International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:12, No:10, 2018

Real-Time Data Stream Partitioning over a Sliding Window in Real-Time Spatial Big Data

Sana Hamdi, Emna Bouazizi, Sami Faiz

***Abstract*—**In recent years, real-time spatial applications, like location-aware services and traffic monitoring, have become more and more important. Such applications result dynamic environments where data as well as queries are continuously moving. As a result, there is a tremendous amount of real-time spatial data generated every day. The growth of the data volume seems to outspeed the advance of our computing infrastructure. For instance, in real-time spatial Big Data, users expect to receive the results of each query within a short time period without holding in account the load of the system. But with a huge amount of real-time spatial data generated, the system performance degrades rapidly especially in overload situations. To solve this problem, we propose the use of data partitioning as an optimization technique. Traditional horizontal and vertical partitioning can increase the performance of the system and simplify data management. But they remain insufficient for real-time spatial Big data; they can't deal with real-time and stream queries efficiently. Thus, in this paper, we propose a novel data partitioning approach for real-time spatial Big data named VPA-RTSBD (Vertical Partitioning Approach for Real-Time Spatial Big data). This contribution is an implementation of the Matching algorithm for traditional vertical partitioning. We find, firstly, the optimal attribute sequence by the use of Matching algorithm. Then, we propose a new cost model used for database partitioning, for keeping the data amount of each partition more balanced limit and for providing a parallel execution guarantees for the most frequent queries. VPA-RTSBD aims to obtain a real-time partitioning scheme and deals with stream data. It improves the performance of query execution by maximizing the degree of parallel execution. This affects QoS (Quality Of Service) improvement in real-time spatial Big Data especially with a huge volume of stream data. The performance of our contribution is evaluated via simulation experiments. The results show that the proposed algorithm is both efficient and scalable, and that it outperforms comparable algorithms.

***Keywords*—**Real-Time Spatial Big Data, Quality Of Service, Vertical partitioning, Horizontal partitioning, Matching algorithm, Hamming distance, Stream query.

I. Introduction

THe demand of real-time spatial data has been increasing recently. Nowadays, we are talking about a real-time spatial Big Data that process a large amount of heterogeneous data (may be in the size of terabyte). As a result, the real-time spatial Big Data can be overloaded and many transactions may miss their deadlines because data retrieval processes are time consuming. In order to speed up query processing, several works have proposed many optimization techniques as data partitioning. Therefore, breaking a large table into several smaller units is a necessity.

Sana Hamdi is with the Tunisia Polytechnic School, BP 2078 La Marsa,University of Carthage, Tunisia (e-mail: hamdisana@gmail.com).

Emna Bouazizi is with the MIRACL Laboratory, BP. 1088, Sfax 3018 Sfax University, Tunisia (e-mail: emna.bouazizi@gmail.com).

Sami Faiz is with the LTSIRS Laboratory, BP 37 Le Belvedere 1002, Tunis, Tunisia (e-mail: sami.faiz@insat.rnu.tn).

Data partitioning [24] is a fragmentation of a logical database into distinct independent units. It is applied in large-scale databases to improve responsiveness, scalability and availability of data. Several works have showed the importance of this approach. But traditional partitioning approaches are not a real time process. Thus, in real-time spatial Big Data, the traditional partitioning technologies have encountered many problems as:

- Traditional partitioning technologies are based on known table structure. They don't have the ability to partition for unknown database in real-time spatial Big Data;
- Traditional partitioning technology can only deal with persistent and stable workload. But the real-time spatial Big Data can be overloaded and many transactions may miss their deadlines, or real-time spatial data can be violated.
- Traditional partitioning technologies are unable to adapt to high-throughput in real-time spatial Big Data.

In this paper, we research on the limitations of traditional partitioning technologies. Then, we propose a novel approach to process stream queries in real-time spatial Big Data. This contribution is an implementation of the matching algorithm for traditional vertical partitioning. It uses Hamming distance to produce clusters.

The remainder of this paper is organized as follows: In Section II, we introduce some related works. In the Section III, we introduce the our contribution. The simulation model and the results of simulation experiments are given in Section IV. The last section consists of conclusion and some future research directions.

II. Related Works

In this section, we give an overview of real-time spatial Big Data and we discuss pertinent works related to data partitioning approaches.

A. System Overview

Real-time spatial applications have a great importance. Such applications continuously receive a huge amount of heterogeneous data from mobile objects (e.g., moving vehicles in road networks). The streaming nature of real-time spatial data poses new challenges that require combining real-time spatial Big Data and data stream management systems.

