Visual Analysis of Parallel Interval Events

J. Qi, C. Liu, B.C.M. Cappers, and H. van de Wetering

Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, Netherlands

Abstract

System logs typically contain lines with time stamps that each describes an event. Where these events semantically form start and end events, they can be combined into interval events. For visual event analytics, the analysis of interval events is more complex than that of point events, since not only the order of events, but also temporal overlaps have to be taken into account. To address this increased complexity and for the purpose of system understanding and analysis, we present SELE, a domain-independent tool for visualizing parallel interval events. SELE is intended to be used on a single long trace of events. A visual technique named strata timeline is developed to handle visual scalability issues. Finally, a multi-core parallel graph searching algorithm is analyzed to demonstrate SELE.

CCS Concepts

•***Human-centered computing*** → ***Visual analytics; Visualization systems and tools;*** •***Information systems*** → *Data analytics;*

1. Introduction

For many systems (e.g. software systems and network traffic systems), tasks of system comprehension, maintenance, and performance evaluation are typically goal-oriented and highly related to actual system behavior during executions [Bal99]. Therefore, these tasks benefit from analyzing system logs. However, system log analysis may be challenging; not only due to the size of the log, but also due to the complexity of the system. In this paper, we apply visualization to assist users in exploring a log in order to make discoveries and identify problems.

System events can be categorized into two groups, point events and interval events; they differ in whether the logger considers the events to occur on a single time point or during a period of time. Interval events are more difficult to visualize than point events, because not only the temporal order but also temporal overlap has to be taken into account. Furthermore, systems nowadays are quite often inherently parallel, resulting in more overlapping events, and consequently more visualization challenges.

In this paper, we introduce a domain-independent visualization tool named SELE (System Event Log Explorer) for visually exploring parallel interval event logs. By applying SELE, users working on an unfamiliar system are expected to quickly understand the system for further problem addressing. Also, we develop a novel visual technique, *strata timeline* to handle visual scalability issues.

The remainder of this paper is organized as follows. User tasks and the data are described in Section 2 and Section 3, respectively. Section 4 briefly reviews related work. Section 5 explains our visual design. Section 6 presents a use case and elaborates on the user tasks. Conclusions and future work are presented in Section 7.

© 2018 The Author(s)
Eurographics Proceedings © 2018 The Eurographics Association.

2. User Tasks

With this work, we particularly target domain experts who may work on unfamiliar systems and need to address higher-level concerns, by identifying lower-level problems. For example, for network attack prevention, abnormal network behaviors should be first detected and then resolved. Hence, we expect SELE to assist in:

- **Task 1**: understanding the basics of the system, and
- **Task 2**: identifying problems.

Table 1: Typology of the User Tasks; words in italics indicate task descriptions as used in [BM13].

Why	*Discover* → *Explore* → {*Summarize* the basics of the system (Task 1), *Identify* problems (Task 2)}.
How	*Encode* + *Select* + *Navigate* + *Arrange* + *Change* + *Filter*.
What	*Input*: system execution logs; *Output*: timeline visualizations, statistical information of the system, temporal patterns, abnormal behaviors.

The user tasks are further elaborated based on the multi-level typology of abstract visualization tasks introduced by M. Brehmer et al., by answering three questions: *why* the task is performed, *how* the task is performed, and *what* the task input and output are [BM13]; see Table 1. In Section 6, we present a use case as a real example of our user task.

DOI: 10.2312/eurovisshort.20181074

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

```
<event>
    <!-- Interval information -->
    <string key="software:starttimenano" value="131215"/>
    <string key="software:endtimenano" value="132857"/>

    <!-- Attributes depends on domains -->
    <string key="lifecycle:transition" value="Start"/>
    <string key="concept:name" value="HANDLE_SUC"/>
    <int key="software:threadid" value="6"/>
</event>
```

Figure 1: *An event in our sample input data*

3. The Data

Input data for SELE is a long trace of interval events that are collected from a parallel system. Figure 1 shows a sample of events in the input dataset used in our use case in Section 6. The data is formatted by the XES (eXtensible Event Stream) standard [Gün09].

