International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-2, July 2019

An Efficient Pre and Post Processing Skyline Computational Framework using Mapreduce

P.Venkateswara Rao, Mohammed Ali Hussain

Abstract*: Parallel processing will change the world of computation. Now a day's parallel processing will take leading role in all processing that are aided with computer system. Skyline query processing is not exceptional. Skyline processing with MapReduce will take lead role in skyline computation. Still Skyline processing with MapReduce do not take full advantage of parallelism due to processing of including unnecessary data. This paper contains our proposed Map Reduce - Pre and Post Filter based Skyline Computation (MR-PPFS). The aim of this model is to reduce number of candidates before sending dataset to MapReduced skyline computation. This idea reduces MapReduced skyline computation time compare to other algorithms.

Keywords*: Angle Based partitioning, Grid Partitioning, Skyline query and parallel processing, and Threshold filter.

I. INTRODUCTION

Skyline query [1] processing is having inherent capacity of pruning unwanted data from the huge collection of data space.

Definition1. (Skyline). The skyline of S is a set of points say P, $P \prec S$, So that points in P are not dominated by any other points in S.

Skyline sets derived by Skyline queries are useful in multi-criteria decision making system. Processing of skyline queries are known as skyline computation. Now a day's skyline computation is a cost effective process, because of increasing dataset size. Tuples dominance check is needed to consider the tuples in skyline set. Tuples dominancy check in centralized involves all d dimensions; therefore IO-consumption and CPU-intensiveness will be increased. By considering this point and to improve overall efficiency it is necessary to adopt distributed and parallel computing.

Implementing skyline processing with distributed and parallel environments have been proposed in different approaches is mentioned in a survey [2]. Few studies [3, 4] shown how to adopt MapReduce in skyline computation. MapReduce frame work is widely used to process huge Data for maintaining scalability. This is possible with Hadoop [5] environment. Hadoop can support scalable MapReduce system for processing large scale parallel skyline computation. Hadoop having inter node communication and coordination for MapReduce, so this is notable difference with other existing techniques for distributed and parallel computation. Till now the assumptions made by the MapReduce based skyline query algorithms that the data is uniformly distributed and dimensionally independent. But the real time data is not satisfying these conditions. When we consider Hotels example, there is a request for hotels that are near to destination, hi-quality and with low price. In Reality we never get that type of hotels, because always hotels with hi quality and near to destination is costlier. This type of phenomena in statistics is called anti-correlation [6]. In anti-correlated data whose value is increases in one dimension and decreases in another dimension. Now skyline algorithm suffers with low performance for d-dimensional anti-correlated data space, to overcome this more efficient skyline computation algorithms required.

We proposed a frame work named as Map Reduce - Pre and Post Filter based Skyline Computation (MR-PPFS). Our framework aimed to reduce the number of candidates in a dataset prior to the skyline computation. Here we created a model before computing skyline for anti-correlated data. A threshold filter is generated, Based on this filter data remains below the threshold is filtered.

We generate threshold filter line, based on that we filter data and make reduced data space for skyline query processing. In this framework to prune out no dominated points in local skyline. We used angel-based partitioning technique, to derive Global skyline points set from local skyline points set that adopted grid-based data partition. Finally, our experimental result shows it work for wide range of settings.

We organized the rest of this paper in four sections. Related work in Section 2. Then Section 3 covers frame work details. Then Performance analysis with both analytical and experimental results had shown in Section 4. Section 5 concluded with conclusion and feature work.

II. RELATED WORK

Here we will discuss about MapReduce basic concepts and later different popular Mapreduced skyline algorithms that can be comparable to proposed MR-PPFS model.

Revised Manuscript Received on 30 July 2019.
* Correspondence Author
P.Venkateswara rao*, Asst. Professor and Research Scholar, Department Of CSE, koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur, India.
Dr. Mohammed Ali Hussain, Professor , ECSE, koneru Lakshmaiah Education Foundation (Deemed to be University), Vaddeswaram, Guntur , India.

A. MapReduce basic concept

We need a Special programming paradigm to speed up our work on distributed computing Environment, for that we are using MapReduce. This model

contains two basic tasks, they are Map and Reduce. Map takes data from HDFS and split as tuples (key, value) pairs. Reducer takes input from map and shuffles and produce smaller set of tuples.

