Simple Random Sample from data
6:  estimate Var using DEFF
7: if ($Var_{srs}$ > EVar)
8:      Divide  RDDs
9:  Select Suitable Bock Size for approximation
10: end while
```

Lines 1–2 of the algorithm is initializing the variables. Note that *DV*, *IDS*, and *BS* show the Data variety, Input Data Size and Block Size. The loop in Lines 3 to 10 selects *SBSA* as *BS*. Line 4 select all data as Block Size. Line 5 estimates data skew by simple random sampling. In line 6, 7 and 8 if the variance of SRS is bigger than Expected Variance the RDDs will be divided.

*Complexity Analysis* Line 6–9 of the algorithm is a loop procedure and takes O (log2 (n)) time. Where n represents the volume of input data.

Figure 6 shows the overview of algorithm 2. In the case of existing low skew in input data, the block size can be big. In the case of existing negligible skew in data, the input data can be on the block. In this case, our algorithm works like ApproxHadoop.

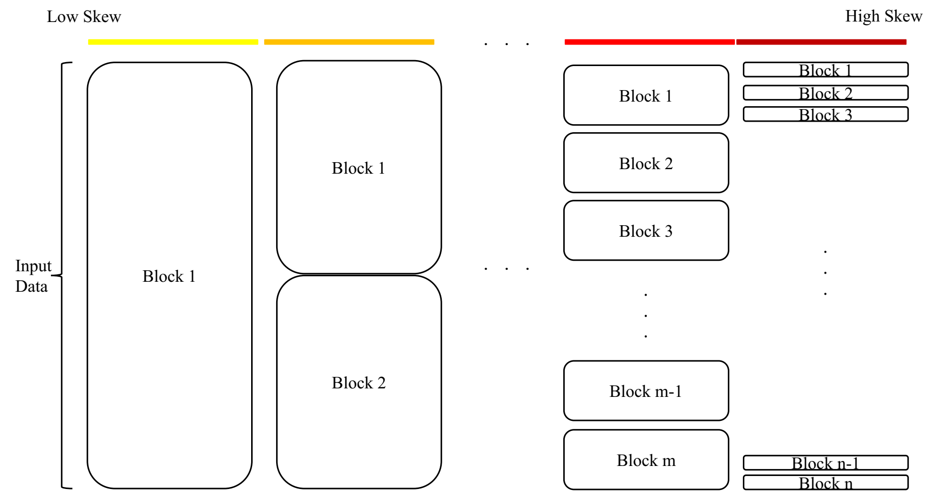

**Fig. 6** Algorithm 2 overview

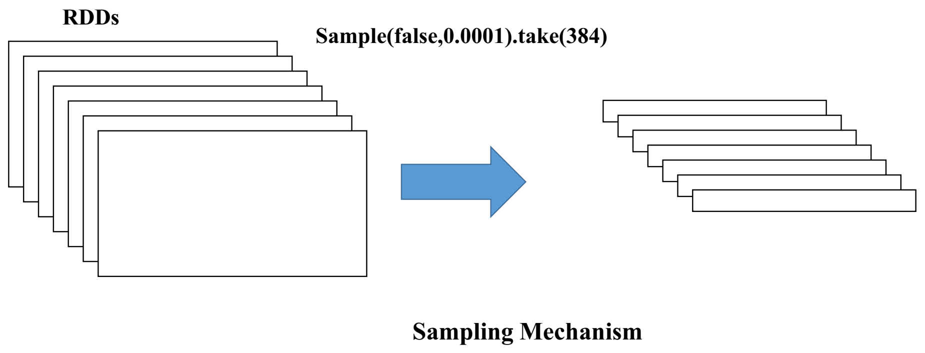

**Fig. 7** Implementation of our approach in Spark

By using these two algorithms, we take Decision 1 when low skew exists and take Decision 2 when there is a high skew in data.

### Implementation in Spark

We have used commands and abilities of Spark and Scala language to implement our approach. We get some samples from RDDs as Fig. 7 shows. For this purpose, we consider frames within RDDs. In our approach, the frame is the smallest unit of our sampling. Based on the Cochran sampling technique, when we get a certain number of frames as a sample, the acceptable confidence interval is achieved. For example in Fig. 7, we get 384 samples with a volume ratio of 0.0001 (per each sample) of all data. This samples make a new RDD and will be processed.

## Results and discussion

We evaluate our approach using applications from a wide area, multiple applications, and datasets.

*Datasets* We evaluated seven benchmarks WordCount, Grep, and AverageLength from BigDataBench suite [58]. The datasets comprised of 100 GB of data from three different sources [53–55]. We also used well-known TPC benchmark [18] and Amazon review for our evaluations [19, 59]. We have used a bootstrapping method for generating 100 GB data as input datasets [52].