In this work, an interval event $e = (\tau, \alpha)$ consists of two parts: an interval τ indicating the time duration, and an attribute vector α containing further attributes, depending on the system that recorded the log. While the interval information is obligatory, there are no requirements for α. In that sense, SELE is domain-independent.

We call two events *disjoint* if their intervals are disjoint, otherwise, they are called *parallel*. For two parallel events, if the interval of one of them is a subset of the interval of the other, we call these events *fully* parallel, otherwise *partially* parallel. In some related work, limited visual scalability occurs, particularly for partially parallel events. We further discuss this point in Section 5.

Some visualizations, like EventFlow [MLL*13], visualize collections of short traces. However, we focus on visualizing a single long trace collected from a parallel system. Hence, the input for SELE is an event log that typically consists of more than 100,000 events from a single system execution.

4. Related Work

In this section, we mostly review related work based on the visualization technique applied: UML sequence diagram, Gantt chart, stacked timeline, and 2D projection approach (Figure 2). For a general review of time-oriented data visualization, we refer to the survey of W. Aigner et al. [AMST11].

UML sequence diagram is a widely used technique in this scope [OMG07]. As Figure 2a shows, events are rendered in a multi-timeline layout, where events are distributed over different timelines based on a reasonable attribute, such as object ID or thread ID

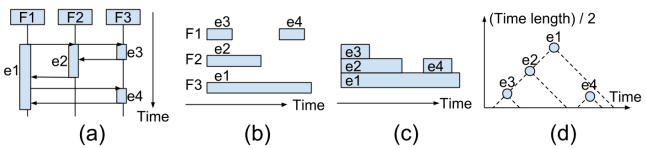

Figure 2: *Existing approaches: (a) UML sequence diagram, (b) Gantt chart, (c) stacked timeline, (d) 2D projection approach. $F_1 \sim F_3$ indicate different timelines, while $e_1 \sim e_4$ represent events.*

in software execution. Interactions between timelines are presented by arrows. Timelines are typically displayed vertically. Some previous research focuses on improving the efficiency and usability of UML sequence diagrams. W. De Pauw et al. introduced a more efficient layout that partially merges different timelines without involving temporal overlap [DPLVW98]. S. Xie et al. simplified the encoding of events within a single timeline [XKS*09]. Additionally, interactive techniques were developed to improve the space usage of UML sequence diagrams [SR05].

Gantt charts also employ the multi-timeline layout, but are rendered horizontally and typically without arrows [CPT52]; see Figure 2b. Among others, the shape, color, and thickness of the bars are used to encode properties of the events. For example, C. Plaisant et al. employed the thickness of the bars and additional icons [PSM98]. There are some variations of the layout. J.C. De Kergommeaux et al. added arrows back for important interactions only [DKdOS00]. S. Luz and M. Masoodian introduced an approach of rendering Gantt charts called temporal mosaic that displays concurrent events by allocating a fixed drawing area to time intervals and partitioning that interval by the number of parallel events in that interval [LM10]. J. Jo et al. introduced an interactive schedule for Gantt charts that aggregates similar nearby events to improve visual scalability [JHP*14]. Some authors also varied the type of data. N.W. Kim et al. introduced TimeNet that improved Gantt charts especially for genealogical data [KCH10]. L. Chittaro and C. Combi introduced approaches mainly focusing on relations between temporal intervals [CC03].

A disadvantage of a multi-timeline layout is the limited visual scalability, resulting from the inefficient usage of visual space. With increased data sizes, the number of timelines can increase sharply and the charts can easily get very large but sparse. In contrast, stacked timelines render events stacked one by one in a single timeline (Figure 2c). Stacked timelines with a more compact layout are used by P. André et al. [AWR*07] to visualize music history; see Figure 4b. However, the vertical dimension no longer shows the temporal order of the events. Sometimes, there are special relations between events, such as interplays between software threads visualized by B. Karran et al. [KTD13], and the software method calling relations visualized by J. Trümper et al. [TBD10].

All the techniques above project the time component of events in a one-dimensional space. However, some techniques project the time component to a two-dimensional space. For instance, Y. Qiang et al. introduced a triangular model to visualize interval events, see Figure 2d, where the y-axis represents the half event length and the x-axis represents time [QDV*12]. Start and end time of events are projected sideways to the time dimension by 45 and 135 degrees. Differently, J.F. Rit introduced an approach named SOPOs (Set Of Possible Occurrences), where the two dimensions correspond to the possible start and end time of events [Rit86]. 2D projection approaches usually tend to be less intuitive, due to the unfamiliarity with the chosen mapping.