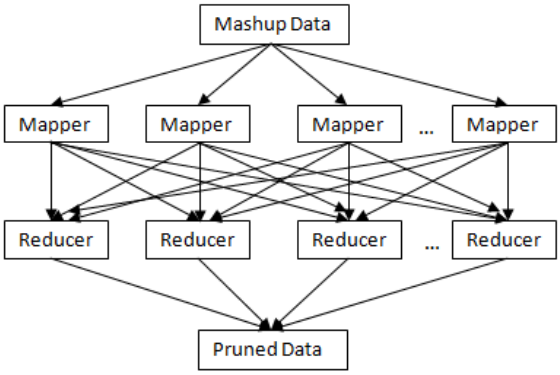


Fig. 1.MapReduce Process

The Map function read each and every record from file and produces a (Key, value) pair. Map will take (key, value) as input and produces List (k2, v2) as output. Reducer will take this list as input and produce (key, value) pair. That is List (key, value). By combining results of all reducers will give final result for MapReduce. The data processed by MapReduce job is stored in distributed file systems. When MapReduce is running then the given input will be splitted and it is distributed on different nodes. This process performed by architecture by itself. Replication of data on different nodes will be automated. In Hadoop to implement MapReduce they use the Distributed cache. Data required for Map and reduce function after MapReduce job starts are moved from Distributed cache to all their nodes.

B. Understanding MapReduce Skyline

Skyline processing in distributed and parallel computing environments has been proposed by many works [7, 8, 9, 10, and 11]. Still it is difficult to implement the above works in MapReduce. Skyline processing in distributed and parallel computing require inter process communication for parallel processing among nodes, But Mapper and Reducer not supports inter process mapper and reducers communication in MapReduce, and it is always communicate by the key-value form. Zang et al. [12] adapted three centralized skyline algorithms with MapReduce. Block nested Loop (BNL) with MapReduce [13] Divides data space into halves, those partitions distributed among Mapper. BNL Skyline computes Local skyline for each portioning. With a single reducer Global skyline is generated by combining all local skyline generated by BNL. Another algorithm called MapReduce based Sort Filter Skyline (MR_SFS) [14] having same processing with presorting technique. Third algorithm MapReduce Bitmap (MR-Bitmap) determines the dominance in skyline computation on each node with bitmap algorithm, and Global skyline is computed using Multiple Reducers. This algorithm can handle with increased data dimensions with less number of distinct values.

MapReduce Angle (MR-Angle) proposed by Chen et al. [16] by adopting the angular-Partitioning [17]. This method divides the data based on angular partitioning. In MapReduced Angel based algorithm data Distributed on Mapper that generate local skyline points and existing single reducer will be used to generate global skyline.

The following points make difference from existing system to proposed MR-PPFS system [3, 4]. In this paper we first used generalized grid partitioning to partition the data space. To represent grid partitioning we use bit string structure that represent empty partitions and dominated partitions. Our design allows early pruning before sending the data to the mapper for local skyline computation. This model generate global skyline using multiple reducers. The following Experimental studies done by comparing MR-PPS with MR-Angle, MR-BNL.

There is another model scheme which computed skyline and reverse skylines on MapReduce, which is SKY-MR which is proposed by Park et al. [18]. To distribute data into partitions, quard-tree based histogram is generated based on that non-skyline points are pruned. In this corresponding reducers computing skyline points from each partition. MapReduce Sky requires additional MapReducer functionality to combine the final results. This causes use of high I/O overhead.

C. MR-GPMRS

The MapReduce Grid Partitioning based Single Reducer (MR-GPSRS) extended with multiple reducer (MR-GPMRS) [19]. This algorithm generate global skyline from local skyline by further dividing and processing them in parallel manner with multiple reducers.

MR-GPMRS [19], this approach adopts grid based partitioning and bit string for skyline computing. This method uses multiple reducers to generate global skyline from local skyline candidates. In this parallel it compare every un-pruned partition P, with the points are located in another un-pruned partition S to prune dominated points [23].

***Definition. 2 (Dominance) if p1 is dominating p2 then it can be represented as p1 ≺ p2. Where point p1∈ S and another point p2 ∈S, where S is a set of points*. In this case p1 is not worse than p2 in all dimensions and p1 is better than p2 for at least in one dimension.**

III. ARCHITECTURE AND WORK FLOW

A. Architecture Description

This architecture shows clear idea about proposed MR-PPFS model. There are three phases in this model. Figure.1 shows first preprocessing phase before skyline processing starts. Next two phases described in figure.2. Phase2 to generate skyline points and Phase 3 filters empty patterns.The primary data filtering before skyline computation is performed in phase 1. Phase 2 performed Skyline query processing with MapReduce platform to quickly derive skyline points.