*Application*

- WordCount: This application Counts the number of words in the file.
- Grep: It searches and counts a pattern in a file.
- AverageLength: This application calculates the average length of words in the file.
- We also consider AVG for TPC-H datasets and SUM for Amazon datasets.

## Methodology

*Hardware (Servers)* We ran Experiments on 5 machines, Intel Core-i7 4 core CPU at 2.8 GHz with 4 GB of RAM.

*Software* We have used Spark version 2.0 on Ubuntu12.04 as the framework of experiments. For our experiments.

## Experimental results

The following figures presented the impact of Data variety on the volume of data that must be processed to achieve a 95% confidence interval and 5% error bound. In this experiment, we consider 0.5 GB for Block size.

In this section, we used two well-known datasets for our evaluations:

- Amazon datasets (Mc: Music, Bs: Books, Ms: Movies, Cg: Clothing, PS: Phones), and
- TPC-H datasets (M: MAIL, S: SHIP, A: AIR, R: RAIL, T: TRUCK).

Amazon product data contains product reviews and metadata from Amazon, including 142.8 million reviews spanning May 1996–July 2014.

(TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industry-wide relevance. This benchmark illustrates decision support systems that examine large volumes of data, execute queries with a high degree of complexity, and give answers to critical business questions.

We compared our work with other approaches in two factors: the *amount of data processed* and the *processing time*. In Figs. 8, 9, and 10. Vertical axis shows the relative amount of data to be processed and the horizontal axis presents the datasets. As discussed in previous sections, our approach can surpass Approxhadoop and Sapprox. Based on the uneven distribution of real datasets, ApproxHadoop processed a large amount of data to achieve processing requirements. Sapprox cannot perform efficient estimation of the amount of data to be processed. So, our approach can surpass other approaches.

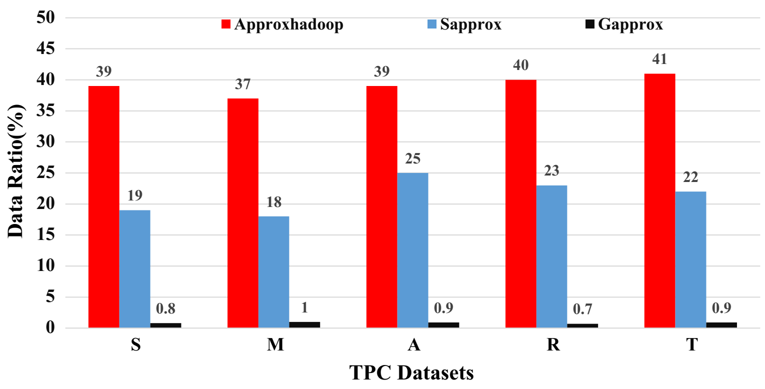

**Fig. 8** Amount of data must be processed to achieve the sampling requirements (TPC datasets)

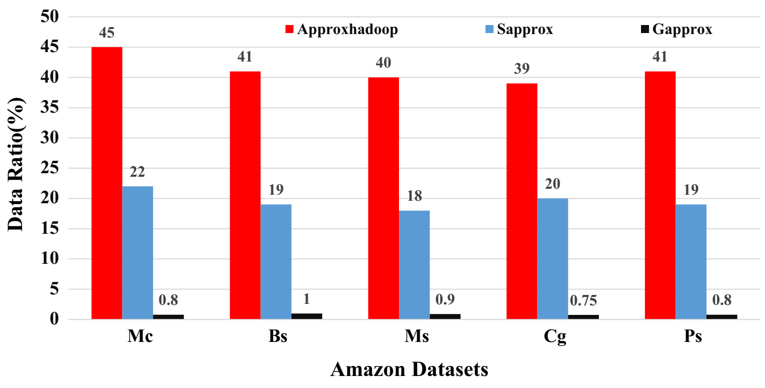

**Fig. 9** Amount of data must be processed to achieve the sampling requirements (Amazon datasets)

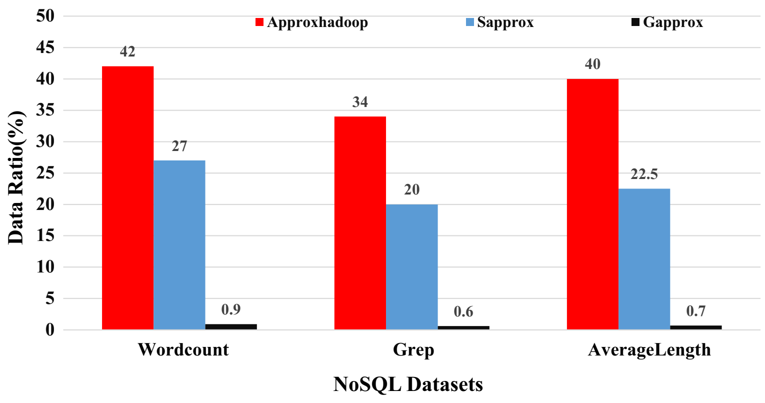

**Fig. 10** Amount of data must be processed to achieve the sampling requirements (NoSQL datasets)

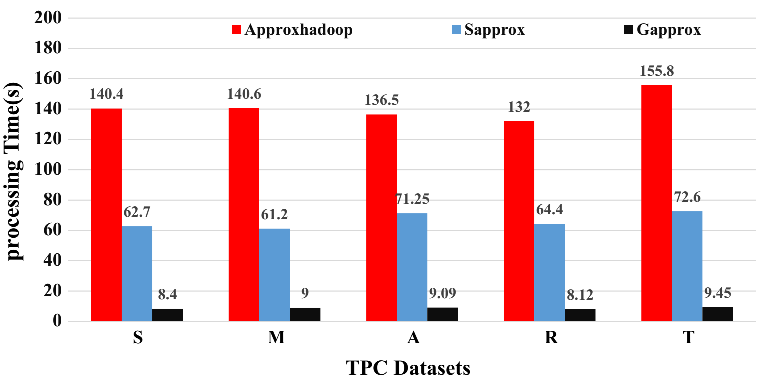

**Fig. 11** Processing time for sampling (TPC datasets)

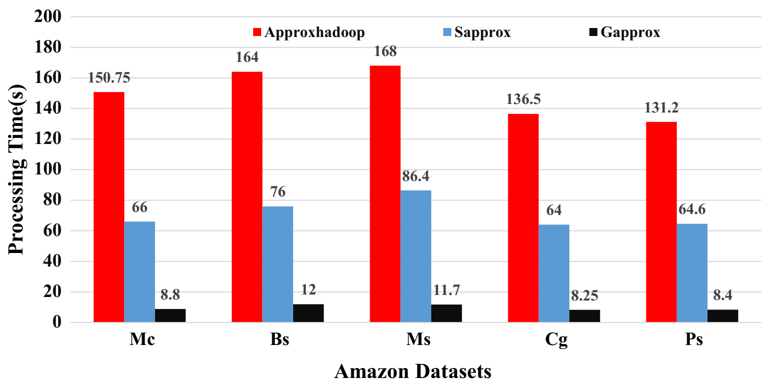

**Fig. 12** Processing time for sampling (Amazon datasets)

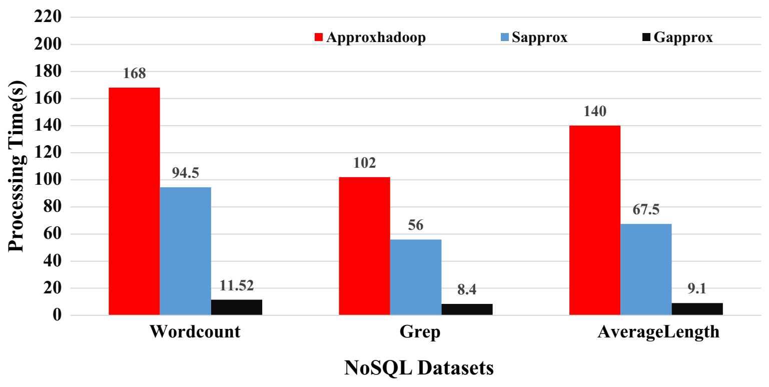

**Fig. 13** Processing time for sampling (NoSQL datasets)

In Figs. 11, 12, and 13, processing time (in seconds) is presented as the vertical axis and the horizontal axis shows the datasets. Our approach can reduce the time of processing as these figures show.

