Hindawi Publishing Corporation
Discrete Dynamics in Nature and Society
Volume 2013, Article ID 209672, 10 pages
http://dx.doi.org/10.1155/2013/209672

Research Article

Efficient Processing of Continuous *Skyline* Query over Smarter Traffic Data Stream for Cloud Computing

Wang Hanning,[1] Xu Weixiang,[2] Jiulin Yang,[3] Lili Wei,[4] and Jia Chaolong[1]

[1] *State Key Laboratory of Rail Traffic Control and Safety, Beijing JiaoTong University, Beijing 100044, China*
[2] *School of Traffic and Transportation, Beijing JiaoTong University, Beijing 100044, China*
[3] *China National Tendering Center of Mach. & Elec. Equipment, Beijing 100142, China*
[4] *Chongqing Public Security Bureau, Chongqing 401147, China*

Correspondence should be addressed to Xu Weixiang; wxxu@bjtu.edu.cn

Received 19 September 2013; Accepted 4 November 2013

Academic Editor: Huimin Niu

The analyzing and processing of multisource real-time transportation data stream lay a foundation for the smart transportation's sensibility, interconnection, integration, and real-time decision making. Strong computing ability and valid mass data management mode provided by the cloud computing, is feasible for handling *Skyline* continuous query in the mass distributed uncertain transportation data stream. In this paper, we gave architecture of layered smart transportation about data processing, and we formalized the description about continuous query over smart transportation data *Skyline*. Besides, we proposed *mMR-SUDS* algorithm (*Skyline* query algorithm of uncertain transportation stream data based on *micro-batchinMap Reduce*) based on sliding window division and architecture.

1. Introduction

Recently, tremendous changes have taken place in city transportation data sources, transportation data services, and information infrastructure. Traditional *ITS (intelligent transport systems)* present many defects in higher-dimensional space-time continuous data stream collected and passed back from mass perceptible and measurable sensor networks and the storage, processing, and analysis of big data. With the advent of computing technology such as Internet of things, cloud computing [1], and smarter transportation [2] has emerged, as a new concept of comprehensive transportation system. As shown in Figure 1, smarter transportation system covers various aspects of transportation and is a complex and comprehensive system consisting of plenty of subsystems. Analytical processing of multi-source and real-time transportation data stream [3] is the basis of realizing perceptible Smarter Transportation with interconnection integration and real-time decision. Besides, such analytical processing is critical to establishing global sustainable transportation surveillance, network optimization of dynamic transportation, automatic response to accidents, and integration of location-based transportation services.

With the rapid development of information technology, monitoring platform in various types of transportation information management collects complex mass transportation stream data including video information [4, 5] from cameras, monitoring information of sensors, positioning system information of vehicle, and so on. Hence, transportation stream data are provided with diverse sources, wide varieties, different forms, and typical data-intensive processing characteristics. For example, by December 28, 2012, there were 8842 fixed transportation monitoring equipment in Beijing and merely dispatch center for transportation operational monitoring *TOCC* in Beijing updated over 3500 data immediately and replaced more than 20 thousand video pictures in real time. Operational applications of environmental sensor station are shown in Figure 2. Real-time transportation data stream lays important data foundation for road transportation stream control of various decision analysis and emergency response in smarter transportation system. *Skyline* [6] query, as a key data mining technology,

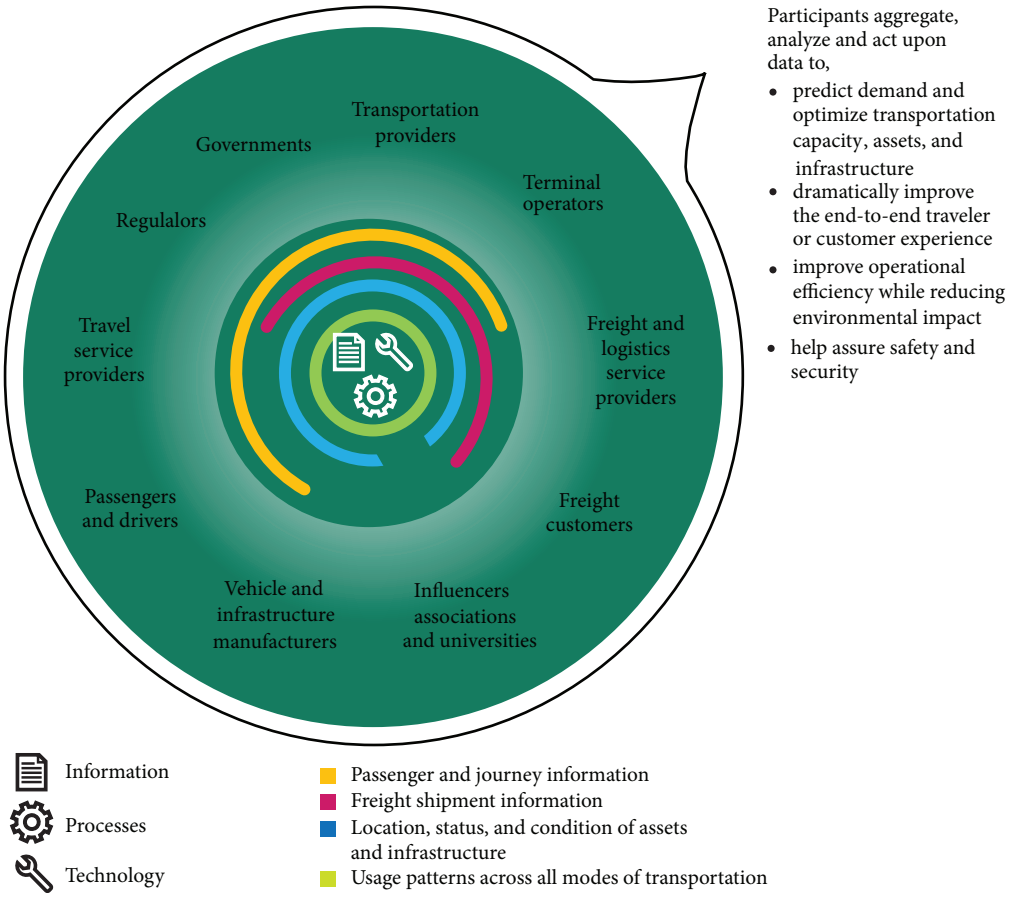

Figure 1: Smarter Transportation Information.

