Visualisation Research Centre
University of Stuttgart
Universitätsstraße 38
D-70569 Stuttgart

Master Thesis

Visual Analytics of Big Data from Distributed Systems

André Kutzleb

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Daniel Weiskopf
Supervisor:	Dr. rer. nat. Michael Burch

Commenced:	April 5th, 2017
Completed:	October 5th, 2017
CR-Classification:	C.2.4, D.2.6, G.3, H.2.1, H.3.4

Abstract

Distributed Systems are challenging to debug because the temporal order of events and distributed states are hard to track. The high complexity of distributed systems make fully automatic reasoning difficult to apply. Domain experts are often required to reason about the behavior of a system based on log files from various sources. This situation presents a good opportunity for visual analytics. Data from multiple sources can be preprocessed and visualized, and then domain experts can conduct exploratory analysis to accelerate the identification of issues. The goal of this master thesis was to create such a visual analytics tool to help domain experts explore data collected from distributed systems more efficiently and assist in identifying bugs and anomalies. The system was used by domain experts and helped to identify issues in a distributed system, showing that visual analytics can be a useful tool to assist domain experts in their daily work.

Kurzfassung

Fehlersuche in verteilten Systemen ist eine Herausforderung, da es schwierig ist, die zeitliche Ordnung von Ereignissen sowie verteilte Zustände im Auge zu behalten. Die hohe Komplexität von verteilten Systemen macht es schwierig, vollautomatisch Schlussfolgerungen zu ziehen. Domänenexperten müssen oft Rückschlüsse über ein komplexes, verteiltes System auf Grundlage von Logdateien aus verschiedenen Quellen ziehen. Diese Situation bietet eine gute Möglichkeit, Visual Analytics anzuwenden. Daten aus diversen Quellen können vorverarbeitet und visualisiert werden, woraufhin Domänenexperten explorative Analyse zur Beschleunigung der Fehlersuche betreiben können. Das Ziel dieser Masterarbeit war es, solch ein Visual Analytics-Werkzeug zu erschaffen, um Domänenexperten das Erkunden von Daten von verteilten Systemen zu erleichtern und bei der Identifizierung von Fehlern und Anomalien zu helfen. Das System wurde von Domänenexperten verwendet und half bei der Identifizierung von Problemen in einem verteilten System, was zeigt, dass Visual Analytics ein nützliches Werkzeug ist, um Domänenexperten bei ihrer täglichen Arbeit zu unterstützen.

Contents

List of Figures

List of Listings

1 Introduction

Identifying inconsistent state and behavior in a distributed system is a challenging task. Distributed systems engineers are faced with an abundance of information collected from different nodes of the system. This information, typically collected in the form of log files, can be very abundant. Filtering a single log file for relevant information can be accomplished by simple tools, but even in a non-distributed system, relevant information may be spread out over multiple log files.

The complexity increases even more when a distributed system is involved: analyzing the information collected by multiple nodes and extracting a logical sequence of events manually becomes a tedious and error prone task. Log files, while intended to be readable by humans, are also a very overwhelming format: They intend to capture all information relevant to find problems after they occurred. A majority of this information may be irrelevant for a particular problem, or not initially necessary to get an overview of key events.

Components in a distributed system may also regularly collect additional information during its runtime, e.g. metrics about CPU, Memory and disk space usage, or any other domain specific information. Correlating these metrics with information found in log files may further help the understanding of abnormal behavior, a task which becomes increasingly difficult in a text based environment.

Furthermore, a rare sequence of events may lead to unintended behavior under specific circumstances. Reproducing this sequence of events may be very hard or impractical to do: It is paramount to extract as much information as possible with the data already at hand.

The goal of this thesis is to research and develop a visual analytics system to process and display data collected in this distributed environment, thereby assisting domain experts with exploring the abundance of data in a more intuitive manner.

Structure

The remaining part of this thesis is structured as follows:

Chapter 2 – Related Work: Literature relevant to this thesis is described in this chapter.

Chapter 3 – Data Model and Processing: This chapter goes into detail about the data model, associated challenges, and steps necessary to transform the data so it can be explored in the visual analytics system.

Chapter 4 – Visual Analytics Techniques: In this chapter we explain what visual analytics techniques are used for this thesis, and how the domain specific data can be explored.

