QEVIS: Understanding and Diagnosing the Fine-grained Execution Process of Hive Query via Visualization

Fig. 1. In the Clouds: Vancouver from Cypress Mountain. Note that the teaser may not be wider than the abstract block.

Abstract— Understanding the query execution of distributed databases is crucial to many real-world practices such as detecting the query bottleneck and improving the system performance. Such analysis is always challenging due to the large volume of tasks executed in parallel and the complex task dependencies. Moreover, the unpredicted system behaviors also affect the query execution case by case. Existing techniques usually evaluate the distributed query by the statistics of system performance (e.g., CPU, memory, and disk I/O) or execution metrics(e.g., operator duration), which lost the fine-grained information at the atom task level. To tackle this problem, we propose QEVIS, a visual analytics system to interactively understand and diagnose the distributed query execution procedure from multiple levels. An optimized algorithm for the temporal directed acyclic graph(TDAG) layout is devised to show the overall query plan structure and execution process. A suite of novel visualization and interaction designs are integrated to estimate the correlation between the atom task trace and the system performance metrics. We illustrate the effectiveness of our QEVIS with three case studies from real-world applications and interviews with domain experts.

Index Terms—Radiosity, global illumination, constant time

1 Introduction

The distributed database system is becoming increasingly pervasive due to the explosive growth of data in science, industrial, life, etc. Meanwhile, many tools such as Hadoop [5], Flink [2], Myria and Vertical are developed to facilitate the query execution procedure including optimizing the query, translating traditional query language to logic execution plan, generating and dispatching tasks to clusters to perform the data acquisition in parallel.

To maximally leverage the distributed systems, it is crucial for users to understand and evaluate how the query runs across the clusters. The frequently asked questions include "Where does the time go?", "What is the bottleneck of my query?", "Can we improve the performance of the specific query?". Many research work devoted to evaluating and improving the performance of data analytic frameworks, but most of them try to reveal the performance by making high-level statistics about the correlated metrics collected from the execution logs or experiment conducted on benchmarks, which cannot be used for the understanding of the special case and provide the details answer for these questions case by case.

Understanding query logic and execution in the distributed environment is challenging. Three stages are involved for human-beings to understand a database query comprehensively: 1) understand the query logic, 2) understand the execution plan structure, and 3) understand the plan execution procedure. Understand the query logic itself is long studied direction, especially when the queries are becoming increasing deep and nested. With the pervasive of distributed database system, visual explaining of distributed execution plan structure is studied. These methods transform the abstract execution plan with thousands of lines of description to intuitive diagrams such as tree or graph and design interactions allowing users to interactively explore the nested structures. In our paper, we mainly focus on the stage 3) i.e. understand the execution procedure of query execution.There are three challenges to facilitate the fine-grained analysis of query executions.

Large number of atom tasks: When issuing the execution plan on the distributed database system, large number of atomic tasks will be generated and executed on the nodes in the cluster in parallel. The duration of tasks may be significant different from each other even executed on the same server. Identifying the unnature long cases and analyzing their performance from the large amount of tasks is challenges, since it

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

is difficult to define clear ground truth to fits all cases.

Complex dependencies among the tasks: The processing of a task always depends on several prerequisite tasks which provide the necessary input data. Analyzing the complex many to many dependencies and identifying the trace of interested are difficult in the large task set.

Unpredicted behavior of distributed system: also increases he difficulty to understand the model execution procedure. For instance, the developer find that the same query execution plan run today may be different from that of yesterday. In general, four aspects are considered to affect the performance of clusters: CPU usage, memory usage, network IO and disk status. Existing work studies try to reveal how these metrics related to the system performance or quantify the impact and significance of these features. These studies are conducted based on the observed performance data from the experiment or logs collected from the production environment. One work inspired us is QVA which tries to linkage the resources status to query performance and resource usages. However, these work fail to provide the fine-grained execution traces for users to inspect the reasons of model behavior.