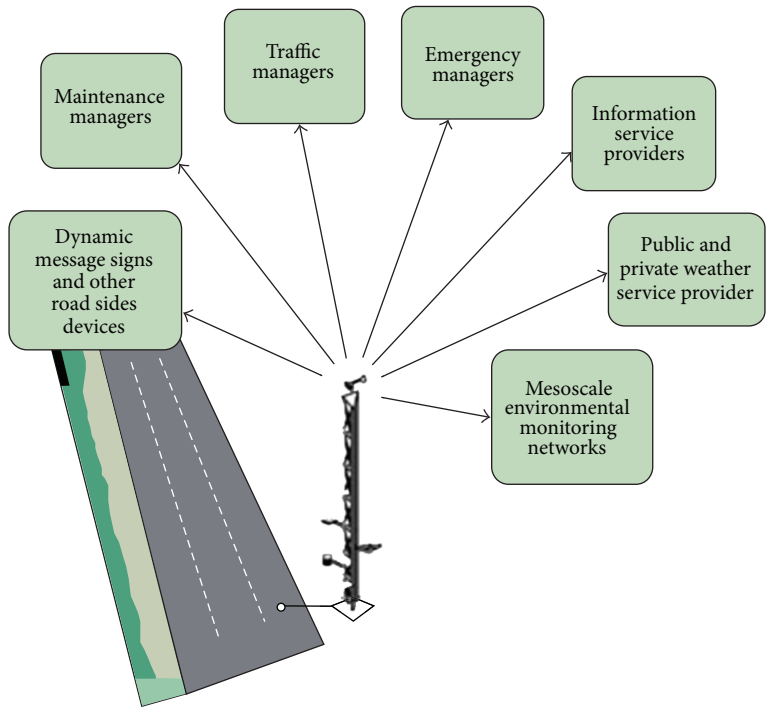

Figure 2: Operational application of environmental sensor station.

is of great significance in multiconstrained decision support, city navigation, user preference query, visualization of data mining, and so on under dynamic environment [7–9]. Hence, such query is consistent with practical application of data stream processing of smarter transportation. In addition, collection and analytical processing of transportation stream data present geographically distributed characteristic and are often influenced by uncertain sources such as wireless sensor networks, wireless radio frequency identification, location-based services, moving object management, and so on. Thus, data objects in data stream present uncertainty. Therefore, uncertain [10] real-time transportation data stream is characterized by difficult prediction, variability, rapid arrival, mass and infinite arrival, and so forth. Meanwhile, analytical processing of transportation stream data requires multiservice parallel processing and very high timeliness. In the environment of cloud computing, this paper combined the processing requirements for complex, parallel, and real-time transportation stream data and investigated continuous *Skyline* query algorithm with low cost, rapid response, and efficient scalability based on parallel processing framework of mass data. Compared with traditional *Skyline* query, *Skyline* query over uncertain transportation stream data faces the following challenges.

(1) In computational process of *Skyline* query on uncertain data stream, both dominant relations between computing objects and *Skyline* probability need to be calculated. However, traditional strategies fail to perform this process directly. Obviously, *Skyline* query calculation is CPU intensive [11, 12] and very high processing ability is required.

(2) Transportation stream data arrives continuously and is required to be processed immediately. So, when data stream is too rapid and users pay attention to a great number of objects (sliding window [13] is very large), traditional algorithm of centralized stream processing is difficult to satisfy the query demand.

Cloud computing with high storage capacity and calculating ability can fully satisfy application requirements of *Skyline* query on mass data. Main contributions of this research are as follows.

(1) Processing architecture of stratified transportation stream data is demonstrated.

(2) In the environment of cloud computing, this paper proposes the issue of continuous *Skyline* query on mass distributed uncertain transportation data stream and provides formal description.

(3) This research develops an *mMR-SUDS* algorithm based on sliding window division and the architecture proposed.

Section 2 introduces the processing architecture of stratified stream data of smarter transportation, background information, relevant work, and formal description of the problem. Section 3 explains design conception and optimization strategy of *mMR-SUDS* algorithm. Besides, experimental result comparison is demonstrated in Section 4, while summary of the entire research is made in Section 5.

2. Setting

2.1. Processing Architecture of Stratified Transportation Stream Data. In smarter transportation, processing architecture of stratified transportation stream data is shown in Figure 3. Bottom layer is front end of perceptible equipment consisting of N acquisition nodes for remote real-time data monitoring. Interlayer consists of M coordinator nodes connected to high speed network, while all transportation data processing centers are placed on top layer, providing transportation data services such as control, analysis, early warning, and so on.

2.2. Relevant Work. Early *Skyline* query is commonly applied to centralized database. Relevant researches mainly focus on centralized algorithms such as *block-nested-loops, BNL* algorithm [6]; *divide-and-conquer, D&C* algorithm [6]; *sort-filter-Skyline, SFS* algorithm [14]; *nearest neighbor, NN* algorithm [15]; *branch-and-bound Skyline, BBS* algorithm [16]; *bitmap* algorithm [17], and so on. Jian et al. [18] first proposed *Skyline* query technology on uncertain data and presented two query algorithms: bottom-up algorithm and top-down algorithm. In addition, in terms of uncertain data presentation, relevant researches usually pay more attention to discrete data. Therefore, according to literature [19], based on uncertain data at attribute level, three defined constraint methods including *uncertainty reduction*, *pairwise comparison,* and *adaptive bound tightening* were proposed to optimize *Skyline* query calculation.

In the field of *Skyline* query over data stream, aimed at continuous *Skyline* query based on sliding window model, literature [20] proposed Lazy algorithm and Eager algorithm which improves space and time efficiency using the method of advanced data cleaning. In addition, literature [21] investigated *Skyline* query of *n-of-N* data stream model in sliding window and proposed continuous *n-of-N* algorithm that improves system space performance by defining "key domination."