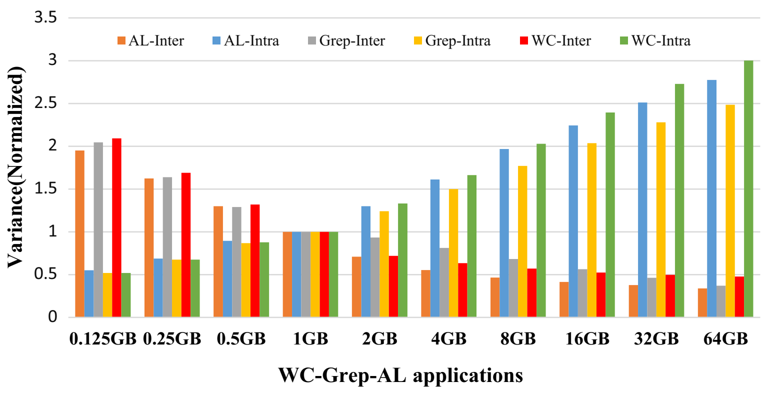

**Fig. 14** Impact of block size on the intra/inter variance (NoSQL datasets)

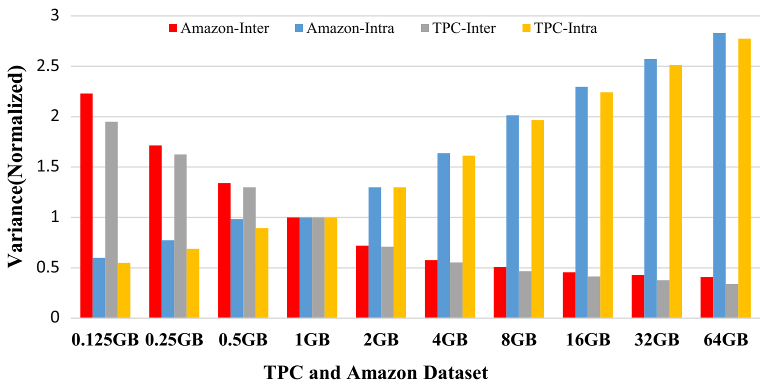

**Fig. 15** Impact of block size on the intra/inter variance (TPC and Amazon datasets)

Our approach improved the processing time up to 17× and 8× compared to ApproxHadoop and Sapprox, Respectively. We also reduce the amount of data processed to achieve the sampling requirement up to 58× and 32× compared to ApproxHadoop and Sapprox. In NoSQL datasets, Sapprox has lower performance and we achieve up to 9× speed up in compared with it.

## Sensitivity analysis

*Impact of Block Size on Inter/Intra Block Variance* Figures 14 and 15 show the impact of BS on Inter/Intra block Variance in SQL and NoSQL datasets. Vertical axis presents the variance (normalized to the 0.5 GB variance) and horizontal axis presents different block size.

1. In the case of increasing the Block Size, the intra Block variance is increasing and the inter-block variance is decreasing.
2. In the case of decreasing the Block Size, the inter Block variance is increasing and the intra-block variance is decreasing.

Based on these observations we decide to select 0.5 GB as Block Size. By this decision, we are able to achieve desirable confidence interval and error bound.

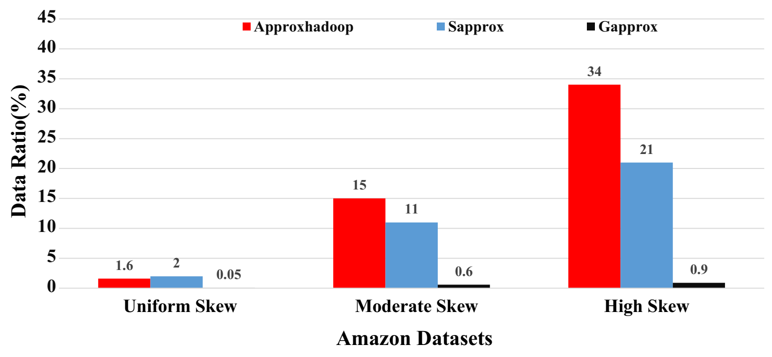

**Fig. 16** Impact of data variety on the sampling overhead (Amazon datasets)

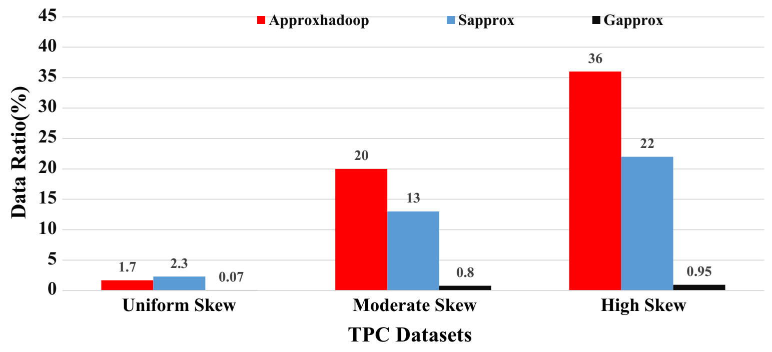

**Fig. 17** Impact of data variety on the sampling overhead (TPC datasets)

*Impact of data skew on volume of sample* We discover the impact of data skew on volume of sampled data. We used a statistical method to model skewed data.

*Modeling data skew* We have used Zipfian [41, 60], distribution to generate skewed data. Zipf's law states that out of a population on $N$ elements, the frequency of elements of rank $k$, f(k; z, N) is:

$$f(k; z, N) = \frac{\frac{1}{k^z}}{\sum_{n=1}^{N}\left(\frac{1}{n^z}\right)}$$

Following the Zipfian distribution, the frequency of occurrence of an element is inversely proportional to its rank.

In the current context, let:

1. $N$ = total number of input partitions;
2. $k$ be their rank; partitions are ranked as per the number of records in the partition that satisfy the given predicate;
3. $z$ be the value of the exponent characterizing the distribution.