In this work, we develop a visual analytics system called QEVIS (Figure 1) for database users to monitor, understand and diagnose query behavior on the distributed system. The system can be run with three modes: 1) monitoring mode: the system runs with the query execution process, collects and visualizes the query status in real-time; 2) simulation mode: the system will replay the execution process with given simulation rate; 3) analysis mode: the system will directly show the final results for users to explore the final results. In the visualization component, we design an optimized algorithm to layout the temporal directed acyclic graph(TDAG) which is used to display the execution plan and execution process dynamically and seamlessly. To enable the scalable visual analysis tasks executed on the clusters and their dependencies, we implement the task view which integrates the point cloud form and progress bar form together to meet the different analysis requirements, such as the anomaly identification, trace tracking. The system performance metrics are visualized in the monitoring view and linked with the other analysis views through a suit of flexible interactions.

- The design and implementation of QEVIS, a visual analytic system for understanding and analyzing the distributed query execution process.
- Well-established visualization front-end to support the interactive investigating, comparing and diagnosing the query process. The system includes a set of novel designs for visualizing the temporal DAG and sequence group.
- Case studies on the analysis of query process performed on the Hadoop platform.

2 Related work

2.1 Query analysis

Understanding the query behaviour and evaluting database performance has been studied for decades since the database management systems(DBMSs) have been found. Both database and visualization communities have proposed methods to analyze the performance and dignose the queires in automatic or manual way[][]. We give a brief introduction about the related works from the two following aspects: *query logic structure* and *query execution structure*, and refer the interested readers to [7] for the systematic overview about the database query debug and performance analysis.

Analyzing query structure. (SQL)Queries can be hard to read since they are always have a deep and nested structure. Many research works have been conducted to help the database users to quickly understand the quires. The most common method is to utlized visualization techniques to show the logic structure of operations[all]. For example, extended from previosu work, QueryVis utilzes the node-link diagrams to show the relationship between operators, the unambiguity is also proved in this paper. Other than the form of visualization, Gawade et al. proposed a method translate a query to Natural Language.

Analyzing query execution. After a query is issued by the database users, the query will be optimized and executed on the database platforms. Especially for the distributed database system, the query will be translated into the logic execution plans which is used to generating the physical tasks. Understanding the execution plans is important for users to expect the query performance. Many existing industrial sorfwares are developed to visualize the query execution process [6]. These softwares always utilize the gantt chart to show the progress and use Tree or directed acyclic graph(DAG) to show the relationship among the operators. VQA [21] displays the logic of query plan as a tree with the nodes indicating operators and the edge indicating the dataflow. Barcharts are inserted into the node to show the mertric of the operator(e.g., execution time, memory allocated). Perfopticon [16] display the overall plan structure from two levels: fragment level and operator level. The system also allow users to observe the execution trace of fragment or operator across the workers.

2.2 Visualization for sequence data

Nowadays, large amount of sequence data are generated from a variety of applications such as health care [14, 22], social media [13, 25], and education [3, 4, 9, 11, 17]. As a special type of time-series data, event sequence record a series of discrete events in the time order of occurence [10]. For the detail taxnomy about the time-series and event sequence visualization, we refer the readers to read the surveys [10, 20].

Sequence visualization are designed to reveal the information of event such as the event type, start time, end time and duration. Moreover, for the complex application requirement, various of the visualization tasks are proposed such visual summarization, prediction & recommendation, anormaly analysis and comparison. Existing visualization techniques can be classfied into five categories according to the form of visual represention, i.e., *sankey-based visualization*, *hierarchy-based visualizations*, *chart-based visualizations*, *timeline-based visualizations* and *matrix-based visualizations* [10].

Hierarchy-based visualizations [8], ***sankey-based visualizations*** and ***matrix-based visualizatios*** are always designed for displaying the sequence or sequence collection after modeling the them as special structures such as graph or tree. For example, LifeFlow [23] utilizes the tree structure with a node presenting a group of events to summarize the sequences. Outflow [22] models the progression paths of sequences as directed acyclic graph with a node indicating a cluster of states, and then visualize the graph as sankey diagram [19]. These methods always provide the highly abstract summarization for sequences and cannot directly reveal the pattern of specific individual sequence. Matrixwave [26] uitilzes a sequence of matrix to show the connections between specific events, which can provide details connecting information of sequence. However, Matrixwave loss the detail temporal information and cannot be used to very large sequence collections.