In the field of *Skyline* query over uncertain data stream, the data model in literature [22] was a data set consisting of certain objects where variable amounts of examples were presented for each object. And the concept of *Skyline* probability was proposed based on *Skyline* probability of examples for each object. Hence, the data model in literature [23] was virtually a discretionary version of uncertain attribute, while this paper focused on the case of uncertain tuple. On the other hand, literature [24] concentrated on static dataset, while this paper concentrated on data stream. Moreover, aimed at efficient *Skyline* calculation of uncertain data stream, literature [13] proposed *Skyline* query based on probability threshold and used the optimization methods like *Skyline* candidate sets and so on to execute continuous *Skyline* query efficiently. In contrast, literature [25] presented *Skyline* over probabilistic data stream algorithm. Based on grid index with better adaptability, heuristic rules such as probability delimitation, stepwise refinement, elimination in advance, optional indemnity, and so on were employed to optimize the algorithm temporally and spatially. By comparison, literature [26] investigated expectation evaluation of *Skyline* probability

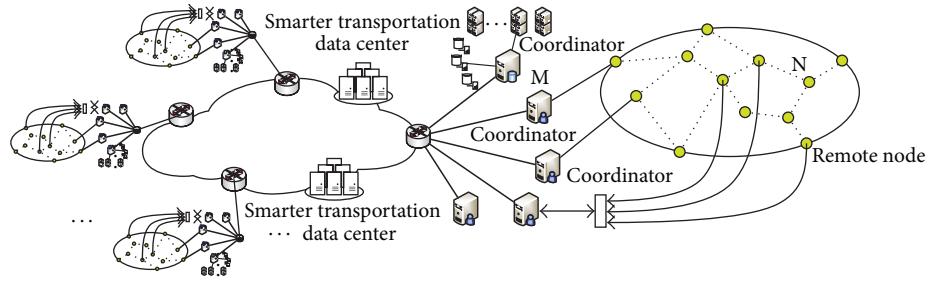

Figure 3: Processing architecture of stratified transportation stream data.

and presented the relation between probability threshold and expectation of *Skyline* probability.

In the field of distributed parallel *Skyline* query, current researches mainly focused on static data. Literature [27] suggested that integral query performance of system could be improved by defining execution order of *Skyline* query on each server. In addition, parallel distributed *Skyline* algorithm proposed in literature [28, 29] divided relevant sites into several groups by data division method and queries among groups were executed in parallel.

According to processing requirements of mass data, several existing relevant researches combined *Map Reduce* technology with *Skyline* query algorithm. Literature [4] proposed preview *Skyline* query algorithm and attempted to reduce size of input data in *Map* task and *Reduce* task through preview filtration. Thus, the performance of *Skyline* query based on *Map Reduce* framework was improved.

2.3. Terms and Definition

2.3.1. Data Stream. In a formal way, a data stream is any ordered pair (s, Δ) where s is a sequence of tuples and Δ is a sequence of positive real time intervals. For instance, there is a data stream with following tuple model in management system of road transportation stream (see Figure 4).

Road Stream is defined as data stream of tuple model processed in data stream management system. In the tuple model, attribute *Road Stream* denotes the name of the data stream, while *Vechicle_ID* denotes the unique identifier of a vehicle. Moreover, *X_Way* denotes road section of a vehicle; *X_Pos* presents the location of a vehicle; *Express_Way* denotes the expressway number; *Speed* denotes the current speed of a vehicle; *Timestamp* denotes that, when relevant information dispatched by a vehicle arrives at data stream system, system assigns a value to T according to the time sequence of received information.

2.3.2. Skyline

Definition 1. Skyline A point $p \in S$ is said to dominate another point $q \in S$, denoted as $p \prec q$, if (1) in every dimension $d_i \in D$, $p_i \leq q_i$; (2) in at least one dimension $d_j \in D$, $p_j \leq q_j$. The *Skyline* is a set of points $SKY(S) \subseteq S$ which are not dominated by any other point. The points in $SKY(S)$ are called *Skyline* points.

Definition 2. The *Skyline* probability of an instance p, that is, $\Pr_{sky}(p)$, is the probability that p exists and no instance of other uncertain objects that dominates p exists. Let m be the total number of uncertain objects and let $p \in O_k$; we have

$$\Pr_{sky}(p) = \Pr(p) \cdot \prod_{i=1, i \neq k}^{m} \left(1 - \sum_{q \in O_i, q \prec p} \Pr(q)\right). \quad (1)$$

Definition 3. Given a dataset S with n instances that belong to m uncertain objects and a probability threshold ϑ, the instance-level probabilistic *Skyline* analysis returns all instances with *Skyline* probabilities at least ϑ. That is, return the *Skyline* set S_{sky} such that

$$S_{sky} = \left\{p \in S \mid \Pr_{sky}(p) \geq \vartheta\right\}. \quad (2)$$

3. *Skyline* Query Algorithm (*mMR-SUDS*) of Uncertain Transportation Stream Data Based on *micro-batchinMap Reduce* Framework

3.1. Division of Sliding Window. According to the architecture of distributed transportation stream data processing, coordinator nodes collect continuous uncertain data stream monitored by each remote monitoring node. In this paper a cross method using count sliding window model divided whole sliding window so that data in the whole large sliding window of uncertain data stream are divided effectively. Then, data were distributed to various parallel computational nodes in order that each parallel computational node could actually correspond to a valid part of the whole sliding window. The basic conception was as follows: coordinator nodes dispatch arrived data successively to parallel nodes, and each parallel node maintains a count sliding window part. Thereby, the sliding window parts on all parallel nodes are combined across in turn, logically corresponding to the whole sliding window of uncertain data stream. And the corresponding relations are shown in Figures 5 and 6.

Roadstream	*Vechicle_ID*	*X_pos*	*X_way*	*Express_way*	*Dir*	*Speed*	*Timestamp*

Figure 4

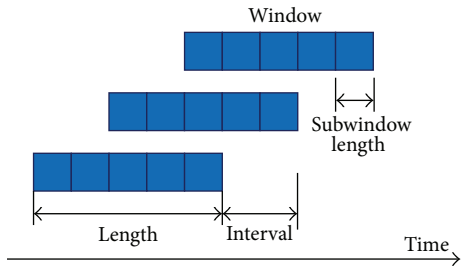

Figure 5: Sliding windows and Subwindows.