Concluding, we consider stacked timelines more suitable for our case, because they have better visual scalability than UML sequence diagrams and Gantt Charts. Also, they render events more intuitively than 2D-projection approaches.

© 2018 The Author(s)
Eurographics Proceedings © 2018 The Eurographics Association.

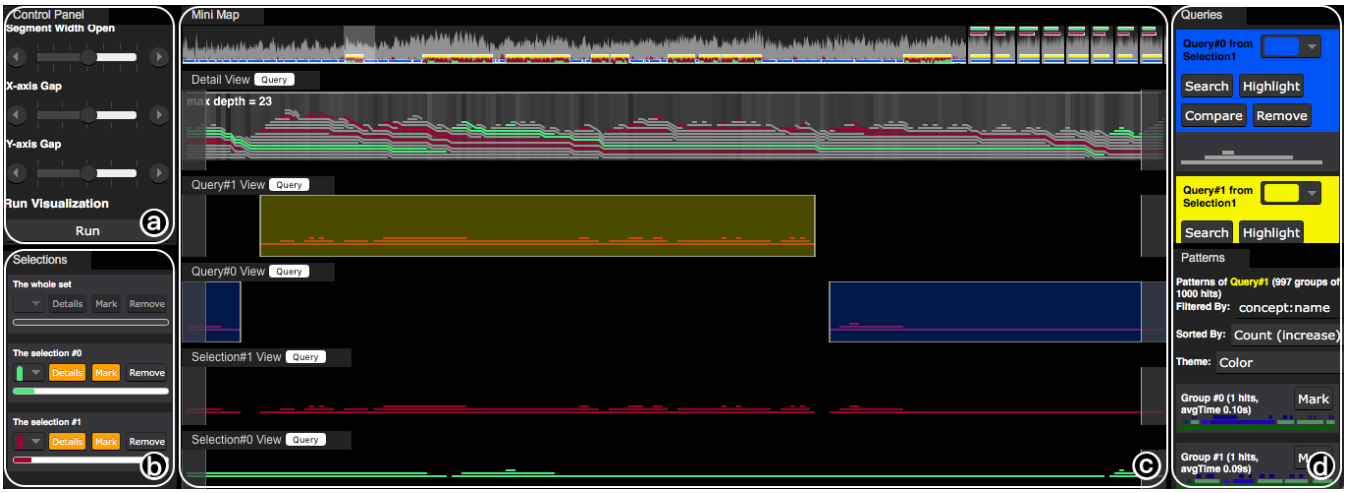

Figure 3: *SELE shows a running log of a multi-core searching algorithm containing 205,272 events. (a) The control panel contains basic settings for the visualization. (b) The selection view lets users generate selections from the whole data based on domain knowledge and tasks. (c) The main view displays events from the overview and detailed angles. (d) The pattern view shows visual pattern matches.*

5. Visual Design

In this section, we motivate the visual design of SELE. Figure 3 shows the graphical user interface of SELE. The main view contains a minimap, showing the context, and a number of detailed views showing different facets of events. The main view lays in the middle to attract the most attention, with the control panel, the selection view, and the pattern view surrounding it.

A core decision is to choose the visual technique employed in the main view (Figure 3c), which can significantly affect the performance of users. In Section 4, we conclude that stacked timeline can be a better choice than the other techniques. However, visual scalability issues still remain, particularly for visualizing partially parallel events, which are considered to be typical in our case. As Figure 4a and 4b show, visual space can be easily wasted in stacked timelines. Space formed by partially parallel events cannot be efficiently filled by events coming later. To overcome this limitation, we develop strata timeline as a visual technique for efficiently rendering partially parallel interval events.