In this section, we give an overview of heterogeneous real-time spatial data model and transaction model.

1) Heterogeneous Real-Time Spatial Data Model: Stored data in real-time spatial applications are from heterogeneous sources and are maintained under heterogeneous formats and structures. These data can be divided into two types: the structured data and unstructured data:

- Structured data: can be processed automatically by machines.
- Unstructured data: no common pattern can be used to process for this type of data which come from different sources and have a different format as text, pictures, multimedia content or numeric traces, etc.

Real-time spatial data must be integrated. Structured data and unstructured content are simultaneously accessed via an integrated user interface. The issue of real-time and heterogeneity is extremely important for taking effective decision. As a solution we propose the use of ETL (Extract-Transform-Load) process as follows:

- Data extraction: extracts data from heterogeneous data sources.
- Data transformation: transforms the data for storing it in the proper format or structure for the purposes of querying and analysis.
- Data loading: loads it into the final target (data warehouse).

A real-time spatial data stream distinguishes itself from a traditional real-time data stream in the following: real-time spatial data have the ability to change their locations continuously. Thus, the arrival of a new location information about the data, say p, at some time t_2 ($t_2 > t_1$) may result in expiring the previous location information of p at time t_1. This is in contrast to traditional data where data are expired only after its deadline as it becomes in the system [11].

2) Transaction Model: Spatial real-time transactions can be classified into two classes: update transactions and user transactions.

- Update transactions: update the values of real-time spatial data in order to reflect the state of the real world.
- User transactions (continuous queries): user requests arrive aperiodically and may read real-time data and non-real-time data. This type of transaction can be executed several times or continuously during a period as required by the user.

B. Data Partitioning Approaches

Several surveys on data partitioning algorithm classify them into horizontal and vertical data partitioning methods:

- Horizontal partitioning [17], [2], [20], [15] divides a table into disjoint sets of rows. There are three techniques of horizontal partitioning based on values of data sets (Round-Robin partition, Range partition and Hash partition). Range partitioning is the most popular approach specially when there is a periodic loading of a new data.
- Vertical partitioning [5], [21], [22], [3], [8], [27] divides a table into vertical and disjoint sets of columns. There are two major classes of vertical partitioning:
  - cost-based approach [16], [26], [12], [23]: During this approach, a cost model is constructed to predict the performance of the system for any given configuration. Then, an algorithm enumerating the configuration space is used.
  - procedural approach [28], [22], [9]: During this approach, there is not a cost model. Procedural approach proposes some kind of a procedure which will result in a good configuration.

Both of these strategies (horizontal partitioning and vertical partitioning) have a significant impact the performance of the database systems specially with respect to responsiveness, storage and processing cost. But, they still static (they are not able to adapt to dynamic environments) i.e. a configuration is selected once. In case of changes in the workload (new transaction) or the data (new data) the algorithm has to be re-run. Our goal is to adapt the partitioning scheme to a constantly changing workload in real-time spatial Big Data.

Curino et al., in [2], proposed a workload-driven approach named Schism for database partitioning. Schism creates a graph and uses a method called METIS [6] to divide this graph into K balance parts. Schism has a significant impact the performance of the database systems. But it can't deal with the large volume of stream data and with large-scale dynamic queries.

To solve the problem associated with dynamic data partitioning, Miguel Liroz-Gistau et al. in [13], have proposed a dynamic workload-based partitioning algorithm for continuously growing databases (like databases used in scientific applications where the data is continuously growing to the database). This algorithm defines a mathematical model of dynamic partitioning. This definition is designed with heuristic that considers the affinity of data with queries and fragments. In fact, this approach is quite interesting because the execution time of this algorithm depends only on newly arrived data and not on entire size of the database. But, it is not able to get real-time result after every query.

In this paper [1], Alekh Jindal et al. have presented an efficient O^2P (One-dimensional Online Partitioning) algorithm. The main idea of this algorithm is computing the affinity between every pair of attributes and clustering them [7], [21], [4], [5]. Then, it uses a greedy strategy to calculate the cost of every possible split line to get the best partitioning scheme. Actually, the importance of this approach appears clear. But, it must know the table structure in advance which is not available in real-time spatial Big Data. Besides, it cant deal with stream queries and cant get real-time result after every query.