Chart-based visualizations uses barchart, linechart or scatter plot to visualize the trend or distribution of events, which always works as assisting views to support the interactive explorations. For instance, barchart and linechart are always used to show the attributes distribution of sequences or temporal trends [1, 8]. Scatter plot can be used to show the overview of coarse-level overview of the sequence or sequence groups by project them to 2D canvas through dimension reduction algorithms or two specific attributes [8, 15, 24]. Other than the distribution of sequences, the scatter plot can also reveal the outliers.

Timeline-based visualizations are known as the most intuitive ways which demostrate the events in a time order. Gantt chart is a direct way to show the temporal information of event sequences, including the start time, end time and duration. Many industrial tools such as the TezViz uses gantt diagram to clearly demostrate progres of the operations. Lifeline [18] use Gantt diagram to display the sequence as well as the events and each sequecne takes a single row. LiveGantt [12] proposes an algorithm to visualize the scheduling events with better scalability. However, these methods cannot directly be used in our application since the dependencies of these sequences are ignored. Moreover, the gantt chart also suffers the series scalability problem when applied it to large sequence dataset and the abnormal sequence will be hidden without aligment.

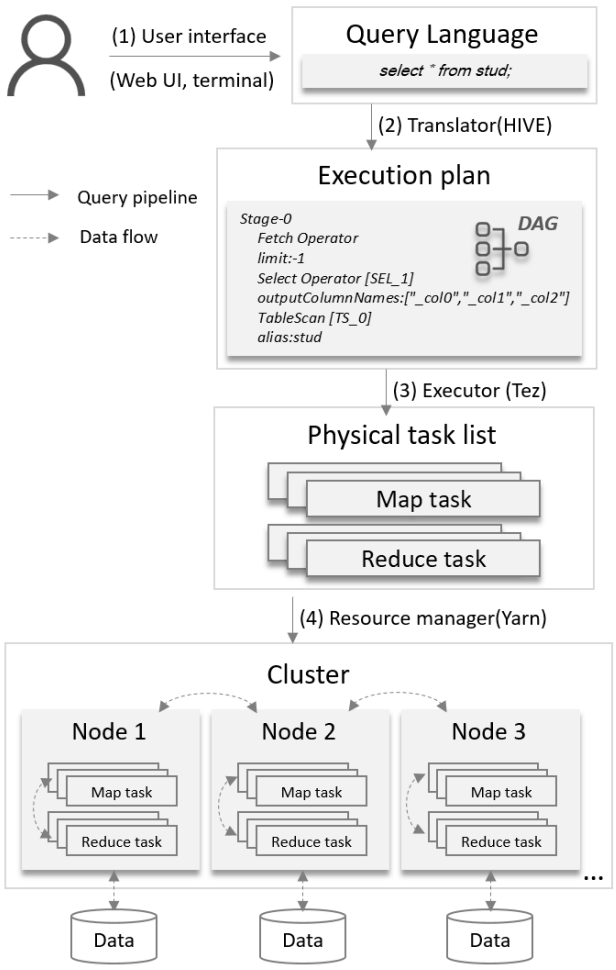

Fig. 2. Overview of distributed query pipeline(Hadoop 2.0).

3 BACKGROUND

3.1 Background of distributed query execution

The architecture and terms are introduced in this section to serve the as basis for the further discussions. The figure 2 demonstrates how a query is processed by a distributed query system. We use Hive based Hadoop architecture in this paper, and the analysis pipeline can be easily extended to other systems.

When user issues a query (shown as Figure 2(1)) through the interface such as a web-based user interface or SQL terminal. Hive uses a cost-based optimizer to optimize the query such as determining the best methods for scan operators, join orders and aggregate operation, and then translate it as the logic execution plan shown as Figure 2(2). The logic exaction plan may contains hundreds of lines of description, which describes the execution process as a Directed Acyclic Graph(DAG) that can be processed by the executor(e.g., Tez).

A DAG is a collection of vertices and edges. Logically, a **vertex** consists of a sequence of logical operators(such as filter, aggregate, etc) which describes the execution of a part of the query. Further more, there are two types of vertices in the Tez DAG, **map** vertex and **reduce** vertex. In the DAG, the **edges** define the data movement between the adjacent vertices. We name of source vertex as the **producer** vertex and the target vertex as the **consumer** vertex. Noticed that a producer vertex may connect to multiple consumer vertices and a consumer vertex could be connected by multiple producer vertices.

