A real-time monitoring and query framework for large-scale power system data

Zhenming Sheng*, Tiao Qian, Xueyun Wei, Xunhui Hu, Xiao Han, Liqiu Zhou
NARI Group Corporation, State Grid Electric Power Research Institute, NARI Technology Co., Ltd., Hangzhou, China

ABSTRACT

With the continuous improvement of automation technology, unmanned substations are developing rapidly throughout China. In order to more conveniently guarantee the basic operation and management of unattended substations, centralized control systems for substations are increasingly used in major real-time systems. In order to ensure the effective conduct of centralized control, State Grid Corporation of China proposed construction of a new generation of central control station equipment monitoring system. As a result, a huge amount of electricity data is generated in real time and need to be processed continuously. For instance, historical and current electricity data are analyzed differently, so that real-time analysis helps users to make quick decisions about centralized electricity regulation. This paper addresses the real-time processing and analysis of data in the centralized control system of a distributed substation from the perspective of real-time data processing. Specifically, our proposal is focused on accomplishing real-time distance processing of spatial-temporal data representations (graphs can represent many structured forms of data). To accelerate this computation, we propose a set of spatial indexing techniques, as well as the implementation of a complete KNN query system on this basis, and our approach has experimentally proven to be superior in terms of performance taken in constructing both.

Keywords: Real-time, data stream, KNN query, distance metric

1. INTRODUCTION

In electric power system, the substation occupies a very important position because it not only plays a vital role in the transmission and distribution but also is responsible for the assembly of electrical equipment, the adjustment of voltage, and the connection of important missions. State Grid Corporation of China has coordinated grid dispatching and equipment operation resources, incorporated substation equipment operation information realized integrated monitoring of grid load. Recently, with the country's power grid and substation equipment expanding scale, the contradictions between normal operation and maintenance management mode of equipment are increasingly prominent, while current technology is not satisfying.

It is well known that, huge amount of electricity data is generated in real time and needs to be processed continuously. For instance, users could make quick decisions or predictions about centralized electricity regulation through real-time analysis on historical and current information. The key observation in real-time processing system, is that each input tuple playing both the retrieval and update roles. Specifically, for such a system all existing tree indexes (like R-tree) can only meet the requirements in query efficiency, but do not perform well in update efficiency. Our work is focused on spatial indexing which can complete the update and query in massive amount of power graph data.

A crucial aspect of real-time graph processing is graph similarity calculation. There have been many related works on spatial data, roughly divided into three types: (1) how to measure the similarity between two graphs, such as Hausdorff[1], Fréchet[2], DTW[3], LCSS[4], EDR[5], ERP[6], DISSIM[7], etc. (2) how to perform graph similarity search on a large graph set[8-11]; (3) how to perform graph join on a large graph set[8, 12-14]. Without exception, all of the above works are processing graph data offline, so it is difficult to apply them directly to the centralized-control system.

The central control station monitoring system often processes generated graphs immediately, because outdated data may have a negative impact on the analysis results. We propose RSDSS (**R**eal-time **S**imilarity processing for **D**istributed **S**ubstation **S**ystems), our proposal includes:

* shengzhenming@sgepri.sgcc.com.cn

International Conference on Computer Application and Information Security (ICCAIS 2021), edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12260, 122601H · © The Authors. Published under a Creative Commons Attribution CC-BY 3.0 License · doi: 10.1117/12.2637526

(1) Spatial index-based graph similarity calculation is designed to improve the execution performance.

(2) We implemented the KNN query based on the index proposed above.

(3) We use a large number of real data sets to validate our approach and have implemented all of our proposed technologies based on *Flink* which deployed cases successfully on top of this.

2. PRELIMINARIES

2.1. Problem formulations

As mentioned above, we view RSDSS as a graph similarity query problem. In this section we first formalize the problem and give the meaning of the notations commonly used in this paper, as shown in Table 1.

Table 1. The notations.

Notation	Definition
$T = < t_1, t_2, \dots t_m >$	A graph consisting of *m* sample points.
t_i	The i-th sample point of the graph *T*.
$\mathbf{T} = \{T_1, T_2, \dots, T_w\}$	**T** is a graph set consisting of *w* graphs
T^i	T^i is the sub-graph of *T*, which can be represented as $T = < t_1, t_2, \dots t_m >$. T^m is also a sub-graph of *T*, which contains all points of *T*.
T^{i-j}	T^{i-j} is the sub-graph of *T*.
$TIME_{t_i}$	The timestamp of sample point t_i
$\|\|p, q\|\|$	Euclidean distance between the sampling points p and q.
$dis(p, T)$	The distance between sample point p and graph T. $\mathrm{dis(p, T)} = \min_{\mathrm{t_i} \epsilon \mathrm{T}} \|\|\mathrm{p}, \mathrm{t_i}\|\|$
D	The measurement method of graph similarity.
$HAU(T, Q)$	Hausdorff distance between graph *T* and *Q*.