In this paper [10], Mengyu Guo et al. present a workload-driven stream partitioning system named WSPS to solve the above problems by the integration of partitioning technology and streaming framework. WSPS constructs a dynamic data model, cluster and merge nodes according to the node affinity, then get the optimal partitioning scheme according to a cost model. WSPS can deal with stream data and obtain real-time partitioning scheme. But, it uses distributed queries; a query accessed attributes on different partitions and on several nodes. This costs more resources and

the transactions risk to miss their deadlines while waiting for its validation.

III. A Data Partitioning Approach for Real-Time Spatial Big Data

In this section we describe our contribution. We propose a novel data partitioning approach for real-time spatial Big data; the implementation of the Matching algorithm [14] for vertical partitioning. This algorithm uses Hamming distance to produce clusters.

This approach is divided into three steps that are detailed in the following sections:

- Data model initialization
- Implementation of Matching algorithm
- Data Partitioning.

A. Data Model Initialization

Given a query workload W_t which is a stream of queries seen till time t W_t=$\{q_0, q_1, q_2, .., q_t\}$.

Step 1 : Assuming that the query q accesses the attribute a, we begin by the definition of the access function as follow:

$$Access(q, a) = \left\{ \begin{array}{ll} 1 & q\, access\, a \\ 0 & otherwise \end{array} \right\} \quad (1)$$

Then, we define a matrix M. Rows in the matrix are the attributes accessed by query q (0<i<t) in the workload W_t and columns are the queries. Each element in the matrix $M[i, j]$= Access(q_i, a_j) where $i \in [1, t]$,$j \in [1, m]$ and m is the number of attributes accessed by t queries.

Let us consider an example. Suppose that we have five queries accessing six attributes:

q1: SELECT a FROM T WHERE a = 10;
q2: SELECT b, f FROM T WHERE b = f;
q3: SELECT c, d FROM T WHERE a $\geq$ c;
q4: SELECT f FROM T WHERE f $\leq$ 100;
q5: SELECT e FROM T;

In this case,W_t=$\{q_1, q_2, q_3, q_4, q_5\}$ and

M=

	a	b	c	d	e	f
$q1$	1	0	0	0	0	0
$q2$	0	1	0	0	0	1
$q3$	0	0	1	1	0	0
$q4$	0	0	0	0	0	1
$q5$	0	0	0	0	1	0

When the sliding window continues, some existing transactions are deleted from the sliding window and some new transactions arrive. Thus, M is dynamically updated at every window. If a new query accesses to attributes already exist in M, only a new row will be added on the end. If the query accesses to new attributes not exist in M, a new row will be added on the end and new columns will be added to the matrix on the right. If an existing query is deleted from the sliding window, the row of this query and the attributes acceded only by this query have to be deleted.

B. Implementation of Matching Algorithm

This algorithm is developed to reorganize data and to identify clusters [14]. We start with mentioning the different steps of the Matching algorithm:

Step 1: From an m x t matrix array M compute the m x m array B=$M^T * M$

Step 2: Select one of the m rows of $M^T * M$ arbitrarily; set i= 1.

Step 3: Select j = i + 1.

Step 4: Try placing the j^{th} row in each of the (i + 1) positions. Compute the sum $\phi = \sum_{i=1}^{m-1} b_{i,i+1}$.

Step 5: j = j + 1 and repeat Step 4 until j = m.

Step 6: Place the row k in the position where maximum value of ϕ is obtained , i + 1 $\leq$ k $\leq$ m,

Step 7: i = i + 1 and repeat steps 3,4, 5, 6 and 7 till i=m

We use the same matrix M in our previous example and we apply the different steps of the Matching algorithm as follow:

B =

	$q1$	$q2$	$q3$	$q4$	$q5$
a	1	0	0	0	0
b	0	1	0	0	0
c	0	0	1	0	0
d	0	0	1	0	0
e	0	0	0	0	1
f	0	1	0	1	0

	a	b	c	d	e	f
$q1$	1	0	0	0	0	0
$q2$	0	1	0	0	0	1
$q3$	0	0	1	1	0	0
$q4$	0	0	0	0	0	1
$q5$	0	0	0	0	1	0

=

	a	b	c	d	e	f
a	1	0	0	0	0	0
b	0	1	0	0	0	1
c	0	0	1	1	0	0
d	0	0	1	1	0	0
e	0	0	0	0	1	0
f	0	1	0	0	0	2

Initially, $\phi = \sum_{i=1}^{5} b_{i,i+1}$ =1

The final reordering given through the application of the algorithm is:

B =

	d	c	f	b	a	e
d	1	1	0	0	0	0
c	1	1	0	0	0	0
f	0	0	2	1	0	0
b	0	0	1	1	0	0
a	0	0	0	0	1	0
e	0	0	0	0	0	1

$\phi = \sum_{i=1}^{5} b_{i,i+1}$ =2

The optimal attribute sequence Oas=$\{d, c, f, b, a, e\}$. Every time a new query comes, the matrix M is calculated, then the new OaS is dynamically created.