With a given vertex, Tez further creates a set of atomic **tasks**(shown as Figure 2(3)). Then these tasks will be dispatched on the physical machines by Yarn, the resource manage tool of Hadoop2.0. A task takes a piece of data as input and execute all operators defined in the corresponding vertex. To ease the analysis of tasks, we define the sub-processes of a task as five steps. For a map task, the steps are *Initialization*, *Input*, *Processor*, *Sink*, *Spill*. For a reduce task, the second step is *Shuffle* instead. Moreover, the data read by a task could come from the local file or producer tasks.

3.2 Requirement analysis

During one year of collaboration, we have closely collaborated with three experts in distributed database, who are also the co-authors of this paper.

In the first month of the collaboration, we have held brainstorming to collect the most frequent raised questions when analyzing the distributed query system performance. Based the discussions with domain experts and review of existing literature, we have formulated the following design requirements.

R1 **Understand the general query execution progress and query plan structure.** Before our collaboration, the domain experts have used profiling software (Tez UI, etc) or visualization tools(Tableau, etc) to show the query progress as Gantt chart and query plan structure as directed graph. However, these two visualizations are always displayed in separated views which require users to switch their focus thus break the continuity of exploration.

R2 **Understand the query process at the task level.**Tez task is the atomic level of executions because ny failure operation in the task will lead to the re-run of whole task. Understand the execution of single task can be helpful to identify the bottleneck of the whole query process. However, visualize the tasks is challenge. First, to visualize the tasks in traditional way(Gantt chart) need a very large rendering space. Multiple features such as the size of input/output data and the operators should be visualized for understanding the task. Moreover, the many to many relationship among the tasks also makes it difficult to design clear and **scaleble** visualization.

R3 **Provide the visual insight to reason the behaviour and pattern of a specific task.**To solely visualize the tasks themselves are not enough to explain the specific pattern of tasks. Many performance of hardware resource such as the network status, hard disk waiting list is also related to the patterns. Such kinds of information should be vitalized effectively to assist the exploration of query executions.

R4 **Support interactive exploration.** Other than the visualization designs, a flexible interaction should be implemented for users to navigate to any time range, vertex, task group or single task of interest. The linkage among the correlated visual elements are also should be considered om the design to coordinate the information.

3.3 Task analysis

Guided by the aforementioned requirements, we discussed with the domain experts about the visualization form and distilled the following visualization tasks:

T1 **Visualize the execution process and query plan structure effectively.** To guarantee the continuity of exploration(R1), the process and plan structure should be integrated into one visualization view. Several criteria should be considered such as the minimize usage of canvas, minimize the cross of links and provide clear topology structure.

T2 **Effectively visualize the information of tasks.** To facilitate the fine grained exploration of query execution(**R1**, **R2**), the information about the tasks should be visualized, including: the size of data processed by the task; the data-flow among the tasks; the temporal information of task(start the time, end time, duration, etc) and the corresponding sub-process. Moreover, the abnormal(tasks taking longer time) tasks and the specific execution trace should be easily observed.

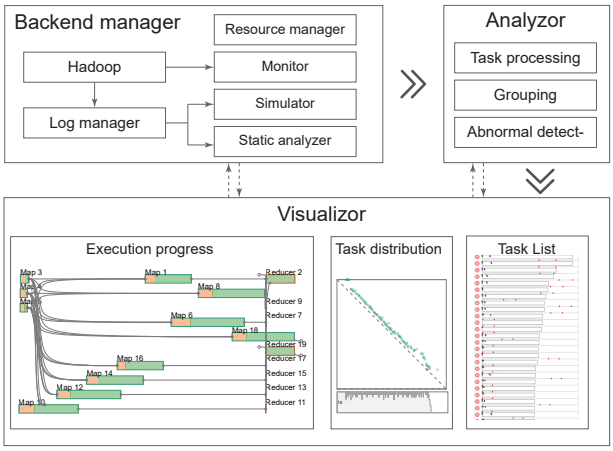

Fig. 3. QEVIS system consists of three major components: backend manager, analyzer and visualizer.