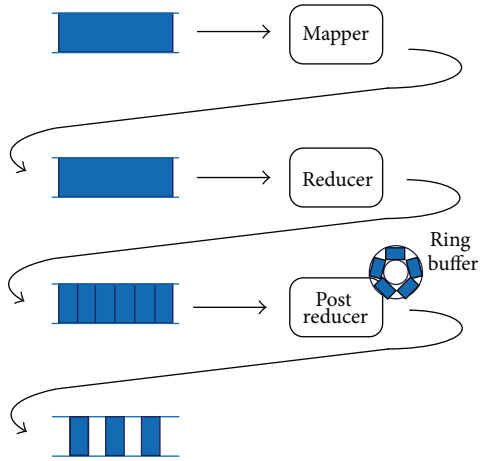

Figure 6: Sliding windows Implementation.

3.2. Processing Framework of Transportation Stream Data. Based on the sliding window division and *micro-batchinMap Reduce* model, processing framework of transportation stream data is designed in Figure 7. The framework consists of four types of nodes: *Coordinator* nodes that are responsible for reception of input data stream and data dispatch to *Map-PE* nodes (map-processing element); *Map-PE* nodes that are responsible for maintenance of data refresh in sliding window of *Map-PE* nodes and calculation of *Skyline* probability presented in the form of $MP_1, MP_2 \ldots MP_n$, which can mutually communicate with each other; *Reduce-Q* nodes (*reduce-query*) that are responsible for reception of *Skyline* results from each computational node; and Master nodes that are responsible for status maintenance of *Map-PE* nodes and *Reduce-Q* nodes. Besides, u, v, and w denote investigated uncertain data. According to processing framework of parallel data stream based on division of sliding window, *Skyline* query process of uncertain smarter transportation stream data is as follows.

(1) When uncertain data u arrives at *Coordinator* nodes, *Coordinator* nodes dispatch u to *Map-PE* node MP_1.

(2) MP_1 maintains renewed variation of *Skyline* probability caused by overdue data v and incoming data u in the window of *Map-PE* node. Then, MP_1 node dispatches overdue data v and newly incoming data u to other *Map-PE* nodes.

(3) Each *Map-PE* node maintains renewed variation of *Skyline* probability resulting from overdue data v and newly incoming data u in the window of each *Map-PE* node. This type of nodes is only in charge of updating *Skyline* probability and sending the updated results to *Reduce-Q* nodes. And all the parallel nodes dispatch feedback about *Skyline* probability of data u in the corresponding node to MP_1.

(4) Taking the feedbacks from all nodes about *Skyline* probability of data u into account, MP_1 calculates global *Skyline* probability of data u and outputs the result to query nodes.

(5) When new uncertain data w arrives at *Map-PE* nodes, *Map-PE* nodes dispatch w to MP_2 which performs the above mentioned process circularly.

3.3. mMR-SUDS Algorithm. The basic conception of *Skyline* query algorithm on uncertain transportation stream data based on *micro-batchinMap Reduce* framework is as follows. The task of updating *Skyline* probability of uncertain transportation data tuple in the whole sliding window is distributed to each parallel node. Then, parallelism among *Map-PE* nodes is employed to improve the operational efficiency of overall system. Hence, algorithm realization of all types of nodes is discussed in this section.

Coordinator nodes are responsible for data cache and data dispatch. Processing algorithm on *Coordinator* nodes is illustrated as follows.

Input. Uncertain data stream; response message of all parallel nodes,

Output. Data block of uncertain data.

(1) *Coordinator* nodes receive and then cache the incoming uncertain transportation stream data.

(2) If *Coordinator* nodes receive response message from a *Map-PE* node, the following results will be presented.

(2.1) New data block is obtained from the cache.

(2.2) Data are dispatched to the next *Map-PE* node.

Data cache and data dispatch are two procedures executed in parallel in the algorithm above mentioned. Communication of *Coordinator* nodes is followed by the corresponding *Reduce-Q* nodes. *Reduce-Q* nodes are in charge of receiving, synchronizing, and then displaying *Skyline* results dispatched

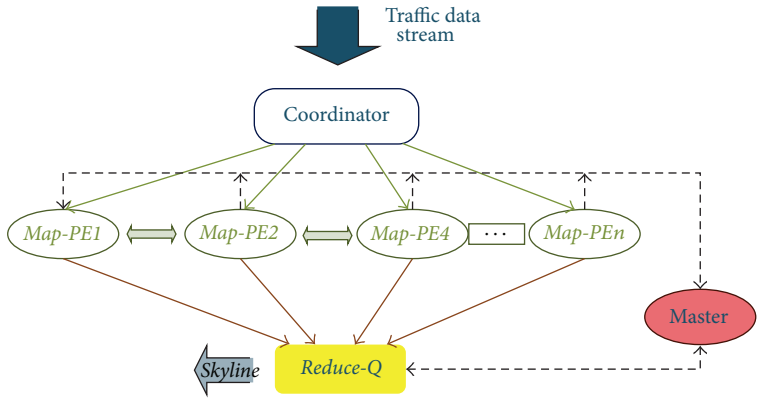

Figure 7: Processing framework of transportation stream data.

from all parallel *Map-PE* nodes. Processing algorithm on *Reduce-Q* nodes is as follows.

Input. *Skyline* results dispatched from all *Map-PE* nodes.

Output. Global *Skyline* results.

(1) *Skyline* results from all *Map-PE* nodes are received and cached.

(2) Received information is synchronized and global *Skyline* results are output.

Finally, taking MP_i as an example, processing algorithm on parallel *Map-PE* nodes is presented as follows.

Input. Data block of uncertain data; feedbacks from *Map-PE* nodes.

Output. Local *Skyline*; global *Skyline*.

(1) MP_i receives and analyzes information.

(2) If newly incoming data from *Coordinator* nodes are received, the results are as follows.

(2.1) New data tuple is obtained from information.

(2.2) Overdue tuple is obtained from the current window.