In this frame work Map phase using angle based partitioning to derive local skyline. By applying grid based cell dominance algorithm global skyline is derived from local skyline.

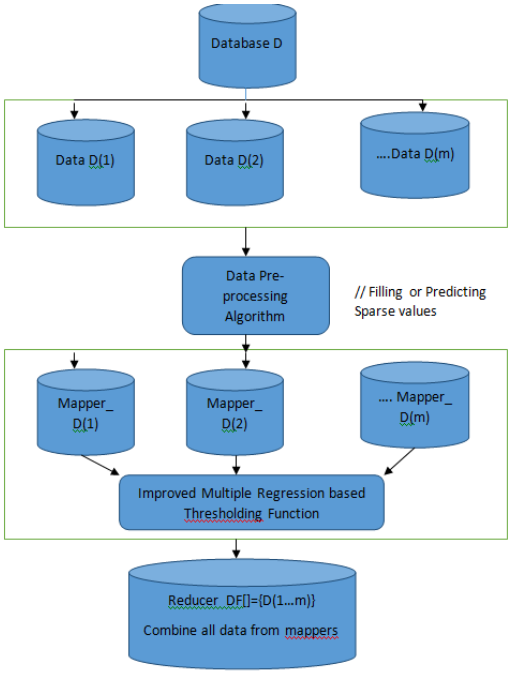


Fig. 2.Map Reduce Framework for Data Pre-processing based on Threshold Computation

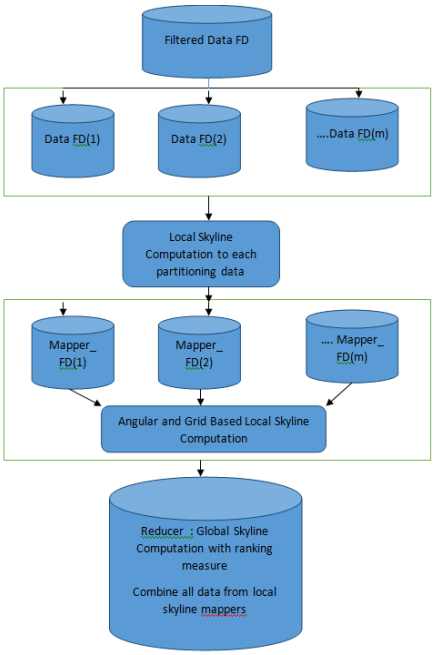


Fig. 3.Map Reduce Framework for local skyline and global skyline computation

B. Work flow

The following description proves proposed model MR-PPFS shown in Fig. 2 and 3where better in processing time compared to MR-Angle, MR-GPMRS, and MR-SKETCH.

1) *Computing Threshold Line for Filtering*

We can reduce unnecessary processing time in skyline by reducing unnecessary data. For this we are using multiple regression analysis.

In this phase before start processing skyline query we try to partitioning the given data set as unpromising and promising points, from that we filter out unpromising data points. This partitioning is done by using Threshold Line, which is built based on multiple regression analysis of the sample data $\hat{D}$= {s1. . . s|D|} in a master node. To model the relationship between a scalar dependent variable y and one or more exploratory variables denoted as x is a linear regression approach. In place of one more than one exploratory variable is used under multiple linear regressions. The Eq.1 shows simple linear regression. In this equation X_i is an explanatory variable and Y_i is a dependent variable, β_0 and β_1 are the regression coefficients. Ordinary Least Squares (OLS) technique [20] is used to estimates regression coefficient β.

$$Yi = \beta 0 + \beta 1 Xi`$$

$$\therefore \beta_i = \frac{\sum_{i=1}^{n} XiYi - X\sum_{i=1}^{n} Yi}{\sum_{i=0}^{n} Xi^2 - X\sum_{i=1}^{n} Xi}, \quad \beta_0 = \bar{Y} - \widehat{\beta 1}\,\bar{X} \qquad (1)$$

We are using more than two dimensions for skyline query processing therefore we used Multiple Linear Regression. Estimation of regression line, and regression coefficients, is shown in Eq. 2.