Chapter 5 – Case Studies: This chapter shows case studies of problems solved by domain experts with the help of the visual analytics system presented in this thesis.

Chapter 6 – Discussion and Limitations: In this chapter we discuss the advantages and limitations of the proposed visual analytics system.

Chapter 7 – Conclusion and Future Work: In this chapter we come to a conclusion about the visual analytics system and point out some possible research topics for future work.

2 Related Work

The theoretical foundation for the visual analytics system proposed in this thesis is *Visual Analytics: Definition, Process, and Challenges* [KMS+08] by Keim et al., who describe an approach to visualize and interact with data where an expert user can explore and gain insights from the data he or she explores. Keim et al. point out that there is no single solution that would fit the requirements of every domain. As such, as part of this work, a visual analytics system will be proposed which specifically caters to the needs of distributed systems experts. Keim et al. also go into more details about the challenges and possible solutions for various questions surrounding visual analytics in their book *Mastering the Information Age: Solving Problems with Visual Analytics* [KKE10], and *A survey of visual analytics techniques and applications: State-of-the-Art Research and Future Challenges* [SWLL13] by Sun et al. goes further into detail about the challenges with visual analytics systems.

Since a sizable amount of the data encountered in the context of distributed systems are time-series, the book *Visualization of time-oriented data* [AMST11] by Aigner et al. provided useful insight into the challenges of visualizing and interacting with data of such nature. Silvestro et al. provide further insight into the challenges related to visualizing and interacting with ever-increasing amounts of time-dependent data in their publication *Visual Analysis of Time-Dependent Multivariate Data from Dairy Farming Industry* [SBC+14].

The question on how to deal with *Big Data* is tackled in *Big data analytics: a survey* [TLCV15], a survey by Tsai et al., which provides valuable insight for the decisions made in this thesis regarding the processing, storing and visualization of large amounts of data. Similarly, *Visualization and visual analysis of multifaceted scientific data: A survey* [KH13] by Kehrer et al. resembled a useful guidebook on how to deal with specific problems related to the treatment of inhomogeneous data. *Coping with volume and variety in temporal event sequences: Strategies for sharpening analytic focus* [DSP+17] by Du et al. also provided further insight into strategies for dealing with large amounts of data points.

The article *Human–Computer Interaction and Knowledge Discovery (HCI-KDD): What Is the Benefit of Bringing Those Two Fields to Work Together?* [Hol13] by Holzinger et al. provides insight into a similar visual analytics approach, yet in another domain, and was

used as a reference to prevent pitfalls encountered by the authors which could similarly appear in the visual analytics for distributed systems domain.

The publication *The Role of Uncertainty, Awareness, and Trust in Visual Analytics* [SSK+16] by Sacha et al. served as a guidance on how to deal with data uncertainty. This uncertainty can both be inherent to the data, as well as be introduced by the visual analytics system due to data transformations, aggregations, sampling or limitations in the ability to visualize large data sets precisely and expert users need to be made aware of these issues.

To get an overview over state-of-the-art systems tackling visual analytics in the industry, we consulted *Visual analytics for the big data era: A comparative review of state-of-the-art commercial systems* [ZSB+12] by Zhang et al. To investigate the possibility of incorporating UI learning learning behavior in the proposed visual analytics system, the research conducted in *HindSight: Encouraging Exploration through Direct Encoding of Personal Interaction History* [FDPH16] by Feng et al. has been studied.

Pip: Detecting the Unexpected in Distributed Systems [RKW+06] by Reynolds et al. describes a system to analyze behavior in distributed systems and provided a good overview of common issues encountered in distributed systems.

The mentioned related work covers many of the individual aspects of this work, and the insights gained will be used to build the visual analytics system for Big Data in distributed systems.

3 Data Model and Processing

In this chapter we will go into detail about the data model for the visual analytics system, as well as all challenges related to transforming the data into formats that are beneficial for efficient exploration. We will also discuss the nature of the data, and how its properties can introduce uncertainty, which has to be taken into account to provide expert users with a useful visual analytics tool.