(2.3) *Skyline* probability variation caused by overdue data is updated.

(2.4) *Skyline* probability variation caused by newly incoming data is updated.

(2.5) Local *Skyline* probability of newly incoming data is calculated.

(2.6) *Skyline* probability variation in data block caused by dominance relation is calculated.

(2.7) Data block is added to local window.

(2.8) Updated information including newly incoming tuple and overdue tuple is dispatched to other *Map-PE* nodes.

(3) Otherwise, if updated information from a *Map-PE* node is received, the following results are presented.

(3.1) New data tuple is obtained from information.

(3.2) Overdue data tuple is obtained from information.

(3.3) *Skyline* probability variation caused by overdue data is updated.

(3.4) *Skyline* probability variation caused by the arrival of new data is updated.

(3.5) Local *Skyline* probability of new data is calculated.

(3.6) Feedbacks including *Skyline* probability of newly incoming tuple in this node are dispatched to nodes transmitting the updated information.

(3.7) Local *Skyline* results are dispatched to *Reduce-Q* nodes.

(4) Otherwise, if feedbacks from a Map-PE node are received, consolidated calculation is performed.

(4.1) New data tuple is obtained from information and local *Skyline* probability of new tuple is calculated.

(4.2) *Skyline* probability is updated.

(4.3) If feedbacks from all the Map-PE nodes are collected, the results are as follows.

(4.3.1) *Skyline* results are dispatched to *Reduce-Q* nodes.

(5) Otherwise, if node *i* receives unrecognized command, error message is presented.

3.4. The Optimization of Algorithm

(1) Reduction of Window Scanning Times. When analyzing the processing algorithm on parallel computational nodes, it can be found that three times of window scanning were presented, respectively, in procedures 2.3, 2.4, and 2.5. Besides, there were also three times of window scanning, respectively, in procedures 3.3, 3.4, and 3.5. To reduce window scanning times, three times of window canning can be integrated into one time scanning. Moreover, in each window scanning, data in the window is compared with new data and overdue data.

Thus, processing performance of the algorithm is improved by reducing window scanning times.

(2) Intermediate Filtration. Computational process of data *Skyline* probability shows

$$P_{sky}(\mathrm{a}) = P(\mathrm{a}) \times P_{\mathrm{old}}(\mathrm{a}) \times P_{\mathrm{new}}(\mathrm{a}). \quad (3)$$

That is, *Skyline* probability of tuple a $P_{sky}(\mathrm{a})$ equals the product of three probabilities including existing probability of tuple a $P(\mathrm{a})$, probability of tuple a not dominated by the data arriving earlier $P_{\mathrm{old}}(\mathrm{a})$ and probability of tuple a not dominated by the data arriving later $P_{\mathrm{new}}(\mathrm{a})$. Among the three probabilities, with new data arriving and old data expiring, $P_{\mathrm{old}}(\mathrm{a})$ increases continuously, while $P_{\mathrm{new}}(\mathrm{a})$ decreases constantly. Moreover, $P(\mathrm{a})$, $P_{\mathrm{old}}(\mathrm{a})$, and $P_{\mathrm{new}}(\mathrm{a})$ are all in the interval $(0, 1)$ throughout. Therefore, if $P_{\mathrm{new}}(\mathrm{a}) < \rho$, the relation that $P_{sky}(\mathrm{a}) < \rho$ is established. In addition, during the life cycle of a (time when a is in the sliding window), $P_{sky}(\mathrm{a}) < \rho$ is established permanently so it is unnecessary to calculate $P_{sky}(\mathrm{a})$. Hence, through the method of intermediate filtration, times of comparison are reduced and algorithm processing speed is accelerated owing to the fact that result set is far less than source dataset.

(3) Decrease of Idle Waiting Time of Nodes. It is presumed that all *Map-PE* nodes are provided with the same processing ability. t denotes average time that a node takes to communicate once with another node, while T denotes average time of one calculation update of *Skyline* probability except consolidated calculation. Besides, T' denotes average time of consolidated calculation. And the relation of the three is that $t < T < T'$. In basic scheme, calculation period of *Skyline* probability update caused by one data update is shown in Figure 8.

Figure 8 indicates that, when MP_1 receives newly incoming data, local *Skyline* probability update is achieved first and then updated information is dispatched to other *Map-PE* nodes. Therefore, MP_1 is completely in idle waiting state before consolidated calculation and idle waiting time of MP_1 is $T + t$. Similarly, it can be obtained that idle waiting time of other nodes is $T' + T + 2t$. So, in this condition, it takes $T' + 2T + 3t$ to complete an entire calculation period.

To decrease idle waiting time of all *Map-PE* nodes, when receiving newly incoming data, *Map-PE* nodes can dispatch updated information to other parallel nodes first and then calculate local *Skyline* probability for update. The revised calculation period is illustrated in Figure 9.

Figure 9 shows that, in optimized scheme, idle waiting time of MP_1 is t and that of other parallel nodes is $T' + 2t$. As a result, it takes $T' + 2T + 3t$ to achieve a complete calculation period. Therefore, compared with basic scheme, optimized scheme saves T in a calculation period.

4. Experimental Evaluation

Algorithm in this paper was realized using Java language and experiments were conducted in practical data-centered environment. Every processing node was configured with a CPU of Pentium4 with 2.0 GHz, a DDR memory of 2 GB, and Ubuntu operating system. Besides, synthetic data (characterized as independently distributed data) in literatures was adopted in experimental tests and existing probability of tuples followed Gaussian distribution. In synthetic data, data in all dimensions is mutually independent and presents uniform distribution in the interval $[0, 1]$. To test the real processing performance of *mMR-SUDS* algorithm, this paper presumed that *Coordinator* nodes cache numerous data tuples. When parallel nodes finish processing a batch of data and dispatch data request to *Coordinator* nodes, *Coordinator* nodes dispatch new stream data to parallel nodes for processing. In addition, probability threshold in experiments was set to 0.3 and window length was measured by data tuples contained in window with the value of 10000, 100000, 500000, and 1000000, respectively. Ranges of other experimental parameters were as follows: data dimension ranged from 2 to 6; the size of transmission data block (the number of data) was set to 1, 10, 100, and 1000, respectively, while the number of nodes participating in calculation was set to 1, 2, 4, 8, and 16, respectively. Each group of experiments was conducted 10 times and the average value was taken as the result. In contrast experiments, as a single machine algorithm, *Base* algorithm includes two nodes: a data cache node and a computational node. The data cache node is responsible for data cache and data dispatch to the computational node, while the computational node is responsible for the maintenance of sliding window and *Skyline* calculation. Besides, the computational node adopts the method of circularly dominating comparison. That is, once data arrives or expires, the computational node compares the incoming data or overdue data with all the data in sliding window and then updates *Skyline* probability.