Here, X indicates a matrix, X^{-1} is an inverse matrix, and $X^{!}$ is a Transpose matrix of X.

$$Y = X^{\circ} B$$

$$Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}, \quad X = \begin{bmatrix} 1 & X_{21} & X_{31} & \cdots & X_{k1} \\ 1 & X_{22} & X_{31} & \cdots & X_{k2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & X_{2n} & X_{3n} & \cdots & X_{kn} \end{bmatrix}, \quad B = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_k \end{bmatrix}$$

$$\therefore B = (X'X)^{-1} X'Y \qquad (2)$$

The conditional means of independent variables are represented by multiple linear regression of dependent variable. We consider points in lower side of regression line because they are not dominated by any other points on top of the regression line. Based on this point Linear regression line consider as a filtering threshold. The above threshold line data used to generate skyline. We compute n-number of filter threshold lines for n dimension data set. Finally combining of this non filtered dataset is used for skyline computation.

Algorithm 1: Initial Data Filtering

Input: Data set D
Output: Candidate cell list

Step 1: Read sample data (D)
Step 2: Find the most dominating cells List
Step 3: calculate multiple regression line
Step 4: filtering with multiple regression line
Step 5: If a grid cell is not in most dominating cell
Step 6: discard the cell
Step 7: else consider the cell as candidate list

2) Skyline Computation Phase

This skyline computation phase derives skyline points. To derive this skyline points we are using hybrid partitioning technique. Hybrid partitioning uses angle based and grid based data partitioning for local and global skyline computation. For validity of this hybrid-partition method we refer [3]. The angular partition the data space into hyper spherical space and then divides into N sectors.

In two dimensional space, left corner of the data space towards origin is skyline, and the points towards upper right corner are non-skyline. This principle is one reason for adopted in angle and grid based portioning in this process. Grid based partitioning divides the available area of dataset by n x n disjoints.

In this sample case assume two dimensions are used (i.e. distance and price). Then further it is divided into four parts. In grid based partitioning dominance relationship among partitions. Before discussing this method we will discuss angle based partitioning and grid based partitioning independently later processing of skyline with hybrid partition.

Angle-Based Data Partition:

This partition helps to reduces redundant computation and balance the workload in MapReduce. This process contains two steps: 1. generating hyper spherical space from Cartesian coordinates.2. Using angular coordinates divides the data space into N sectors.

When we consider n dimensional data space, then the Cartesian coordinates of a service is represented by a vector S = {v_1, v_2… v_n}, here v_i refers to the value of S in the i^{th} attribute dimension. The hyper spherical coordinate of the service S is represented by *(n-1)* angular coordinates {$Ø_1$, $Ø_2$… $Øn_{-1}$}, and *r* radial coordinates as follows:

$$r = \sqrt{v_n^2 + v_{n-1}^2 + \ldots + v_1^2}$$

$$tan(ø_1) = \frac{\sqrt{v_n^2 + v_{n-1}^2 + \ldots + v_2^2}}{v_1}$$

$$\ldots$$

$$tan(ø_{n-2}) = \frac{\sqrt{v_n^2 + v_{n-1}^2}}{v_{n-2}}$$

$$tan(ø_{n-1}) = \frac{\sqrt{v_n^2}}{v_{n-1}} \qquad (3)$$

With the angular coordinates $Ø_i$ the data space is divided into N sectors. A point S = (x, y) in a two dimensional data space is expressed as

Grid Partition Dominance:

First Wang, et al [21] introduced MapReduced based grid partitioning algorithm (MR-Grid) and later extended by Wu, et al [22].

Definition 3 (Grid- partition dominance) A Grid g_i dominates another Grid- partition g_j, denoted by $g_i \prec g_j$ iff $g_{i.max}$ dominates g_j.min. The following Lemma1 allow us to prove the correctness based on the transitivity property [1].

Lemma 1. Given two partitions g_i, g_j, if $g_i \prec g_j$, then all tuples in g_j is dominated by any tuple in g_i, i.e., $\forall t_i \in g_i$ and $\forall t j \in g_j$ we have $g_i \prec g_j$.

The Skyline processing algorithm 1 is given below, which is processed by a MapReduce job. Each Map phase computes local skyline points after dividing the data space by using angle-based partitioning. In the Reduced phase using grid partitioning we prune local skyline points using dominance relationship. After pruning local skyline in the frame work we get global skyline points. After merging the remaining local skyline points we get final skyline points S.