T3 Visualize the machine status. Display the machine status such as network status, disk IO pending list, CPU usage and Memory Usage will be useful to investigate the characters of task, and reasoning the patterns of the query execution(**R3**). These performance metrics should be well linked with the specific patterns of tasks.

T4 Interaction and linkage. System should provide the flexible interactions allowing users to switch the focus among the different point of interest, such as a specific time range, a vertex or a group of tasks(**R2**, **R3**, **R4**). For example, user may select a vertex and explore if the tasks in this vertex are CPU-bound or I/O-bound. This requires the visualization to show the related tasks when choosing a vertex and highlight the corresponding CPU usage and disk information simultaneously.

4 System Design

As shown by Figure 3, QEVIS consists of three modules: backend manager, analyzer, and visualizer.

In our current system, we use Hadoop2.0 as the **distributed query engine**. The system can run in three modes: monitoring mode, simulation mode and analytics mode. The monitoring mode directly collect the real-time execution log from Hadoop2.0, then process it and send the result to analyzer. The simulator and log analyzer are linked with log file manager, a module to process and save logs as as structured data form. Log analyzer directly output the strusctured data to the next module. The simulator simulatses the execution progress which allow users to adjust the running speed and explore the dynamic query process.

The analyzer collects the data from the backend manager and further process them for visualization.

The visualization module integrates coordinated views to support interactive exploration of and reasoning about the query behaviour at the different perspectives. Execution overview demostrates the execution process in at the vertex level. A new algorithm for the temporal DAG is proposed to visualize the structure and process at the same time. The task view visualize the temporal information of tasks as well as the dataflow relationship among the tasks. The profiling view integrates the task view with the resource status for each machines. A rich set of interactions are also supported to link different views together.

5 Data Process

6 Visualization Design

A paragraph to introduce the general design consideration. Overview first, zoom ....

6.1 Query progress view

- Traditional method: Gantt diagram and design consideration
- Proposed algorithms, link processing
- Alternative design
  - Gantt(consider the vertical edge, large space, unclear structure)
  - New design loose(clear structure, large space)
  - New design compact(small space, unclear structure)

6.2 Task view

- Design consideration. Why gantt doesn't work(need very large space, difficult to show the data-flow link, difficult to show and select the abnormal cases)
- Our design
- Compare with design alternative (Gantt)

6.3 Profiling view

- Our design consideration
- Design: Embedding task view to profiling view
- Design: Visualize the profiling results

6.4 Interactions

- Multi-scale navigation
- Inner and inter linking

7 Evaluation

7.1 Case study

7.2 Expert interview

8 Conclusion

Acknowledgments

The authors wish to thank A, B, and C. This work was supported in part by a grant from XYZ (# 12345-67890).

References

[1] B. C. Cappers and J. J. van Wijk. Exploring multivariate event sequences using rules, aggregations, and selections. *IEEE transactions on visualization and computer graphics*, 24(1):532–541, 2017.

[2] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas. Apache flink: Stream and batch processing in a single engine. *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, 36(4), 2015.

[3] Q. Chen, Y. Chen, D. Liu, C. Shi, Y. Wu, and H. Qu. Peakvizor: Visual analytics of peaks in video clickstreams from massive open online courses. *IEEE transactions on visualization and computer graphics*, 22(10):2315–2330, 2015.

[4] Q. Chen, X. Yue, X. Plantaz, Y. Chen, C. Shi, T.-C. Pong, and H. Qu. Viseq: Visual analytics of learning sequence in massive open online courses. *IEEE transactions on visualization and computer graphics*, 26(3):1622–1636, 2018.

[5] A. S. Foundation. The apache hadoop project.

[6] A. S. Foundation. Tez ui.

[7] S. Gathani, P. Lim, and L. Battle. Debugging database queries: A survey of tools, techniques, and users. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pp. 1–16, 2020.

[8] D. Gotz, J. Zhang, W. Wang, J. Shrestha, and D. Borland. Visual analysis of high-dimensional event sequence data via dynamic hierarchical aggregation. *IEEE transactions on visualization and computer graphics*, 26(1):440–450, 2019.