Definition 2.1. (**Graph**). A graph is a sequence of sample points, denoted as: $T = < t_1, t_2, \dots t_m >$. and the graph set consisting of *N* graphs is represented as:

$$\mathbf{T} = \{T_1, T_2, \dots, T_w\}.$$

Definition 2.2. (**Hausdorff Distance**). In this paper, we use Hausdorff distance to measure the distance between graphs. The Hausdorff distance between graph$T = < t_1, t_2, \dots t_m >$and $Q = < q_1, q_2, \dots q_n >$ is defined as:

$$\mathrm{HAU(T, Q)} = \max\left\{\max_{\mathrm{t_i} \epsilon \mathrm{T}} \mathrm{dis(t_i, Q)}, \max_{\mathrm{q_j} \epsilon \mathrm{Q}} \mathrm{dis(q_j, T)}\right\}$$

Definition 2.3. (**KNN Query**). The inputs of KNN Query are a query graph *Q*, a set of graphs $\mathbf{T} = \{T_1, T_2, \dots, T_w\}$: a distance measurement method *D* and an integer *k*. KNN returns the set $\mathrm{S} = (Q, \mathbf{T}, D, K)$, where $|\mathrm{S}| = k$.

Definition 2.4. (**RSDSS**). The inputs of RSDSS are a definition of sliding window including window size and frequency of sliding as Figure 1 describes, is a data stream which consists of the sampling points of graphs, a distance measurement method *D* and an integer *k*. The graph sampling points in the window form a graph data set *T*. The output is also a data stream by which the receiver can calculate the current real-time $\mathrm{S}(Q, \mathbf{T}, D, K), \forall \mathrm{L} \in$ **T.**

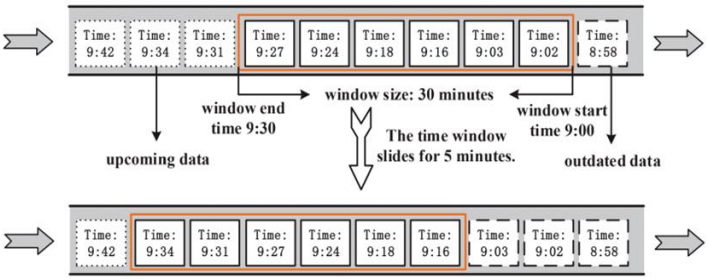

Figure 1. Sliding window demo.

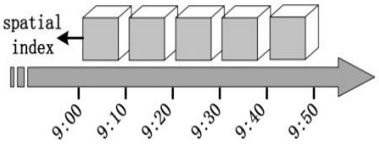

Figure 2. The spatial index with LCSS.

2.2. Solution overview

There are two methods to handle RSDSS, which are One-By-One and Micro-Batch Method, respectively. As shown in Figure 1, each time the window is slid, Micro-Batch Method updates the query result. As for One-By-One Method, every time a sample point reaching, the query result is updated, and after the window is slid, the outdated samples are removed and the query results is updated again. The output stream generated by One-By-One Method allows the receiver to maintain the current query result during all time, while Micro-Batch Method can only maintain discrete query results. For example, in Figure 1, Micro-Batch Method can only output the results of 9:00, 9:05, 9:10 and then. Figure 2 shows the spatial index with LCSS and the structure. Unlike reference[15], this paper uses One-By-One Method to implement RSDSS.

3. KNN QUERY BASE ON SPATIAL INDEX

3.1. Spatial index

KNN query is to find k graphs closest to the query graph Q in a graph set $\mathbb{T}$. The brute solution is to calculate the distance between each graph in $\mathbb{T}$ and Q, then sort, and finally find the top k graphs. In order to speed up the performance of the query, we will first build and update the index in real time.

3.2. KNN query

Compared with Range Query, KNN Query is more challenging to achieve, because KNN Query needs to find a value to make sure that there are more than k graphs in $\mathbb{T}$ whose distance from Q is less than this value. For Range Query, users will provide this value directly. The queries mentioned in the rest of this paper are all KNN Query.

The sample points with the same ID constitute the graph data of a moving object. Therefore, the sample points in the window can be regarded as a graph data set $\mathbb{T}$. RSDSS needs to perform once KNN Query for each graph in $\mathbb{T}$. $\mathbb{T}$ will change with the time window sliding, including some new sample points arriving at, some outdated sample points removed, even new graphs entering the window, or completely sliding out of the window. Changes in $\mathbb{T}$ will also cause the query results of graphs in $\mathbb{T}$ to change, and RSDSS can output these changes.