Strata timelines are similar to stacked timelines in the sense that events are rendered as lines along the temporal dimension. The main difference is the way of rendering partially parallel events. Strata timelines result in a more compact layout than that of either stacked timelines or compact stacked timelines; compare Figure 4 a, b, and c. By design, the vertical order in strata timelines corresponds to the temporal order of events. We claim that this improves readability, especially for visualizing many parallel events. To achieve both compactness and maintain temporal order, strata timelines bend partially parallel events to nestle up to the fluctuation below, instead of being further rendered as a straight line. This approach is named strata timelines because of its visual similarity to strata, geological layers of sedimentary rock that are visually distinguishable from adjacent layers.

Additionally, we employ an accordion-like design to improve the scalability of the minimap. The minimap is divided into segments,

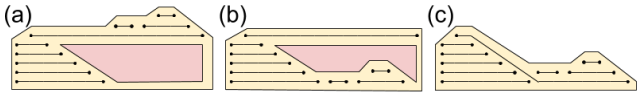

Figure 4: *Comparing stacked timelines with strata timelines in case of visualizing partially parallel events. Horizontal linked dot lines represent events. Yellow areas show the total space occupations and red areas show the wasted space: (a) a stacked timeline, (b) a compact stacked timeline, and (c) a strata timeline.*

where only one segment at a time can be expanded. Also in each of the segment, a stacked histogram shows distributions of selected events and patterns: see Figure 3c. As explained in Figure 5c, the minimap benefits from this "focus+context" design for showing a handier sliding window while keeping a considerable length of the map. The idea was inspired by the accordion drawing introduced by J. Slack [SHMJ04]. Also, R. Kincaid introduced SignalLens showing the efficiency of this accordion-like design [Kin10].

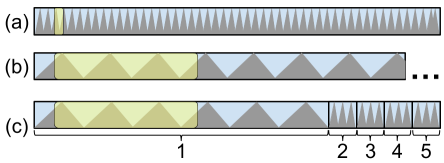

Figure 5: *Alternative designs of the minimap. Blue bars represent the map and the yellow bars represent the sliding windows. (a) The map fits the screen but leads to a too narrow sliding window and a distorted minimap. (b) The minimap is clearly recognizable but may be too long to fit the screen. (c) The accordion-like minimap, with 5 segments, allows reasonable width of both the map and the sliding window, while keeping the readability of the minimap.*

© 2018 The Author(s)
Eurographics Proceedings © 2018 The Eurographics Association.

Since our tool is designed to be domain-independent, the absence of domain knowledge could lead to difficulties in solving domain-specific problems. Therefore, we involve a selection view that enables users to select and individually visualize subsets of data. Those smart selections reflect users' domain knowledge. In this sense, our tool becomes domain-aware. Furthermore, a visual pattern matching function is provided. Patterns are visually selected by range selectors in the main view, while matches are highlighted in separate views. Currently, exact matching of temporal order of events is employed. Some more details are presented in Section 6.

6. Use Case

We conduct a use case for demonstrating the functionality of SELE and illustrating our user tasks. A video showing this use case is provided as supplementary material. This use case features 205,272 events from an actual log. The program mainly realizes a randomized multi-core depth-first search algorithm for parallel decomposition of graphs in strongly connected components (SCCs) [BLvdP16]. The user wants to improve the program performance by preventing unnecessary thread blocking. However, the user has little knowledge about the algorithm. Also, thread blocking is hardly observable within source code. In this case, the user tasks are abstracted as : (**Task 1**) basically understanding the program, and (**Task 2**) detecting potential thread blocking in a log file.

Task 1 After loading a log file, the main view is activated. Users can choose a segment on the minimap and explore the data by dragging the sliding window. Also, details are shown in a tooltip while hovering over an event in the detailed view. After right-clicking on the whole-dataset selection shown in the section view by default, a widget pops up to show basic information about the data. For example, in this use case, there are 9 methods involved in the data, where the most frequently called one is FIND; see the demo video and Figure 6. By using this widget, users can also create selections based on their domain knowledge. In this use case, events in threads 1 and 2 are selected and colored green and red, respectively. The execution of these two threads can be observed within the context of the whole dataset or observed individually (Figure 3c).