We have considered $z=0$ for uniform distribution, $z=1$ for moderate skew and $z=2$ for high skew.

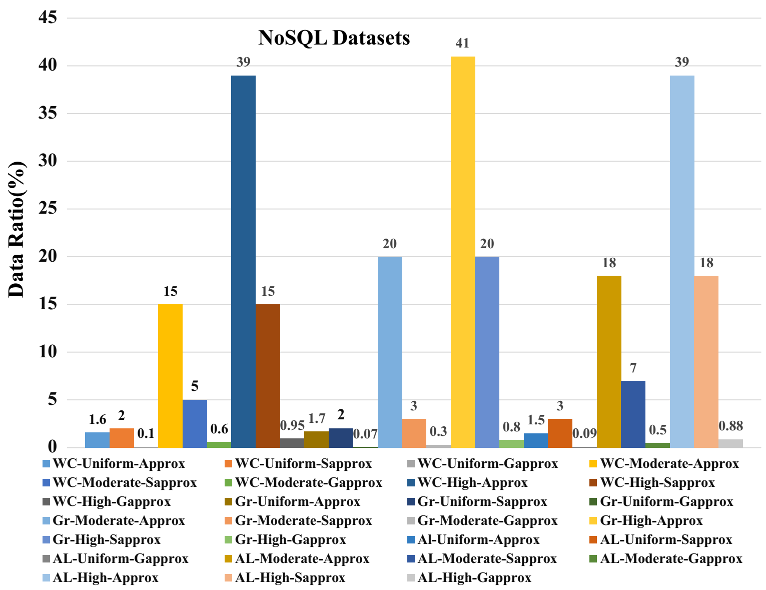

**Fig. 18** Impact of data variety on the sampling overhead (NoSQL datasets)

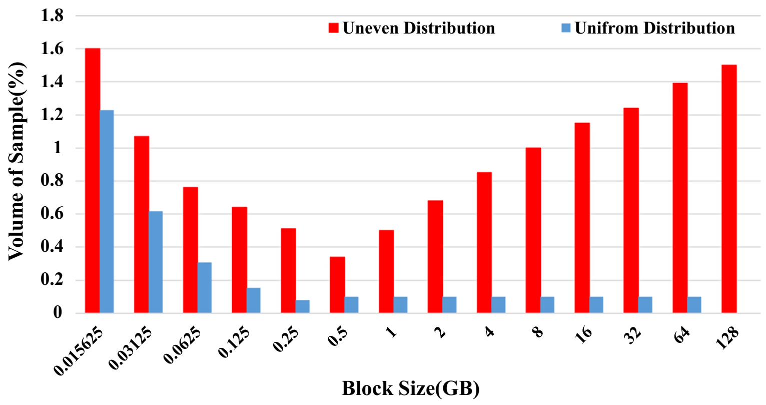

**Fig. 19** Impact of block size on volume of sample

Figures 16, 17 and 18 show the impact of data skew on the amount of data to be processed to achieve processing requirements. The vertical axis shows the relative amount of processed data and the horizontal axis shows the datasets. As above figures present, Sapprox has lower efficiency in case of NoSQL data. The reason for this is that the determination of *Sampling Interval* is complex for NoSQL datasets.

*Impact of Block Size on Volume of Sample* In Fig. 19, we show the effect of block size on the volume of the sample. The vertical axis shows the volume of sample and the horizontal axis shows the block size. Aggregating input data from multiple resources is one of the factors that cause data skew. We have shown it's effect in previous sections. Figure 19, shows that in case of aggregating input data from the single source, the big block is better than a small block and in case of input data from multiple sources, the small block size is better than big block sizes. We have used

**Table 2 Qualitative and quantitative analysis comparisons with other methodologies**

| | Amount of data must be processed | | Amount of data must be processed prediction | Targeted applications |
|---|---|---|---|---|
| | Uniform distribution (%) | Uneven distribution (%) | | |
| Approx. | < 1 | 20–70 | No | Sql/noSql |
| Sapprox | < 1 | 5–30 | No | Sql |
| Gapprox | < 1 | < 1 | Yes | Sql/noSql |

Wikipedia for single source and four different sources [53–55] and Wikipedia as multiple sources. The same behavior is observed for all applications.

## Qualitative and quantitative analysis comparisons

In this section, we present a qualitative and quantitative comparison of approaches. Table 2. Presents the comparisons in summary. More detail descriptions are provided below in Table 2.

*The advantage of Gapprox to ApproxHadoop*

- Determined sized sample
- Considering data variety and skew
- Processing lower amount of input data to achieve acceptable confidence interval and error bound.

*The advantage of Gapprox to Sapprox*

- Processing lower amount of input data to achieve acceptable confidence interval and error bound.