C. Data Partitioning

The main objective of vertical partitioning approach in real-time spatial Big Data is to improve the performance of query execution and the system throughput. The high performance of query execution is related to minimizing the

access cost of data partitions. Especially that the frequency of accessing data on different partitions is a major factor to affect the query execution cost. Thus, it is very important to minimize this frequency for the high performance of query execution.

The improvement of the system throughput can be achieved by maximizing the degree of parallel execution. We can improve this degree if we can minimize the frequency of interfered accesses between data queries.

As a result, we can define the cost model that reflects both objectives of vertical partitioning mentioned above as follow:

$$\text{Cost}(q_i, P(W_t, Oas_t)) = |\sum_{L(q_i)\subseteq L'}(\alpha C(q_i) + I(q_i)) \times |L'| - \sum_{L(q_i)\subseteq L-L'}(\alpha C(q_i) + I(q_i)) \times |L - L'|| \quad (2)$$

where:

- $P(W_t, Oas_t)$ is a partitioning scheme over OaS of workload W on the time t.
- $L(q_i)$ is a collection of attributes the query q visited.
- A partition line split the OaS into two sets L' and L-L'.
- $C(q_i)$ is the access number of q_i.
- $I(q_i)$ is the interfered access number of q_i.
- α is a proportional constant between $C(q_i)$ and $I(q_i)$, $\alpha > 1$.

Our objective is to find the split vector SV that minimize the execution cost, which is defined as follows:

$$SV = arg\,min(Cost(q_i, P(W_t, Oas_t)) \quad (3)$$

D. Algorithm Analysis

The characteristics of $VPA - RTSBD$:

- it deals with stream data; there is no need to have all queries before partitioning.
- it improves the performance of query execution.
- it improves the system throughput by maximizing the degree of parallel execution.
- it can get real-time result after every query: a real-time partitioning scheme.

We compare the following properties: best time complexity, worst time complexity, real-time processing, workload type, table structure of $VPA - RTSBD$ with $WSPS$, $Schism$ and O^2P as shown in Table I.

IV. Simulation Results

In this section, we give our simulation model. Then, we compare the result of VPA-RTSBD and the result of the traditional partitioning approaches like $WSPS$,$Schism$ and O^2P.

Although VPA-RTSBD is the best in its comparison with $WSPS$, $Schism$ and O^2P, the split vector calculation becomes time-consuming especially when the number of partitions grows.

TABLE I
Algorithm Comparison

	VPA-RTSBD	WSPS	Schism	O^2P
Real-time processing	Yes	Yes	No	No
Stream processing	Yes	Yes	No	No
Workload type	Dynamic/ Static	Dynamic/ Static	Static	Static
Table structure	Unknown/ know	Unknown/ known	Known	Known
Best time complexity	O(n)	O(n)	-	O(n)
Worst time complexity	O(n)	O(n)	-	$O(n^2)$
Optimize queries processing	Yes	Yes	-	-
Optimize system throughput	Yes	No	-	-

A. Simulation Model

In order to access the performance of real-time analytics on big data, new several systems have appeared. Well-known systems and prototypes include: Hadoop Online, Storm, Flume, Kafka and S4. But, these systems lacks the most important database properties ACID (Atomicity, Consistency, Isolation, Durability) and data warehousing without the ACID requirement in place within a given system, reliability is suspect. Databases with ACID properties are guaranteed to achieve successful database transactions. Meanwhile, we focus on interactive analytic in a data warehouse, rather than continuous analytic over streams. Thus, we have implemented a simulator in Java which describes the architecture FCSA-RTSBD (Feedback Control Scheduling Architecture for Real-Time Spatial Big Data) [18] as shown in Fig. 1.