In this use case, the program functionality is abstracted from the log by visually detecting frequent event patterns. As shown in Figure 3, the blue and yellow boxes highlight two patterns that indicate

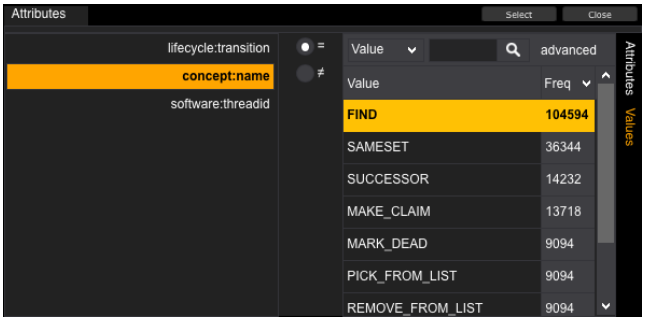

Figure 6: *A pop-up widget for showing basic data information and creating selections by users.*

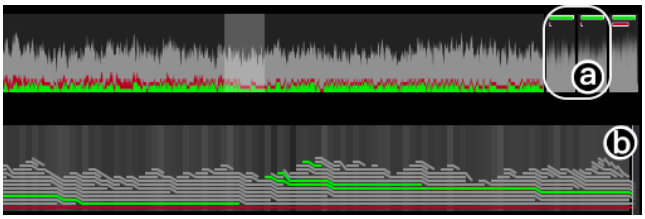

Figure 7: *Potential thread blocking indicated by the abnormal behavior of the red thread, where only one event of the red thread exists during a long time*

the forward-searching and backward-tracking of the algorithm, respectively. Meanwhile, a list of the pattern matches is shown for comparison of matches; see the bottom half of Figure 3d.

Task 2 With the stacked histogram on the minimap, the potential thread blocking can be easily observed. As Figure 7a shows, there are segments containing much fewer red events than others. As evidence, it can be seen in Figure 7b that many other threads (e.g. the green thread) work more actively. In contrast, there is only one red event running on the bottom. This abnormal behavior is considered an indicator of potential thread blocking.

7. Conclusion

In this paper, we address the challenge of visualizing parallel interval events. We summarize our contributions as follows:

- A domain-independent visualization tool named SELE for analyzing parallel interval events within a long trace of events. This tool aims at assisting domain experts to quickly understand systems and identify lower-level problems for solving higher-level concerns. We also demonstrate the usage of SELE with a use case involving a real dataset.
- A visual technique named strata timeline that overcomes the limited visual scalability of existing approaches.

For further research we consider the following steps.

- Evaluation of SELE and especially the strata timeline: Since SELE is designed to be domain-independent, it needs evaluation with users and use cases from different domains. Strata timelines need to be evaluated in a perception study.
- Improvement of the visual pattern matching to support patterns containing semantic information: Others have worked on this already. For instance, temporal relations between events have been realized in the temporal query searching function of EventFlow [MLDO*13]. However, in many cases, semantic relations between events are also important to detect and understand the artifacts in traces. For example, comprehension of programming structures, like co-recursion and self-reference, is challenging when using only source code. Visually detecting patterns of recursions on software execution logs might be a good alternative.

Acknowledgement

We thank Vincent Bloemen of the University of Twente for providing the data. This research was funded by the Dutch 4TU project "Big Software on the Run".

© 2018 The Author(s)
Eurographics Proceedings © 2018 The Eurographics Association.

References

[AMST11] AIGNER W., MIKSCH S., SCHUMANN H., TOMINSKI C.: *Visualization of time-oriented data*. Springer Science & Business Media, 2011. 2

[AWR*07] ANDRÉ P., WILSON M. L., RUSSELL A., SMITH D. A., OWENS A., ET AL.: Continuum: designing timelines for hierarchies, relationships and scale. In *Proceedings of the 20th annual ACM symposium on User interface software and technology* (2007), ACM, pp. 101–110. 2

[Bal99] BALL T.: The concept of dynamic analysis. In *Software Engineering - ESEC FSE* (1999), Springer, pp. 216–234. 1

[BLvdP16] BLOEMEN V., LAARMAN A., VAN DE POL J.: Multi-core on-the-fly SCC decomposition. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (2016), ACM, p. 8. 4

[BM13] BREHMER M., MUNZNER T.: A multi-level typology of abstract visualization tasks. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2376–2385. 1

[CC03] CHITTARO L., COMBI C.: Visualizing queries on databases of temporal histories: new metaphors and their evaluation. *Data & Knowledge Engineering 44*, 2 (2003), 239–264. 2