Based on the experimental environment and experimental data above mentioned, the performance of *mMR-SUDS* algorithm was tested, respectively, in different sizes of transmission data block, window length, data dimension, and number of nodes.

4.1. Influence Tests of Transmission Data Block. Uncertain data stream is transmitted in data block between *Coordinator* nodes and *Map-PE* nodes as well as between *Map-PE* nodes. Therefore, size of transmission data block has a certain influence on algorithm realization. To evaluate such influence, this group of experiments tested the algorithm performance in different sizes of transmission block. In experiments, transmission data block was set to 1, 10, 100, and 1000, respectively; data dimension was set to 2, while window length was set to 1000000. Moreover, there were 16 *Map-PE* parallel nodes participating in the calculation.

Experimental results are demonstrated in Figure 10. With the constant increase of transmission data block, processing speed of *mMR-SUDS* algorithm tends to increase first and then decrease. The main reasons are as follows: when data block is small, overhead communication increases due to frequent data transmission, while when transmission data block is large, computation cost increases due to the increasing complexity of data dominated comparison in block. In conclusion, when size of transmission data block takes the

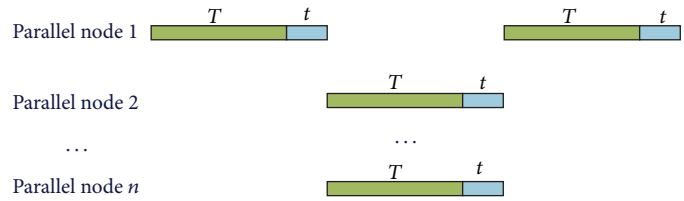

Figure 8: The computing cycle in the basic scheme.

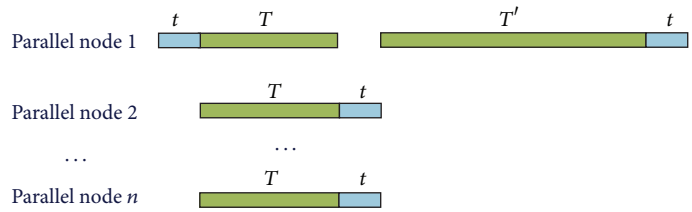

Figure 9: The computing cycle in the improved scheme.

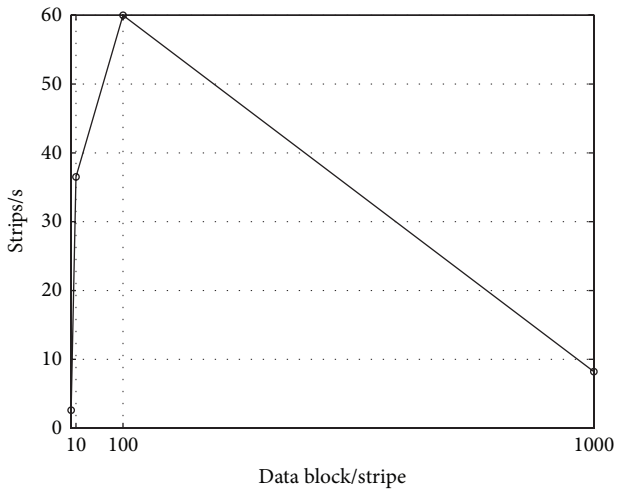

Figure 10: The effect of the size of transmitted data block.

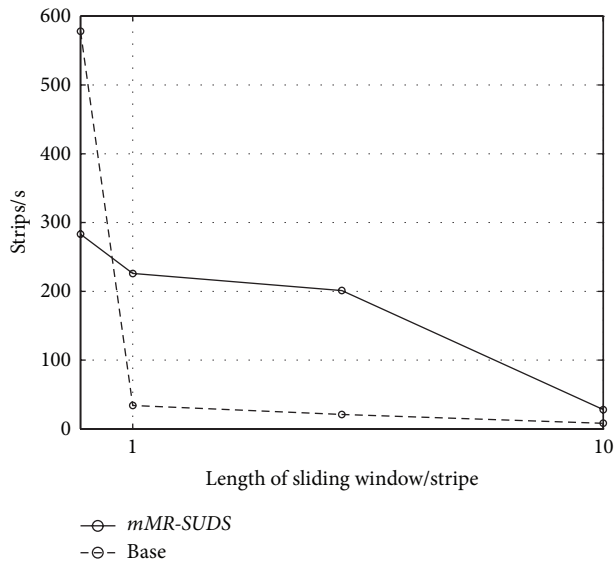

Figure 11: The effect of the length of sliding window.

middle value of 100, the algorithm provides good processing performance.

4.2. Tests of Window Scalability. In this group of experiments, data dimension was set to 2 and size of transmission data block was set to 100, while window length ranged from 10000 to 1000000. To compare the performance of *mMR-SUDS* algorithm with that of *Base* algorithm, 16 *Map-PE* parallel nodes participated in the calculation.

Experimental results are illustrated in Figure 11. As window length increases constantly, system processing performance declines rapidly. When window length is 10000, performance of *Base* algorithm is even better than that of *mMR-SUDS* algorithm. The main reasons are as follows. When window length is small, calculating performance of single machine fully satisfies the requirement of query processing. But in parallel algorithm, in terms of the whole parallel computing system, much time is taken to deal with problems such as communication, synchronization, and so on, although each node participating in calculation completes query processing rapidly. When window length is 100000 or more, single computational node could not fully satisfy the performance requirement of query processing. And for the whole parallel computing system, time overhead is mainly spent on calculation and parallel computing system begins to present the advantage of parallelism.