Algorithm2. MapReduce Skyline query processing algorithm

Map Phase

Input: Candidate cell list CD
Output: Local skyline points LSL
Step 1: Init local skyline points LSL
Step 2: For each data d_i in CD
Step 3: AID = Anglebasedparttion (d_i)
Step 4: GID =: Gridbasedpartition (d_i)
Step 5: create (T[m], (d_i, AID, GID)
Step 6: for each data d_i from T[m] by AID
Step 7: insert points (d_i, LSL)
Step 8: output (LSL)

Reduce Phase

Input: The local skyline LSL from all mapper
Step 9: merge LSL by GID
Step 10: for each d_i in GID
Step 11: Insert points (d_i, LSL)
Step 12: Init global skyline GSL
Step 13: for each cell C_i of GID
Step 14: GSL U CmpPrt ({C_i, d_i}, LSL)
Step 15: output (GSL)

Insert points

Input: A point t and skyline S
Output: skyline S
Step1: Init Status=true
Step 2: for each point t` in S
Step 3: if $t^1 \prec t$ then
Step 4: status = false
Step 5: break
Step 5: if $t \prec t^1$ then
Step 6: remove t^1 from S
Step 7: if status =true then
Step 8: add to S
Step 9: return S

CmpPrt (P, Skyline) //Compare partitions

Input: LSL points S_p for partition P
Output: global skyline in P
Step 1: for each S_{pi} in S
Step 2: if $S_p \prec S_{pi}$ then
Step 3: remove all points that are dominated by point in S_{pi}
Step 4: return S_p

3) Final Optimization

Some time scalability of global skyline is high, there is a mechanism required to reduce scalability of global skyline.

In global skyline still there is an existing of sparsity, if we can remove sparsity then it is helps to fasten the query processing. The final optimization phase works on global skyline to produce final skyline.

In this mechanism we adopted k-nearest neighbor algorithm to recognize the points that are most near to our requirement. In this process we pruned the points that are having null coordinates, other than major coordinates[24]. This is an iterative process to identify the final skyline points. This task will reduce scalability of global skyline.

IV. EXPERIMENTAL SETUP AND ANALYSIS

A. Experimental setup

This frame work executed and tested on 2.4 GHz Intel core i5 7th generation system with 8GB RAM (minimum). Java and Spark third party libraries are used to implement the MR-PPFS frame work. For parallel processing we have taken help of Amazon AWS Cloud Server, cloud instance is Ubuntu operating System, cloud data storage is Amazon S3 .This experiment is conducted on synthetic data contains required number of dimensions.

Here we compare the proposed MR-PPFS algorithms with existing algorithms MR-GPMRS [19], MR-Angle [16], and MR-SKETCH.. All these algorithms are implemented in Scala and java.

B. Result Analysis

The following experiments performed by considering two types of data sets they are Independent data and anti-correlated data. The sample parameters used in this experiment are Data cardinality in the range of 10^5,10^6, 10^7. Different dimensions are considered in this experiment is 2, 4, 6, and the sampling rates are 0.1%, 0.5%, 1%. Next the number of nodes considered here are 3, 4, 5.

We have made the following comparisons to show the progress in our proposed MR-PPFS model, they are

1). Processing time (in seconds) of the considered methods with change in different dimensionality,

2). Processing time of the selected methods versus change in cardinality of data set.

Table-I: Effect of Skyline filtering on dataset With cardinality 10^5

Algorithm	Size of Input data	Size of Local Skyline	Size of Global Skyline
MR- Angular	1,00,000	1,00,000	11934
MR- GPMRS	1,00,000	1,00,000	10183
MR-SKETCH	1,00,000	1,00,000	9842
Proposed MR-PPFS frame work	1,00,000	59,785	4519

From this table we can understand that proposed frame work filtering method reduced global skyline set. This status indicates some improvement in existing method compare to previous methods. The following Figure 4 shows processing time of algorithms, with respect to change in cardinality. This graph is built by considering Independent and Anti-correlated data. Clear change in processing time on independent data when cardinality is 10^6, this change is increased when cardinality went to 10^7. Our proposed MR-PPFS algorithm shows 40 seconds time compare to MR-Angle and MR-GPMRS that is 80 and 70 seconds respectively it is clearly indicates that our proposed frame work shows better performance than other algorithm. Same result will be shown in Anti-correlated data.