[9] M. C. Goulden, E. Gronda, Y. Yang, Z. Zhang, J. Tao, C. Wang, X. Duan, G. A. Ambrose, K. Abbott, and P. Miller. Ccvis: Visual analytics of student online learning behaviors using course clickstream data. *Electronic Imaging*, 2019(1):681–1, 2019.

[10] Y. Guo, S. Guo, Z. Jin, S. Kaul, D. Gotz, and N. Cao. Survey on visual analysis of event sequence data. *arXiv preprint arXiv:2006.14291*, 2020.

[11] H. He, B. Dong, Q. Zheng, and G. Li. Vuc: Visualizing daily video utilization to promote student engagement in online distance education. In *Proceedings of the ACM Conference on Global Computing Education*, pp. 99–105, 2019.

[12] J. Jo, J. Huh, J. Park, B. Kim, and J. Seo. Livegantt: Interactively visualizing a large manufacturing schedule. *IEEE transactions on visualization and computer graphics*, 20(12):2329–2338, 2014.

[13] P.-M. Law, Z. Liu, S. Malik, and R. C. Basole. Maqui: Interweaving queries and pattern mining for recursive event sequence exploration. *IEEE transactions on visualization and computer graphics*, 25(1):396–406, 2018.

[14] S. Malik, F. Du, M. Monroe, E. Onukwugha, C. Plaisant, and B. Shneiderman. Cohort comparison of event sequences with balanced integration of visual analytics and statistics. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, pp. 38–49, 2015.

[15] S. Malik, B. Shneiderman, F. Du, C. Plaisant, and M. Bjarnadottir. High-volume hypothesis testing: Systematic exploration of event sequence comparisons. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(1):1–23, 2016.

[16] D. Moritz, D. Halperin, B. Howe, and J. Heer. Perfopticon: Visual query analysis for distributed databases. In *Computer Graphics Forum*, vol. 34, pp. 71–80. Wiley Online Library, 2015.

[17] X. Mu, K. Xu, Q. Chen, F. Du, Y. Wang, and H. Qu. Moocad: Visual analysis of anomalous learning activities in massive open online courses. In *EuroVis (Short Papers)*, pp. 91–95, 2019.

[18] C. Plaisant, B. Milash, A. Rose, S. Widoff, and B. Shneiderman. Lifelines: visualizing personal histories. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 221–227, 1996.

[19] P. Riehmann, M. Hanfler, and B. Froehlich. Interactive sankey diagrams. In *IEEE Symposium on Information Visualization, 2005. INFOVIS 2005.*, pp. 233–240. IEEE, 2005.

[20] S. F. Silva and T. Catarci. Visualization of linear time-oriented data: a survey. In *Proceedings of the first international conference on web information systems engineering*, vol. 1, pp. 310–319. IEEE, 2000.

[21] A. Simitsis, K. Wilkinson, J. Blais, and J. Walsh. Vqa: vertica query analyzer. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pp. 701–704, 2014.

[22] K. Wongsuphasawat and D. Gotz. Outflow: Visualizing patient flow by symptoms and outcome. In *IEEE VisWeek Workshop on Visual Analytics in Healthcare, Providence, Rhode Island, USA*, pp. 25–28. American Medical Informatics Association, 2011.

[23] K. Wongsuphasawat, J. A. Guerra Gómez, C. Plaisant, T. D. Wang, M. Taieb-Maimon, and B. Shneiderman. Lifeflow: visualizing an overview of event sequences. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 1747–1756, 2011.

[24] J. Wu, Z. Guo, Z. Wang, Q. Xu, and Y. Wu. Visual analytics of multivariate event sequence data in racquet sports. In *2020 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 36–47. IEEE, 2020.

[25] J. Zhao, N. Cao, Z. Wen, Y. Song, Y.-R. Lin, and C. Collins. # fluxflow: Visual analysis of anomalous information spreading on social media. *IEEE transactions on visualization and computer graphics*, 20(12):1773–1782, 2014.

[26] J. Zhao, Z. Liu, M. Dontcheva, A. Hertzmann, and A. Wilson. Matrixwave: Visual comparison of event sequence data. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pp. 259–268, 2015.