[CPT52] CLARK W., POLAKOV W. N., TRABOLD F. W.: *The Gantt Chart*. London, 1952. 2

[DKdOS00] DE KERGOMMEAUX J. C., DE OLIVEIRA STEIN B.: Pajé: an extensible environment for visualizing multi-threaded programs executions. In *European Conference on Parallel Processing* (2000), Springer, pp. 133–140. 2

[DPLVW98] DE PAUW W., LORENZ D. H., VLISSIDES J. M., WEGMAN M. N.: Execution patterns in object-oriented visualization. In *COOTS* (1998), vol. 98, pp. 16–16. 2

[Gün09] GÜNTHER C.: Extensible event stream xes standard definition. 2

[JHP*14] JO J., HUH J., PARK J., KIM B., SEO J.: LiveGantt: Interactively visualizing a large manufacturing schedule. *IEEE transactions on visualization and computer graphics 20*, 12 (2014), 2329–2338. 2

[KCH10] KIM N. W., CARD S. K., HEER J.: Tracing genealogical data with timenets. In *Proceedings of the International Conference on Advanced Visual Interfaces* (2010), ACM, pp. 241–248. 2

[Kin10] KINCAID R.: SignalLens: Focus+context applied to electronic time series. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 900–907. 3

[KTD13] KARRAN B., TRUMPER J., DOLLNER J.: Synctrace: Visual thread-interplay analysis. In *Software Visualization (VISSOFT), 2013 First IEEE Working Conference on* (2013), IEEE, pp. 1–10. 2

[LM10] LUZ S., MASOODIAN M.: Improving focus and context awareness in interactive visualization of time lines. In *Proceedings of the 24th BCS Interaction Specialist Group Conference* (2010), British Computer Society, pp. 72–80. 2

[MLDO*13] MONROE M., LAN R., DEL OLMO J. M., SHNEIDERMAN B., PLAISANT C., MILLSTEIN J.: The intervals and absences of temporal query. In *Proceedings of the 2013 Annual Conference Human Factors in Computing Systems* (2013). 4

[MLL*13] MONROE M., LAN R., LEE H., PLAISANT C., SHNEIDERMAN B.: Temporal event sequence simplification. *IEEE transactions on visualization and computer graphics 19*, 12 (2013), 2227–2236. 2

[OMG07] OMG O.: Unified modeling language (OMG UML). *Superstructure* (2007). 2

[PSM98] PLAISANT C., SHNEIDERMAN B., MUSHLIN R.: An information architecture to support the visualization of personal histories. *Information Processing & Management 34*, 5 (1998), 581–597. 2

[QDV*12] QIANG Y., DELAFONTAINE M., VERSICHELE M., DE MAEYER P., VAN DE WEGHE N.: Interactive analysis of time intervals in a two-dimensional space. *Information Visualization 11*, 4 (2012), 255–272. 2

[Rit86] RIT J.-F.: Propagating temporal constraints for scheduling. In *AAAI* (1986), vol. 86, pp. 383–388. 2

[SHMJ04] SLACK J., HILDEBRAND K., MUNZNER T., JOHN K. S.: Sequencejuxtaposer: Fluid navigation for large-scale sequence comparison in context. In *German conference on bioinformatics* (2004), vol. 53. 3

[SR05] SHARP R., ROUNTEV A.: Interactive exploration of UML sequence diagrams. In *Visualizing Software for Understanding and Analysis, 2005. VISSOFT 2005. 3rd IEEE International Workshop on* (2005), IEEE, pp. 1–6. 2

[TBD10] TRÜMPER J., BOHNET J., DÖLLNER J.: Understanding complex multithreaded software systems by using trace visualization. In *Proceedings of the 5th international symposium on Software visualization* (2010), ACM, pp. 133–142. 2

[XKS*09] XIE S., KRAEMER E., STIREWALT R. K., DILLON L. K., FLEMING S. D.: Design and evaluation of extensions to UML sequence diagrams for modeling multithreaded interactions. *Information Visualization 8*, 2 (2009), 120–136. 2

© 2018 The Author(s)
Eurographics Proceedings © 2018 The Eurographics Association.