3.3. System implementation

Each graph is a query graph in RSDSS. It is forced to avoid broadcasting query graph, after partitioning the graph set $\mathbb{T}$ following some partition strategy. As analysis in Section 3, the pruning technique can limit the range of candidate graphs of a query graph to the inside of pruning area. Therefore, we can partition the graph set $\mathbb{T}$ by area. The global area is divided into four partitions P1 to P4, which are responsible for each sub-area separately. If a graph Q passes through some sub-areas, it is sent to the partitions which are responsible for these sub-areas called Q's *pass partitions*. For example, graph 3 passes through partitions P1 and P3, and graph 3 appears in pass of P1 to P3, which is a list recording the graphs passing through partitions. If the pruning area of a graph Q intersects some sub-areas, Q should also appear in those partitions, because there may be query results for Q in those partition, which is called Q's *Top-K partitions*, e.g., the pruning area of graphs 6 intersect P1, and graphs 6 appear in *Top K* of P1. Moreover, the master node needs to know how the global area is divided and which partitions in each graph passes through or intersects. It is necessary for each graph. While pruning area should be as small as possible, because the smaller pruning area is, the fewer sub-areas that intersect with pruning area, which can reduce the computational burden of child nodes effectively. While if graphs whose *pass partitions* and *Top-K partitions* are same, it is not necessary.

4. EXPERIMENTS

4.1. Experimental setup

4.1.1. Environment. We implement the approaches and conduct all the experiments on top of Apache Flink, and all algorithms are implemented in Java. To the best of our knowledge, none of the prior studies use such spatial index-based query to solve RSDSS. The Flink system is deployed on a cluster which runs CentOS 7.4 operating system and is equipped with 128 processors (Intel(R) Xeon(R) CPU E7-8860 v3 @ 2.20GHz). Overall, our cluster provides 120 computing nodes and a 512-core environment for the experiment.

4.1.2. Datasets. We used two kinds of graph data sets: *LBS-GJDL* and *LBS-MYL* to conduct comparative experiments. The two sets of data recorded the graph of vehicles in the city, and the parameters including vehicle ID, latitude and longitude, time stamp, etc.

4.1.3. Parameters. We used some parameters in the verification experiment listed in Table 2.

Table 2. The parameters.

Parameter	Definition
k	The *Top-K* closest to the query graph.
L	Length of the sliding window.
t	Used as the distance threshold after query terminating.
Throughout	The reciprocal of the time required to process a static data set using *Flink*.

4.1.4. Topology. After inputting data, the system creates an index structure based on the data, and then calculate the result that meets the query conditions based on the index. Among them, their parallelism is dynamic.

4.2. Experiment evaluation

Footnotes should be avoided whenever possible. If required they should be used only for brief notes that do not fit conveniently into the text. Although there are many distributed graph similarity query frameworks, none of them can be compared with RSDSS due to the different graph distance metrics supported by different frameworks. This section selects the graph similarity query framework DFT introduced in the literature as the main comparison object. In order to enable DFT to provide real-time similarity query service based on graph flow, DFT is modified in this section:

First, change the cutting technology in DFT. Second, increase the parallelism of the master node to solve the performance bottleneck caused by a single master node. This can show the contribution of incremental graph similarity calculation technology to improve query performance. The comparison of DFT+TSI and RSDSS can reflect the effect of continuous query technology on the performance of the framework. Figures 3 and 4 show the performance comparison of the three frameworks when *k* is used as an independent variable. As *k* increases, the computational complexity also increases, leading all throughput begins to decline. The performance ranking is: **RSDSS > DFT+TSI > DFT.**

Figures 5 and 6 show the performance comparison of the three frameworks when *L* is used as an independent variable. As the window size of the sliding window increases, the number of sample points of the graph in the window will increase. As *L* increases, the performance degradation of DFT+TSI and RSDSS is slower than that of DFT. This shows that the incremental calculation technology of graph similarity is more suitable for the calculation of incremental similarity between long graphs.

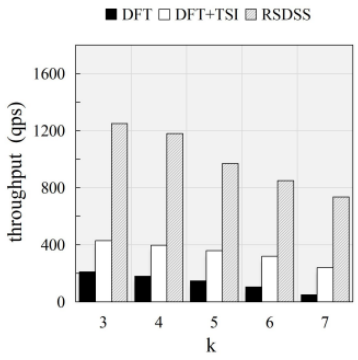

Figure 3. Throughout with k-change in LBS-GJDL.

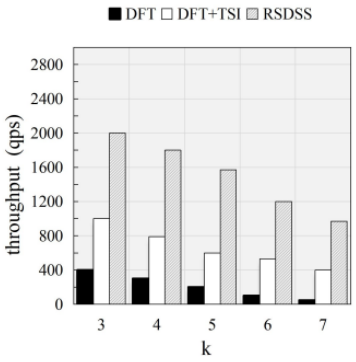

Figure 4. Throughout with k-change in LBS-MYL.