- No need to specify the sampling interval.
- Applicable for SQL and NoSQL applications.
- More simple than Sapprox for implementation

*Our novelties in the presented approach*

- We offer an approach that has acceptable performance in various conditions of data variety/skew.
- Our approach is simple for users to be implemented. The users can easily divide the input data into some same size blocks and apply the Gapprox to them.
- Our approach has low overhead.
- Our approach can be used for SQL and NoSQL simply and presents acceptable efficiency in any type of data.
- We have used abilities in Spark and Scala language to present a practical simple approach. So, the users can easily implement this approach in Spark.

ApproxHadoop have high performance in case of uniform distribution. But in case of skewed data does not have acceptable performance. Sapprox has better performance in case of data variety. Our algorithm also can surpass Sapprox in SQL and NoSQL datasets. Sapprox has another weakness in the case of NoSQL datasets. In this kind of dataset determining *Sample Interval* is complex.

## Conclusions and future work

In this paper, we offer a solution for reducing processing time in case of existing data variety in input data. We consider data variety/skew and use multi-stage sampling to discover the input data. We select fixed block (cluster) and frame size. We use Cochran sampling for intra block sampling.

Based on the experimental results our approach can surpass other approaches, improve processing time and reduce the amount of data to be processed to achieve the desired error bound and confidence interval. Our approach processes a certain amount of input data to achieve processing requirements. For evaluation, we have used Amazon and TPC datasets as SQL datasets and some other as NoSQL datasets. Our approach surpasses two other approaches in SQL and NoSQL datasets. We also have modeled data skew and show that our approach is more efficient than other approaches in various skew conditions. We have shown that in uniform data sets large block size and in skewed datasets, small block size is suitable in case of sampling overheads.

There are some directions for future works. One of them is the key management in intermediate data to reduce data transportation among the network. In aggregation applications, the input and intermediate data are converted to a certain value of a key. By intermediate processing and key management, we are able to reduce data transportation.

**Abbreviations**

Gapprox: using Gallup approach for approximation in Big Data processing; BS: block size; DV: data variety; EB: error bound; IDS: input data size; UBS: upper bound of block size; LBS: lower bound of block size; PEB: preferred error bound; PT: processing time; PPT: preferred processing time; PD: processed data; SABA: suitable block size for approximation; $VAR_{SRS}$: variance of simple random sample.

**Authors' contributions**

HA is the primary researcher for this study. His contributions include the original idea, literature review, implementation and initial drafting of the article. MG guided the initial research concept and played a crucial role in the design of the analytics approaches presented. FF discussed the results with the primary author to aid writing of the evaluation and conclusion sections and played an essential role in editing the paper. All authors read and approved the final manuscript.

**Acknowledgements**

Not applicable.

**Competing interests**

The authors declare that they have no competing interests.

**Availability of data and materials**

BigDataBench: http://prof.ict.ac.cn/.
TPC Benchmark: http://www.tpc.org/information/benchmarks.asp.
Amazon product data: http://jmcauley.ucsd.edu/data/amazon/.
IMDB data files: https://datasets.imdbws.com/.
Gutenberg datasets: https://www.gutenberg.org/.
Quotes-dataset: https://www.kaggle.com/akmittal/quotes-dataset.

**Funding**

Not applicable.

## Publisher's Note

Received: 26 October 2018 Accepted: 14 February 2019
Published online: 26 February 2019

**References**

1. Walton CB, Dale AG, Jenevein RM. A taxonomy and performance model of data skew effects in parallel joins. In: VLDB, vol. 91; 1991.
2. Ananthanarayanan G, Kandula S, Greenberg AG, Stoica I, Lu Y, Saha B, Harris E. Reining in the outliers in map-reduce clusters using Mantri. In: OSDI'10 Proceedings of the 9th USENIX conference on Operating systems design and implementation, Vancouver, BC, Canada; 2010.
3. Dean J, Ghemawat S. MapReduce: simplified data processing on large clusters. Commun ACM. 1958;51(1):107–13.
4. Ananthanarayanan G, Hung MC, Ren X, Stoica I, Wierman A, Yu M. GRASS: trimming stragglers in approximation. In: Proceedings of the USENIX symposium on networked systems design and implementation (NSDI); 2014.
5. Baek W, Chilimbi TM. Green: a framework for supporting energy-conscious programming using controlled approximation. In: Proceedings of the ACM SIGPLAN conference on programming language design and implementation; 2010.
6. Chaudhuri S, Das G, Narasayya V. Optimized stratified sampling for approximate query processing. ACM Trans Database Syst. 2007;32(2):9.
7. Garofalakis MN, Gibbon PB. Approximate query processing: taming the TeraBytes. In: Proceedings of the international conference on very large databases (VLDB); 2001.