4.3. Tests of Dimension Scalability. To compare the dimension scalability of *mMR-SUDS* algorithm with that of *Base* algorithm, window length was set to 1000000 and there were 16 parallel processing nodes in system. In addition, data dimension value was in the interval [2, 11] and size of transmission data block was set to 100 in this group of experiments. And Figure 12 demonstrates the experimental

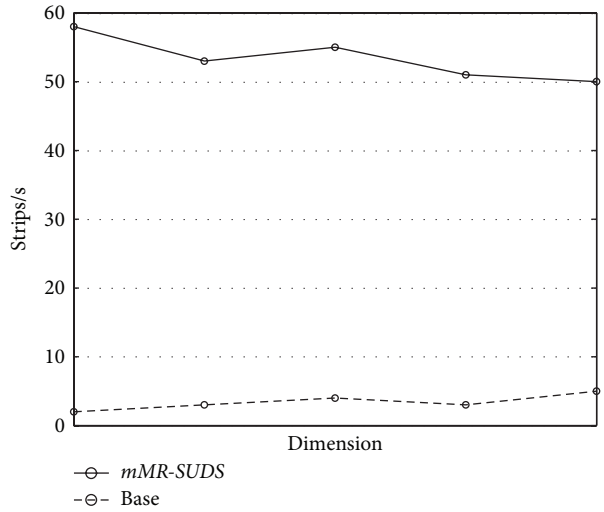

Figure 12: The effect of the dimension of data.

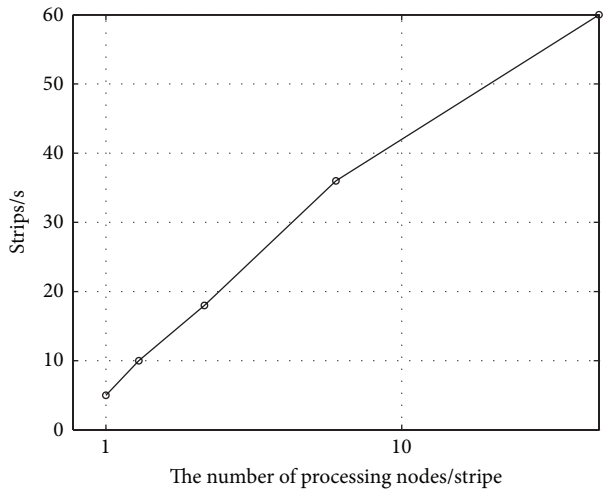

Figure 13: The effect of the number of processing nodes.

results. With the increase of data dimension, processing speed of both *mMR-SUDS* algorithm and *Base* algorithm declines slowly; but processing speed of *mMR-SUDS* algorithm is about 12 times higher than that of *Base* algorithm throughout. All in all, *mMR-SUDS* algorithm provides better dimension scalability.

4.4. Parallel Scalability Tests. To evaluate the parallel scalability of *mMR-SUDS* algorithm, this group of experiments tested processing performance of the algorithm in different numbers of nodes. In the experiments, the number of parallel nodes took the values of 1, 2, 4, 8, and 16, respectively, and total length of window was set to 1000000. Moreover, data dimension was set to 2, while size of transmission data block was set to 100.

Experimental results are illustrated in Figure 13. As the number of nodes increases continuously, processing speed of *mMR-SUDS* algorithm constantly increases, but the increasing range gradually decreases. The main reasons are as follows: with the increasing number of nodes, window length on each node decreases gradually. Hence, computation cost of each computational node gradually declines, while overhead communication gradually increases, which influences system processing performance. When the number of nodes took the value of 16, processing ability of *mMR-SUDS* algorithm was about 12 times better than that of single machine algorithm. And processing ability of *mMR-SUDS* in this case was far less than the theoretically optimum value which is as 16 times as that of single machine algorithm. When the number of nodes took the value of 2, processing ability of *mMR-SUDS* algorithm was the closest to the theoretically optimum value that was nearly twice that of single machine algorithm.

5. Conclusion

Aimed at *Skyline* query requirements of real-time uncertain data stream of smarter transportation with high capacity and large sliding window in the environment of cloud computing, this paper proposed a *Skyline* query algorithm *mMR-SUDS* over uncertain transportation stream data based on *micro-batchinMap Reduce* framework. Such algorithm transforms centralized processing problem of the whole global sliding window into the parallel processing problem of many nodes to their corresponding window by dividing data in sliding window. And such transformation effectively improves integral query processing performance. Experimental results show that *mMR-SUDS* algorithm presents not only high efficiency but, good scalability and load balancing. Therefore, such algorithm could satisfy the processing analysis requirements of various real-time transportation stream data.

In the parallel framework based on sliding window division, future research has to further optimize processing algorithm and improve algorithm processing performance using index structures such as grid, R tree and so on. Meanwhile, research scope of uncertain data shall be expanded to investigate *Skyline* query processing algorithm over uncertain transportation stream data at attribute level.

Conflict of Interests

The authors declare that they have no financial and personal relationships with other people or organizations that can inappropriately influence their work; there is no professional or other personal interest of any nature or kind in any product, service, and/or company that could be construed as influencing the position presented in, or the review of, the paper.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (Grant no. 61272029), the National Key Technology R&D Program (Grant no. 2009BAG12A10), and independent subject of State Key Laboratory of Rail Traffic Control and Safety, Beijing JiaoTong University (Contract no. RCS2009ZT007) and partially supported by the MOE key Laboratory for Transportation Complex Systems Theory and Technology School of Traffic and Transportation, Beijing, JiaoTong University.

References

[1] M. Armbrust, A. Fox, R. Griffith, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Rep UCB/EECS-2009-28, UC, RAD Laboratory, Berkeley, Calif, USA, 2009.

[2] IBM, "The Case for Smarter Transportation," Whitepaper, IBM-Institute for Business Value, 2010.

[3] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proceedings of the 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS '02)*, pp. 1–16, New York, NY, USA, June 2002.

[4] H. Guo, W. Wang, W. Guo, X. Jiang, and H. Bubb, "Reliability analysis of pedestrian safety crossing in urban traffic environment," *Safety Science*, vol. 50, no. 4, pp. 968–973, 2012.