In our system, a transaction T_i is associated with a deadline D_i, period P_i, start time R_i, end time E_i and Execution Time Estimation ETE_i. Update transactions arrive periodically and the arrival of user transactions is defined using the Poisson distribution given by the following formula:

$$F_x(t) = \left\{ \begin{array}{cc} e^{-t} & x > 0 \\ 0 & otherwise \end{array} \right\} \quad (4)$$

T_i is continually evaluated for stream data belonging to a window whose size is defined by either the period P_i or number of the data received most recently.

Real-time spatial transactions have scheduled transactions, according to the Earliest Deadline First (EDF) algorithm. Transaction handler consists of a concurrency controller (CC) by the use of the algorithm SCC-2S-P-IC [19], a freshness manager (FM) and a basic scheduler. A transaction can be aborted and restarted by CC. Freshness manager (FM) checks the freshness of real-time data before the initiation of a user transaction. If the accessing data is currently stale, FM blocks the corresponding transaction will be transferred from the block queue to the ready queue as soon as the corresponding data is put up to date.

Simulation results are measured by the monitor periodically. Miss ratio Controllers and precision control compute the miss ratio and utilization control signals based on the obtained results.

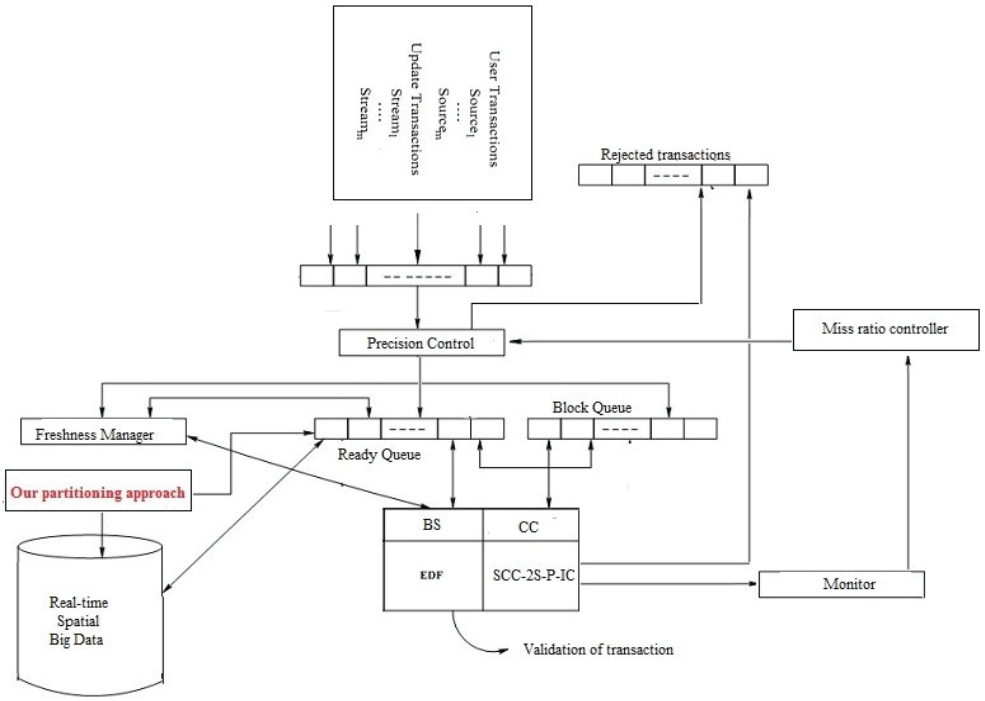

Fig. 1 Simulation model

B. Experiments

1) Experiment 1: Partition Time Evaluating: We compare the partition time of $VPA-RTSBD$ with $WSPS$, O^2P and $Schism$. We use 5 workloads of size 10M (6000 queries), 100M (60,000 queries), 500M (3 million queries), 1G(6 million queries) and 2G (12 million queries) of TPC-DS [25]; a decision support benchmarka for comparing big data processing systems, containing 25 tables, 429 columns and 99 query templates.

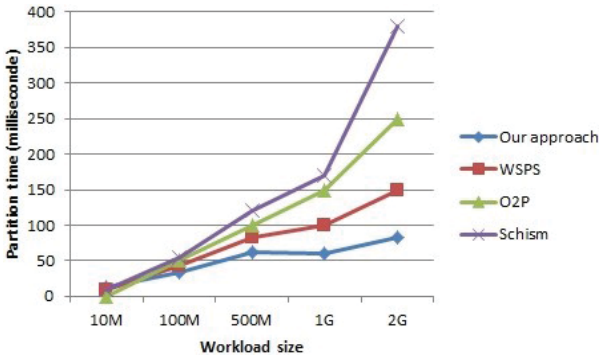

Fig. 2 Partition Time of TPC-DS

By analyzing the result in Fig. 2, we can find, firstly, that when the workload size is increasing, the partition time is increasing also for all algorithms. In other hand, although $VPA-RTSBD$ keeps a query window which means partitioning is done after every N queries contrarily $WSPS$ partitioning is done after every query, $VPA-RTSBD$ and $WSPS$ have the same computing complexity and the partition time of our approach is significantly lower than $WSPS$.