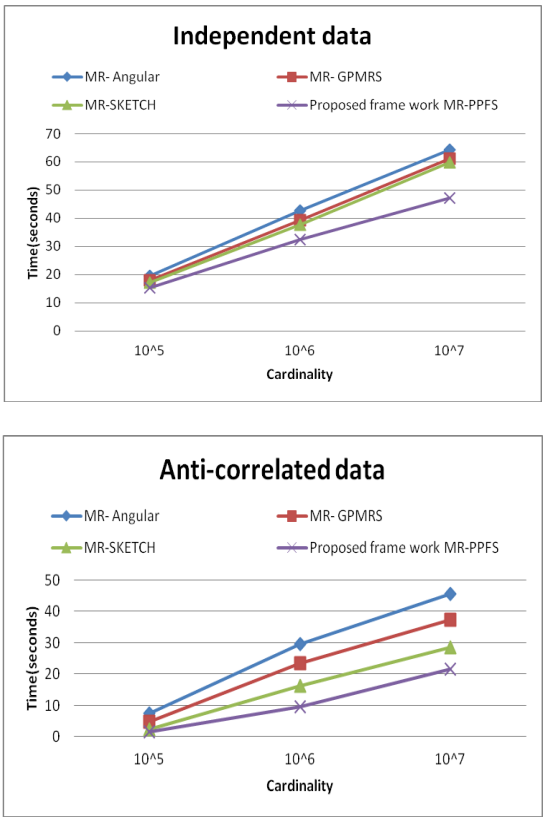
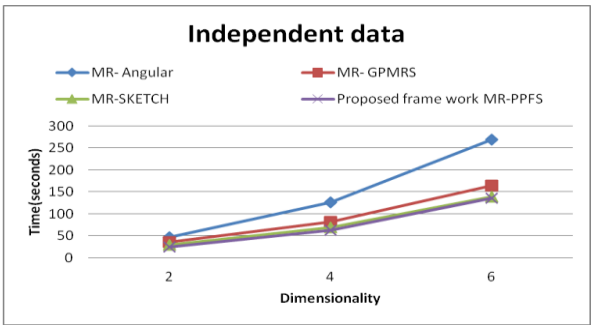


Fig. 4. Processing Time (In Seconds) Of The Considered Methods With Change In Cardinality And Considering Dimension 5 -Independent And Anti-Correlated Data

The following figure shows clear difference between proposed MR-PPFS frame work and existing popular MR-Angular, MR-SKETCH and MR-GPMRS. In Anti-correlated data when we observed processing time of MR-Angular and Proposed MR-PPFS frame work showing similar processing time. But MR-GPMRS taking high processing time like 4000 seconds, this is clearly showing that when increasing dimensionality badly effects MR-GPMRS. Only our proposed MR-PPFS shows less time compared to MR-SKETCH.



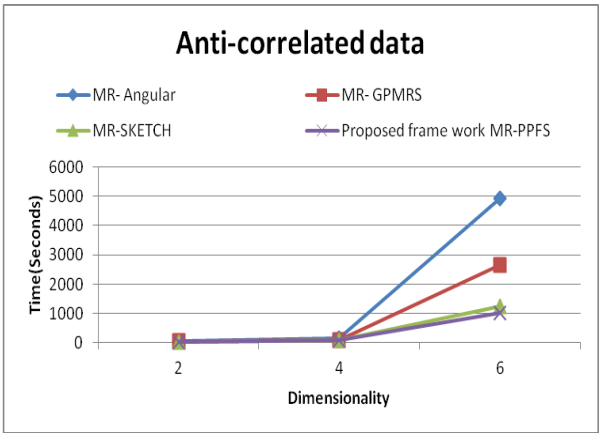


Fig.5 Processing Time (Seconds) Of The Considered Methods With Change In Different Dimensionality

From Above Fig.5 comparisons it is clearly concluded that our proposed MR-PPFS frame work consumes less time compared to MR-Angle and MR-GPMRS algorithms irrespective of Independent and Anti-correlated Data, because using filters reduces overall processing time.

V. CONCLUSION

This paper proposed MR-PPFS a Frame work that includes skyline query processing and data filter to optimize web services in large scale cloud mashup applications. Skyline processing become time consumed when growing data space. For this we proposed pre and post data filtering method that reduces skyline data space, and leads to reducing skyline processing time. To generate Local skyline we used Angular partitioning before data pruning. Next stage before generating global skyline points we partitioned data based on grid portioning. Initial threshold line filtering, filters the basic points that are not relevant to our query, thus minimize skyline computation. Lack of available Machines, we are not able to test our frame work by considering more number of nodes versus increasing cardinality and dimensions. We can also extend our framework to support different Skyline queries.