[5] W. Wang, Y. Mao, J. Jin et al., "Driver's various information process and multi-ruled decision-making mechanism: a fundamental of intelligent driving shaping model," *International Journal of Computational Intelligence Systems*, vol. 4, no. 3, pp. 297–305, 2011.

[6] S. Borzsonyi, D. Kossmamm, and K. Stocker, "The skyline operator," in *Proceedings of the 17th International Conference on Data Engineering*, IEEE Computer Society, Washington, DC, USA, 2001.

[7] C. Jia, W. Xu, F. Wang, and H. Wang, "Track irregularity time series analysis and trend forecasting," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 387857, 15 pages, 2012.

[8] H. Wang, W. Xu, F. Wang, and C. A. Jia, "cloud-computing-based data placement strategy in high-speed railway," *Discrete Dynamics in Nature and Society*, vol. 2012, Article ID 396387, 15 pages, 2012.

[9] J. I. A. Chaolong, X. U. Weixiang, W. E. I. Lili et al., "Study of railway track irregularity standard deviation time series based on data mining and linear model," *Mathematical Problems in Engineering Volume*, vol. 2013, Article ID 486738, 12 pages, 2013.

[10] C. K. -S. Leung, "Mining uncertain data," *Wiley Interdisciplinary Reviews*, vol. 1, no. 4, pp. 316–329, 2011.

[11] G. Juve, E. Deelman, K. Vahi et al., "Scientific workflow applications on amazon EC2," in *Proceedings of the 5th IEEE International Conference on e-Science Workshops (e-science '09)*, pp. 59–66, Oxford, UK, December 2009.

[12] E. Wu, Y. Diao, and S. Rizvi, "High-performance complex event processing over streams," in *Proceddings of the ACM SIGMOD International Conference on Management of Data*, pp. 407–418, June 2006.

[13] W. Zhang, X. Lin, Y. Zhang, W. Wang, and J. X. Yu, "Probabilistic skyline operator over sliding windows," in *Proceedings of the 25th IEEE International Conference on Data Engineering (ICDE '09)*, pp. 1060–1071, IEEE Computer Society, Shanghai, China, April 2009.

[14] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *Proceedings of the 19th International Conference on Data Ingineering*, pp. 717–719, IEEE Computer Society, Bangalore, India, March 2003.

[15] D. Kossmann, F. Ramsak, and S. Rost, "Shooting stars in the sky: an online algorithm for Skyline ueries," in *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB '02)*, pp. 275–286, Hong Kong, 2002.

[16] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "An optimal and progressive algorithm for skyline queries," in *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD '03)*, pp. 467–478, San Diego, Calif, USA, June 2003.

[17] K. L. Tan, P. K. Eng, and B. C. Ooi, "Efficient progressive Skyline computation," in *Proceedings of the 27th International Conference on Very Large Data Bases (VLDB '01)*, pp. 301–310, San Francisco, Calif, USA, 2001.

[18] P. Jian, J. Bin, L. Xuemin et al., "Probabilistic Skylines on uncertain data," in *Proceedings of the 33rd International Conference on Very Large Data Bases (VLDB '07)*, pp. 15–26, Vienna, Austria, 2007.

[19] M. E. Khalefa, M. F. Mokbel, and J. J. Levandoski, "Skyline query processing for uncertain data," in *Proceedings of the 19th International Conference on Information and Knowledge Management and Co-located Workshops, (CIKM '10)*, pp. 1293–1296, Toronto, Canada, October 2010.

[20] Y. Tao and D. Papadias, "Maintaining sliding window skylines on data streams," *IEEE Transactions on Knowledge and Data Engineering*, vol. 18, no. 3, pp. 377–391, 2006.

[21] X. Lin, Y. Yuan, W. Wang, and H. Lu, "Stabbing the sky: efficient skyline computation over sliding windows," in *Proceedings of the 21st International Conference on Data Engineering (ICDE '05)*, pp. 502–513, IEEE Computer Society, Tokyo, Japan, April 2005.

[22] W. Xiaowei, H. Jiuming, and J. Yan, "Probabilistic skyline computation on distributed uncertain data," *Journal of Frontiers of Computer Science and Technology*, vol. 4, no. 10, pp. 951–961, 2010.

[23] X. Chuanfei, L. Shukuan, W. Lei, and Q. Jianzhong, "Complex event detection in probabilistic stream," in *Proceedings of the 12th International Asia Pacific Web Conference (APWeb '10)*, pp. 361–363, April 2010.

[24] X. Ding, X. Lian, L. Chen, and H. Jin, "Continuous monitoring of skylines over uncertain data streams," *Information Sciences*, vol. 184, no. 1, pp. 196–214, 2012.

[25] S.-L. Sun, D.-B. Dai, Z.-H. Huang, Q.-X. Zhang, and L.-X. Zhou, "Algorithm on computing skyline over probabilistic data stream," *Acta Electronica Sinica*, vol. 37, no. 2, pp. 285–293, 2009.

[26] J. B. Rocha-Junior, A. Vlachou, C. Doulkeridis, and K. Nørvåg, "Efficient execution plans for distributed skyline query processing," in *Proceedings of the 14th International Conference on Extending Database Technology: Advances in Database Technology (EDBT '11)*, pp. 271–282, Uppsala, Sweden, March 2011.

[27] Y. Yongtao and W. Yijie, "Towards estimating expected sizes of probabilistic Skylines," *Science China*, vol. 53, no. 1, pp. 1–18, 2010.

[28] B. Cui, H. Lu, Q. Xu, L. Chen, Y. Dai, and Y. Zhou, "Parallel distributed processing of constrained skyline queries by filtering," in *Proceedings of the 24th International Conference on Data Engineering (ICDE '08)*, pp. 546–555, IEEE Computer Society, Cancun, Mexico, April 2008.

[29] X. Ding and H. Jin, "Efficient and progressive algorithms for distributed skyline queries over uncertain data," in *Proceeedings of the 30th IEEE International Conference on Distributed Computing Systems (ICDCS '10)*, pp. 149–158, Genoa, Italy, June 2010.