Schism and O^2P can't deal with the large volume of stream data and with large-scale dynamic queries. So, they have the worst partition time.

2) Experiment 2: High-Throughput Adaption: We use a workload size of 500M and we generate data at different rates (from 0.5G/s to 5G/s). The objective of this experiment is to evaluate the ability of the high-throughput adaption. The result is as shown in Fig. 3.

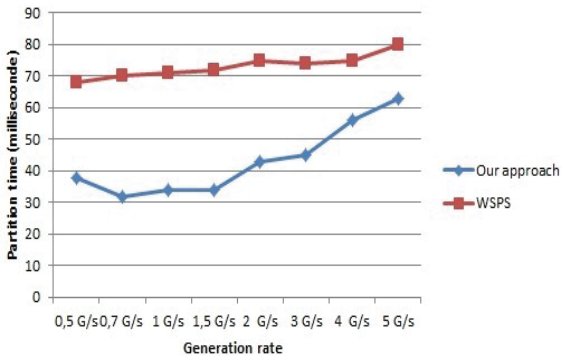

Fig. 3 High-throughput Control

By analyzing the result, we can find that the rate of generating data affect the partition time for both algorithms $WSPS$ and $VPA-RTSBD$. But our approach has the ability to adapt to high-throughput better than $WSPS$. So, the importance of $VPA-RTSBD$ appears clear; it can deal well when facing with large-scale stream queries.