8. Sampson A, Dietl W, Fortuna E, Gnanapragasam D, Ceze L, Grossman D. EnerJ: approximate data types for safe and general low-power computation. In: Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI); 2011.
9. Mittal S. A survey of techniques for approximate computing. ACM Comput Surv. 2016;48:62.
10. Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica I. BlinkDB: queries with bounded errors and bounded response times on very large data. In: Proceedings of the European conference on computer systems (EuroSys). 2013.
11. Doucet A, Godsill S, Andrieu C. On sequential Monte Carlo sampling methods for Bayesian filtering. Stat Comput. 2000;10(3):197–208.
12. Liu JW, Shih WK, Lin KJ, Bettati R, Chung JY. Imprecise computations. In: Proceedings of the IEEE. 1994.
13. Misailovic S, Roy DM, Rinard MC. Probabilistically accurate program transformations. In: International Static Analysis Symposium. 2011.
14. Sidiroglou-Douskos S, Misailovic S, Hoffmann H, Rinard M. Managing performance vs. accuracy trade-offs with loop perforation. In: Proceedings of the 19th ACM SIGSOFT symposium and the 13th European conference on Foundations of software engineering. 2011.
15. Goiri I, Bianchini R, Nagarakatte S, Nguyen TD. Approxhadoop: bringing approximations to mapreduce frameworks. ACM SIGARCH Comput Arch News. 2015;43:383–97.
16. Zhang X, Wang J, Yin J. Sapprox: enabling efficient and accurate approximations on sub-datasets with distribution-aware online sampling. Proc VLDB Endowment. 2016;10(3):109–20.
17. Ahmadvand H, Goudarzi M. Using data variety for efficient progressive Big Data processing in warehouse-scale computers. IEEE Comput Arch Lett. 2017;16(2):166–9.
18. TPC. http://www.tpc.org/default.asp. Accessed 30 Sept 2018.
19. Amazon product data. http://jmcauley.ucsd.edu/data/amazon/. Accessed 30 Sept 2018.
20. Condie T, Neil C, Peter A, Joseph MH, Khaled E, Russell S. MapReduce online. In: Nsdi. 2010.
21. St Amant R, Yazdanbakhsh A, Park J, Thwaites B, Esmaeilzadeh H, Hassibi A, Ceze L, Burger D. General-purpose code acceleration with limited-precision analog computation. In: ISCA '14 Proceeding of the 41st annual international symposium on Computer architecture, Minneapolis, Minnesota, USA. 2014.
22. Li K, Li G. Approximate query processing: what is new and where to go? Data Sci Eng. 2018;3:379.
23. Acharya S, Gibbons PB, Poosala V, Ramaswamy S. The Aqua approximate query answering system. In: SIGMOD '99 Proceedings of the 1999 ACM SIGMOD international conference on management of data, Philadelphia, Pennsylvania, USA. 1999.
24. Chaudhuri S, Das G, Narasayya V. A robust, optimization-based approach for approximate answering of aggregate queries. In: SIGMOD '01 proceedings of the 2001 ACM SIGMOD international conference on management of data, Santa Barbara, California, USA. 2001.
25. Babcock B, Chaudhuri S, Das G. Dynamic sample selection for approximate query processing. In: SIGMOD '03 Proceedings of the 2003 ACM SIGMOD international conference on management of data, San Diego, California. 2003.
26. Agarwal S, Milner H, Kleiner A, Talwalkar A, Jordan M, Madden S, Mozafari B, Stoica I. Knowing when you're wrong: building fast and reliable approximate query processing systems. In: SIGMOD '14 Proceedings of the 2014 ACM SIGMOD international conference on management of data, Snowbird, Utah, USA. 2014.
27. Pol A, Jermaine C. Relational confidence bounds are easy with the bootstrap. In: SIGMOD '05 Proceedings of the 2005 ACM SIGMOD international conference on management of data, Baltimore, Maryland. 2005.
28. Zeng K, Gao S, Mozafari B, Zaniolo C. The analytical bootstrap: a new method for fast error estimation in approximate query processing. In: SIGMOD '14 Proceedings of the 2014 ACM SIGMOD international conference on management of data, Snowbird, Utah, USA. 2014.
29. Zeng K, Gao S, Gu J, Mozafari B, Zaniolo C. ABS: a system for scalable approximate queries with accuracy guarantees. In: SIGMOD '14 proceedings of the 2014 ACM SIGMOD international conference on management of data, Snowbird, Utah, USA. 2014.
30. Yan Y, Chen LJ, Zhang Z. Error-bounded sampling for analytics on big sparse data. Proc VLDB Endowment. 2014;7(13):1508–19.

31. Wang L, Christensen R, Li F, Yi K. Spatial online sampling and aggregation. Proc VLDB Endowment. 2015;9(3):84–95.
32. Laptev N, Zeng K, Zaniolo C. Early accurate results for advanced analytics on MapReduce. Proc VLDB Endowment. 2012;5(10):1028–39.