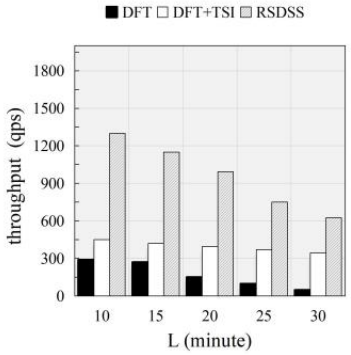

Figure 5. Throughout with L-change in LBS-GJDL.

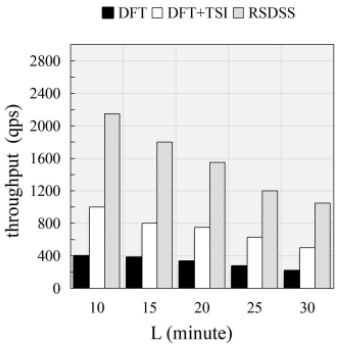

Figure 6. Throughout with L-change in LBS-MYL.

5. CONCLUSION

In this paper, we define the RSDSS problem and view it as a graph similarity query problem. Since huge amount of electricity data is generated in real time and need to be processed continuously. This paper addresses the real-time processing and analysis of data in the centralized control system of a distributed substation from the perspective of real-time data processing. We propose a real-time distance processing framework of spatial-temporal graphs. To accelerate this computation, we propose a set of spatial indexing techniques, as well as the implementation of a complete KNN query system on this basis, and our approach has experimentally proven to be superior in terms of performance.

ACKNOWLEDGEMENTS

This work was supported by funding of State Grid Corporation of China (Research of Key Technologies of Substation Digitalization Based on Centralized Control Mode, SGJSDK00SDJS2100615).

REFERENCES

[1] Hangouet, J. F., “Computation of the Hausdorff distance between plane vector polylines,” 12th Inter. Symp. on Computer-Assisted Cartography, (Charlotte, North Carolina), 1-10 (1995).
[2] Alt, H. and Godau, M., “Computing the Fréchet distance between two polygonal curves,” International Journal of Computational Geometry and Applications, 5, 75-91 (1995).
[3] Yi, B. K., Jagadish, H. V. and Faloutsos, C., “Efficient retrieval of similar time sequences under time warping,” ICDE, 201-208 (1998).
[4] Vlachos, M., Gunopulos, D. and Kollios, G., “Discovering similar multidimensional trajectory,” ICDE, 673-684 (2002).

[5] Chen, L., Özsu, M. T. and Oria, V., "Robust and fast similarity search for moving object trajectory," SIGMOD, 491-502 (2005).
[6] Chen, L. and Ng, R. T., "On the marriage of LP-norms and edit distance," VLDB, 792-803 (2004).
[7] Frentzos, E., Gratsias, K. and Theodoridis, Y., "Index-based most similar trajectory search," ICDE, 816-825 (2007).
[8] Shang, Z., Li, G. and Bao, Z., "DITA: Distributed in-memory trajectory analytics," Proc. of the 2018 Inter. Conf. on Management of Data, 725-740 (2018).
[9] Xie, D., Li, F. and Phillips, J., "Distributed trajectorysimilarity search," PVLDB, 1478-1489 (2017).
[10] Wang, S., Bao, Z., Culpepper, J. S., et al., "Torch: A search engine for trajectory data," 41st Inter. ACM SIGIR Conf. on Research and Development in Information Retrieval, 535-544 (2018).
[11] Zeinalipour-Yazti, D., Lin, S. and Gunopulos, D., "Distributed spatio-temporal similarity search," ACM 15th Conf. on Information and Knowledge Management, 14-23 (2006).
[12] Fang, Y., Cheng, R., Tang, W., Maniu, S. and Yang, X. S., "Scalable algorithms for nearest-neighbor joins on big trajectory data," ICDE, 1528-1529 (2016).
[13] Shang, S., Chen, L., Wei, Z., et al., "Trajectory similarity join in spatial networks," Proc. of the VLDB Endowment, 1178-1189 (2017).
[14] Zhang, S., Han, J., Liu, Z., Wang, K. and Xu, Z., "SJMR: Parallelizing spatial join with MapReduce on clusters," IEEE Inter. Conf. on Cluster Computing, 1-8 (2009).
[15] Ding, J., Fang, J., Zhang, Z., Zhao, P., Xu, J. and Zhao, L., "Real-time trajectory similarity processing using longest common subsequence," HPCC/SmartCity/DSS, 1398-1405 (2019).