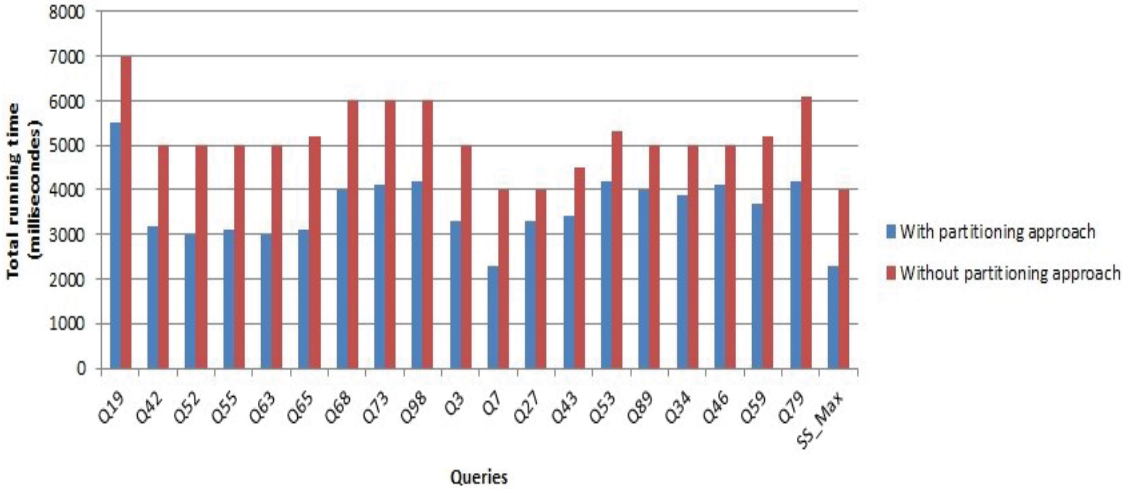

Fig. 4 Evaluation on all queries using 1TB data

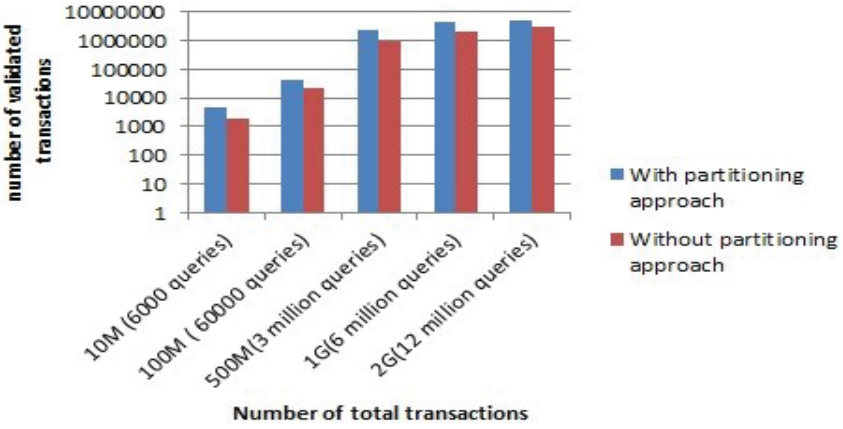

Fig. 5 Success ratio evaluating

3) Experiment 3: Total Running Time Evaluating: Fig. 4 presents the total running time of our simulator on all 20 queries of the benchmark TPC-DS with a dataset size fixed to 1 TBytes. On all queries, FCSA-RTSBD with partitioning approach outperforms FCSA-RTSBD with partitioning approach for all types of queries. The importance of our partitioning approach appears clear because partitioning algorithm improves responsiveness, scalability and availability of data.

4) Experiment 4: Success Ratio Evaluating: Fig. 5 shows that If we increase the number of accepted transactions in the system, the number of validated transactions is increasing also. Moreover, the number of valid transactions (user and update) using our partitioning approach is the best. This is explained by the fact that our approach maximizes the degree of parallel execution. Thus this policy allows a large number of transactions to complete their execution before achieving their deadlines.

V. Conclusion

In this paper, we have researched on the limitations of traditional partitioning technologies. Then, we have proposed $VPA - RTSBD$ a novel approach to process stream queries in real-time spatial Big Data. This contribution is an implementation of the Matching algorithm for traditional vertical partitioning. It uses Hamming distance to produce clusters. $VPA-RTSBD$ is divided into three steps : first, we find automatically the optimal attribute sequence by the use of Matching algorithm. Secondly, we keep the data amount of each partition more balanced limit by the use of a cost model. Finally, we provide a parallel execution guarantees for the most frequent queries.

A simulation study is shown to prove that $VPA-RTSBD$ can achieve a significant performance improvement in terms of success ratio, high-throughput adaption and total running time compared to $WSPS$, O^2P and $Schism$. The importance of our partitioning approach appears clear because partitioning algorithm improves responsiveness, scalability and availability of data. This affects QoS improvement in real-time spatial Big Data especially with a huge number of data and transactions.

As follow, we have to find more policies for QoS improvement in a large-scale real-time spatial data. The most important requirements for these data structures are the ability of providing fast access to the large volumes of data. Thus, we

shall find new techniques for the data indexing. Another future work consists of relaxing transaction real-time constraints (ACID) by allowing the loss of some invocations.

REFERENCES

[1] A. Jindal J. Dittrich, *Relax and let the database do the partitioning online.* In International Workshop on Business Intelligence for the Real-Time Enterprise (pp. 65-80). Springer, Berlin, Heidelberg, 2011, September.

[2] C. Curino, E. Jones, Y. Zhang S. Madden, *Schism: a workload-driven approach to database replication and partitioning.* Proceedings of the VLDB Endowment, 3(1-2), 48-57, 2010.

[3] D. A. Shraddha Phansalkar, *Transaction aware vertical partitioning of database (TAVPD) for responsive OLTP applications in cloud data stores.* Journal of Theoretical and Applied Information Technology, 59(1), 2014.

[4] D. W. Comer S. Y. Philip, *A vertical partitioning algorithm for relational databases*. In Data Engineering, 1987 IEEE Third International Conference on (pp. 30-35). IEEE, 1987, February.

[5] D. W. Cornell P. S. Yu, *An effective approach to vertical partitioning for physical design of relational databases.* IEEE Transactions on Software engineering, 16(2), 248-258, 1990.

[6] G. Karypis V. Kumar, *METIS–unstructured graph partitioning and sparse matrix ordering system*, version 2.0., 1995.

[7] J. A. Hoffer D. G. Severance, *The use of cluster analysis in physical data base design.* In Proceedings of the 1st International Conference on Very Large Data Bases (pp. 69-86). ACM, 1975, September.

[8] J. H. Son M. H. Kim, *An adaptable vertical partitioning method in distributed systems.* Journal of Systems and Software, 73(3), 551-561, 2004.