33. Kandula S, Shanbhag A, Vitorovic A, Olma M, R. Grandl, Chaudhuri S, Ding B. Quickr: lazily approximating complex adhoc queries in bigdata clusters. In: Proceedings of the 2016 international conference on management of data. 2016.
34. Yang J, Yecies B. Mining Chinese social media UGC: a big-data framework for analyzing Douban movie reviews. J Big Data. 2016;3(1):3.
35. Chandramouli B, Jonathan G, Abdul Q. Scalable progressive analytics on big data in the cloud. Proc VLDB Endowment. 2013;6:1726–37.
36. Ramnarayan J, Mozafari B, Wale S, Menon S, Kumar N, Bhanawat H, Chakraborty S, Mahajan Y, Mishra R, Bachhav K. SnappyData: a hybrid transactional analytical store built on spark. In: SIGMOD '16 proceedings of the 2016 international conference on management of data, San Francisco, California, USA. 2016.
37. Zeng K, Agarwal S, Dave A, Armbrust M, Stoica I. G-OLA: generalized on-line aggregation for interactive analysis on Big Data. In: SIGMOD '15 proceedings of the 2015 ACM SIGMOD international conference on management of data, Melbourne, Victoria, Australia. 2015.
38. Li F, Wu B, Yi K, Zhao Z. Wander join and XDB: online aggregation via random walks. ACM SIGMOD Record. 2017;46(1):33–40.
39. Zamani AR, AbdelBaky M, Balouek-Thomert D, Rodero I, Parashar M. Supporting data-driven workflows enabled by large scale observatories. In: 2017 IEEE 13th international conference on e-science (e-science), Auckland, New Zealand. 2017.
40. Wang J, Zhang X, Yin J, Wang R, Wu H, Han D. Speed up Big Data analytics by unveiling the storage distribution of sub-datasets. IEEE Trans Big Data. 2018;4(2):231–44.
41. Grover R, Carey MJ. Extending map-reduce for efficient predicate-based sampling. In: 2012 IEEE 28th international conference on data engineering, Washington, DC, USA. 2012.
42. Venkataraman S, Panda A, Ananthanarayanan G, Franklin MJ, Stoica I. The power of choice in data-aware cluster scheduling. In: OSDI. 2014.
43. Kwon Y, Balazinska M, Howe B, Rolia J. A study of skew in mapreduce applications. 2011.
44. Kwon Y, Balazinska M, Howe B, Rolia J. SkewTune: mitigating skew in mapreduce applications. In: SIGMOD '12 proceedings of the 2012 ACM SIGMOD international conference on management of data, Scottsdale, Arizona, USA. 2012.
45. Singh D, Reddy CK. A survey on platforms for big data analytics. J Big Data. 2015;2(1):8.
46. Apache Spark. http://spark.apache.org/. Accessed 30 Sept 2018.
47. Resilient Distributed Dataset. https://jaceklaskowski.gitbooks.io/mastering-apache-spark/spark-rdd.html. Accessed 30 Sept 2018.
48. What is rdd. https://databricks.com/glossary/what-is-rdd. Accessed 30 Sept 2018.
49. Apache Spark-RDD. https://www.tutorialspoint.com/apache_spark/apache_spark_rdd.htm. Accessed 30 Sept 2018.
50. Lohr S. Sampling: design and analysis. Scarborough: Nelson Education; 2009.
51. Cochran WG. Sampling techniques. New York: Wiley; 2007.
52. Efron B, Tibshirani R. Bootstrap methods for standard errors, confidence intervals, and other measures of statistical accuracy. Stat Sci. 1986;1:54.
53. IMDb data files. https://datasets.imdbws.com/. Accessed 30 Sept 2018.
54. Project Gutenberg. http://www.gutenberg.org/. Accessed 30 Sept 2018.
55. Quotes-dataset. https://www.kaggle.com/akmittal/quotes-dataset. Accessed 30 Sept 2018.
56. Multi-Stage Sampling. https://onlinecourses.science.psu.edu/stat506/node/44/. Accessed 30 Sept 2018.
57. Kish L. Survey sampling. New York: Wiley; 1965.
58. Wang L, Zhan J, Luo C, Zhu Y, Yang Q, He Y, Gao W, Jia Z, Shi Y, Zhang S, Zheng C. Bigdatabench: a big data benchmark suite from internet services. In: IEEE 20th international symposium on high performance computer architecture (HPCA). 2014.
59. Recommender Systems Datasets. https://cseweb.ucsd.edu/~jmcauley/datasets.html. Accessed 30 Sept 2018.
60. Knuth DE. The art of computer programming: volume 3: sorting and searching. Boston: Addison-Wesley; 1973.