REFERENCES

1. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator In Proc. of ICDE, 2001.
2. K. Hose and A.Vlachou. A survey of skyline processing in highly distributed environments. VLDB J., 21(3):359-384, 2012.
3. L. Chen, K. Hwang, and J. Wu. MapReduce skyline query processing with a new angular partitioning approach. In IPDPS Workshops & PhD Forum, pp. 2262-2270, 2012.
4. B. Zhang, S. Zhou, and J. Guan. Adapting skyline computation to the MapReduce framework: Algorithms and experiments. In DASFAA Workshop, 2011.
5. Apache. Welcome to Apache ™ Hadoop ®! http://hadoop.apache.org/.
6. Zhang, B., Zhou, S., Guan, J.: Adapting skyline computation to the MapReduce framework: algorithms and experiments. In: DASFAA Workshops (2011).
7. W.T. Balke, U.Guntzer, and J. X. Zheng. Efficient distributed sky lining for web information systems. In Proc. of EDBT, pages 256–273, 2004.
8. K. Hose, C. Lemke, and K.-U. Sattler. Processing relaxed skylines in PDMS using distributed data summaries. In Proc. of CIKM, pages 425–434, 2006.
9. J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Norvag. AGIDS: A grid-based strategy for distributed skyline query processing. In Proc. of GLOBE, 2009.
10. A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis. SKYPEER: Efficient subspace skyline computation over distributed data. In Proc. of ICDE, pages 416–425, 2007.
11. A. Vlachou, C. Doulkeridis, Y. Kotidis, and M. Vazirgiannis. Efficient routing of subspace skyline queries over highly distributed data Trans. on Knowledge and Data Engineering (TKDE), 22(12):1694–1708, 2010.
12. B. Zhang, S. Zhou, and J. Guan. Adapting skyline computation to the MapReduce framework: Algorithms and experiments. In DASFAA Workshops, 2011.
13. S. Borzsonyi, D. Kossmann, and K. Stocker. The skyline operator. In ICDE, pp. 421-430, 2001.
14. J. Chomicki, P. Godfrey, J. Gryz, and D. Liang. Skyline with presorting. In ICDE, pp. 717-719, 2003.
15. K.-L. Tan, P.-K. Eng, and B. C. Ooi. Efficient progressive skyline computation. In VLDB, pp. 301-310, 2001.
16. L. Chen, K. Hwang, and J. Wu. MapReduce skyline query processing with a new angular partitioning approach. In IPDPS Workshops & PhD Forum, pp. 2262-2270, 2012.
17. A. Vlachou, C. Doulkeridis, and Y. Kotidis. Angle-based space partitioning for efficient parallel skyline computation. In SIGMOD, pp. 227-238, 2008.
18. Y. Park, J.-K. Min, and K. Shim. Parallel computation of skyline and reverse skyline queries using MapReduce. PVLDB, 6(14):2002-2013, 2013.
19. K. Mullesgaard, J. L. Pederseny, H. Lu, and Y. Zhou, "Efficient skyline computation in mapreduce ", in EDBT, 2014.
20. Dismuke, C., Richard, L.: Chapter 9: Ordinary least squares. In: Methods and Designs for Outcomes Research, pp. 93–104. American Society of Health-System Pharmacists (2006).
21. S. Wang, B. C. Ooi, A. K. H. Tung, and L. Xu, "Efficient skyline query processing on peer-to-peer networks," International Conference on Data Engineering (ICDE), pp. 1126–1135, 2007.
22. K. Deng, X. Zhou, and H. T. Shen, "Multi-source skyline query processing in road networks", International Conference on Data Engineering, pp. 796–805, 2007.
23. Y. Park, J. Min and K. Shim, "Efficient Processing of Skyline Queries Using MapReduce," in *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 1031-1044, 1 May 2017.
24. Venkateswara rao, P., & Mohammed Ali Hussain, D. (2018). A Novel Filtered Based Grid partitioning multiple reducers skyline computation using Hadoop framework. International Journal of Engineering & Technology,7(2.7),686-690.doi:http://dx.doi.org/10.14419/ijet.v7i2.7.10923