[9] L. Rodrguez X. Li, *A support-based vertical partitioning method for database design.* In Electrical Engineering Computing Science and Automatic Control (CCE), 2011 8th International Conference on (pp. 1-6). IEEE, 2011, October.

[10] M. Guo H. Kang, *The Implementation of Database Partitioning Based on Streaming Framework.* In Web Information Systems and Applications Conference, 2016 13th(pp. 157-162). IEEE, 2016, September.

[11] M.F. Mokbel, X. Xiong, W.G. Aref, S.E. Hambrusch, S. Prabhakar M.A. Hammad, *PALACE: a query processor for handling real-time spatio-temporal data streams*, In Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, August, pp.1377-1380, 2004.

[12] M. Hammer B. Niamir, *A heuristic approach to attribute partitioning.* In Proceedings of the 1979 ACM SIGMOD international conference on Management of data (pp. 93-101). ACM, 1979, May.

[13] M. Liroz-Gistau, R. Akbarinia, E. Pacitti, F. Porton P. Valduriez, *Dynamic workload-based partitioning for large-scale databases.* In International Conference on Database and Expert Systems Applications (pp. 183-190). Springer, Berlin, Heidelberg, 2012, September.

[14] M. V. Bhat A. Haupt, *An efficient clustering algorithm.* IEEE Transactions on Systems, Man, and Cybernetics, (1), 61-64, 1976.

[15] P. A. Bernstein, I. Cseri, N. Dani, N. Ellis, A. Kalhan, G. Kakivaya, ... T. Talius, *Adapting microsoft SQL server for cloud computing.* In Data Engineering (ICDE), 2011 IEEE 27th International Conference on (pp. 1255-1263). IEEE, 2011, April.

[16] S. Agrawal, V. Narasayya B. Yang, *Integrating vertical and horizontal partitioning into automated physical database design.* In Proceedings of the 2004 ACM SIGMOD international conference on Management of data (pp. 359-370). ACM, 2004, June.

[17] S. Ahirrao R. Ingle, *Scalable transactions in cloud data stores.* Journal of Cloud Computing: Advances and Applications, SpringerOpen, 4:1-14, 2015.

[18] S. Hamdi, E. Bouazizi S. Faiz, *A new qos management approach in real-time gis with heterogeneous real-time geospatial data using a feedback control scheduling.* In Proceedings of the 19th International Database Engineering Applications Symposium, pages 174179. ACM, 2015.

[19] S. Hamdi, E. Bouazizi S. Faiz, *A Speculative Concurrency Control in Real-Time Spatial Big Data Using Real-Time Nested Spatial Transactions and Imprecise Computation.* In Computer Systems and Applications (AICCSA), 2017 IEEE/ACS 14th International Conference on (pp. 534-540). IEEE, 2017, October.

[20] S. Das, A. El Abbadi D. Agrawal, *ElasTraS: An Elastic Transactional Data Store in the Cloud.* HotCloud, 9, 131-142, 2009.

[21] S. Navathe, S. Ceri, G. Wiederhold J. Dou, *Vertical partitioning algorithms for database design.* ACM Transactions on Database Systems (TODS), 9(4), 680-710, 1984.

[22] S. Navathe M. Ra, *Vertical partitioning for database design: a graphical algorithm.* In ACM Sigmod Record(Vol. 18, No. 2, pp. 440-450). ACM, 1989, June.

[23] S. Papadomanolakis A. Ailamaki, *An integer linear programming approach to database design.* In Data Engineering Workshop, 2007 IEEE 23rd International Conference on (pp. 442-449). IEEE, 2007, April.

[24] S. Phansalkar S. Ahirrao, *Survey of data partitioning algorithms for big data stores.* In Parallel, Distributed and Grid Computing (PDGC), 2016 Fourth International Conference on (pp. 163-168). IEEE, 2016, December.

[25] TPC-DS: Transaction Performance Processing Council for Decision Support-decision Support [online] http://www.tpc.org/tpcds/ [accessed, April 30, 2014].

[26] W. W. Chu I. T. Ieong, *A transaction-based approach to vertical partitioning for relational database systems.* IEEE Transactions on Software Engineering, 19(8), 804-812, 1993.

[27] W. Zhao, Y. Cheng F. Rusu, *Workload-Driven Vertical Partitioning for Effective Query Processing over Raw Data.*, 2015.

[28] X. Lin, M. Orlowska Y. Zhang, *A graph based cluster approach for vertical partitioning in database design.* Data Knowledge Engineering, 11(2), 151-169, 1993.