3.1 Distributed Systems Environment

To understand the nature of the data the visual analytics system will display, we have to look at how and where it is collected. To refer to a distributed system with one or more components we shall use the term *cluster* (see Figure 3.1). A cluster consists of

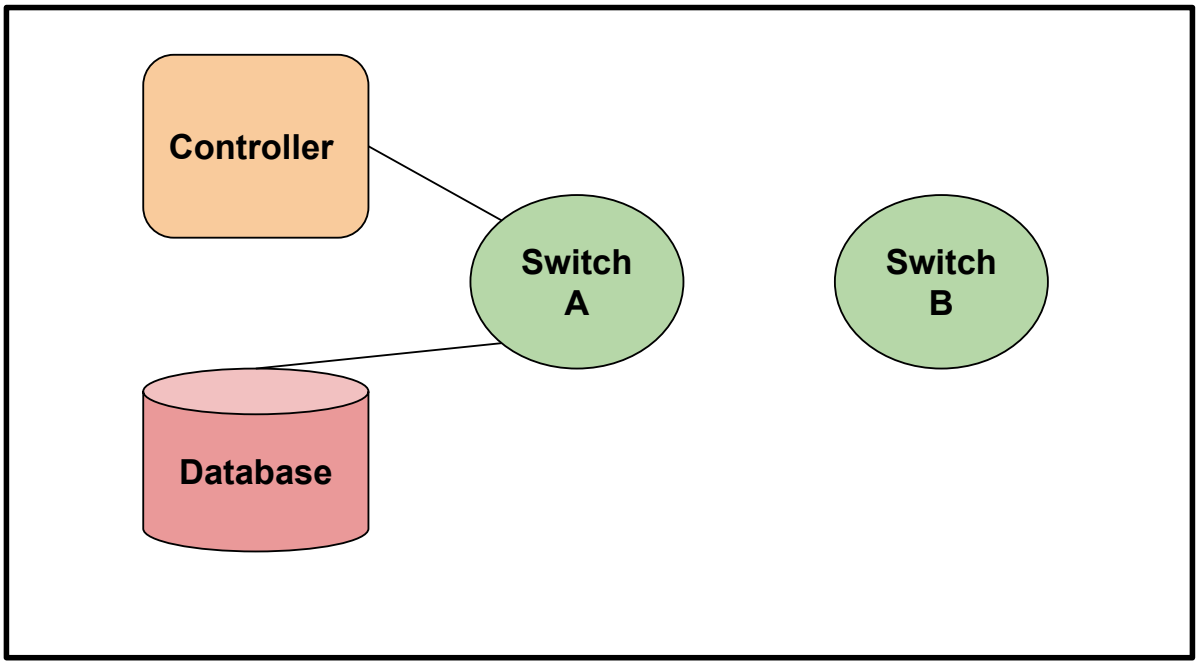

Figure 3.1: A simple cluster with different node types

nodes. A cluster may have an arbitrary amount of nodes, but has at least one. Nodes are some form of computer system (e.g. a server, a networking device like a switch, a laptop), however the exact nature of the nodes are not important, as we are only concerned about the data they produce. Nodes in a cluster can generally communicate with each other (using various protocols). A cluster and its nodes can thus be viewed as an undirected graph. Either by design or due to erroneous behavior, some nodes may not be able to communicate with other nodes, so two or more disjunctive graphs may be present in a cluster.

3.1.1 Cluster Life Cycle

A cluster can exist for different periods of time. Especially with the prevalence of virtualization and container technologies, a cluster can be created and destroyed in seconds (e.g. for a unit test for a distributed system) while other clusters may persist for years (server center). Clusters can be completely static in their set-up, or change over time (due to reconfiguration, equipment failure, or bugs). The data collected by nodes in these clusters may or may not be tied to the cluster life cycle, depending on how the resources for the nodes in the clusters are allocated.

3.1.2 Data Life Cycle

Nodes in clusters usually (but don't have to) collect various data. The type of data collected, as well as its volume, may differ significantly in between different nodes as well as over time. An example for a change in volume would be a node in a cluster with the purpose of forwarding messages: the node may see an increase in traffic depending on the time of day, which can result in a significant increase in messages logged. An example for a change of the type of data collected would be a change of any of the nodes: a specific node might be upgraded, which causes different data to be collected. Alternatively, a node might be configured so that it starts to log even minor events, e.g. in an attempt to gain more information for finding a bug.

As storage capacity is not unlimited, there are also constraints regarding the data collected by nodes. A node might automatically delete the oldest data points it retained. It might also selectively keep the data of certain time windows, or selectively delete specific types of data, e.g. because its volume is too high to store indefinitely.

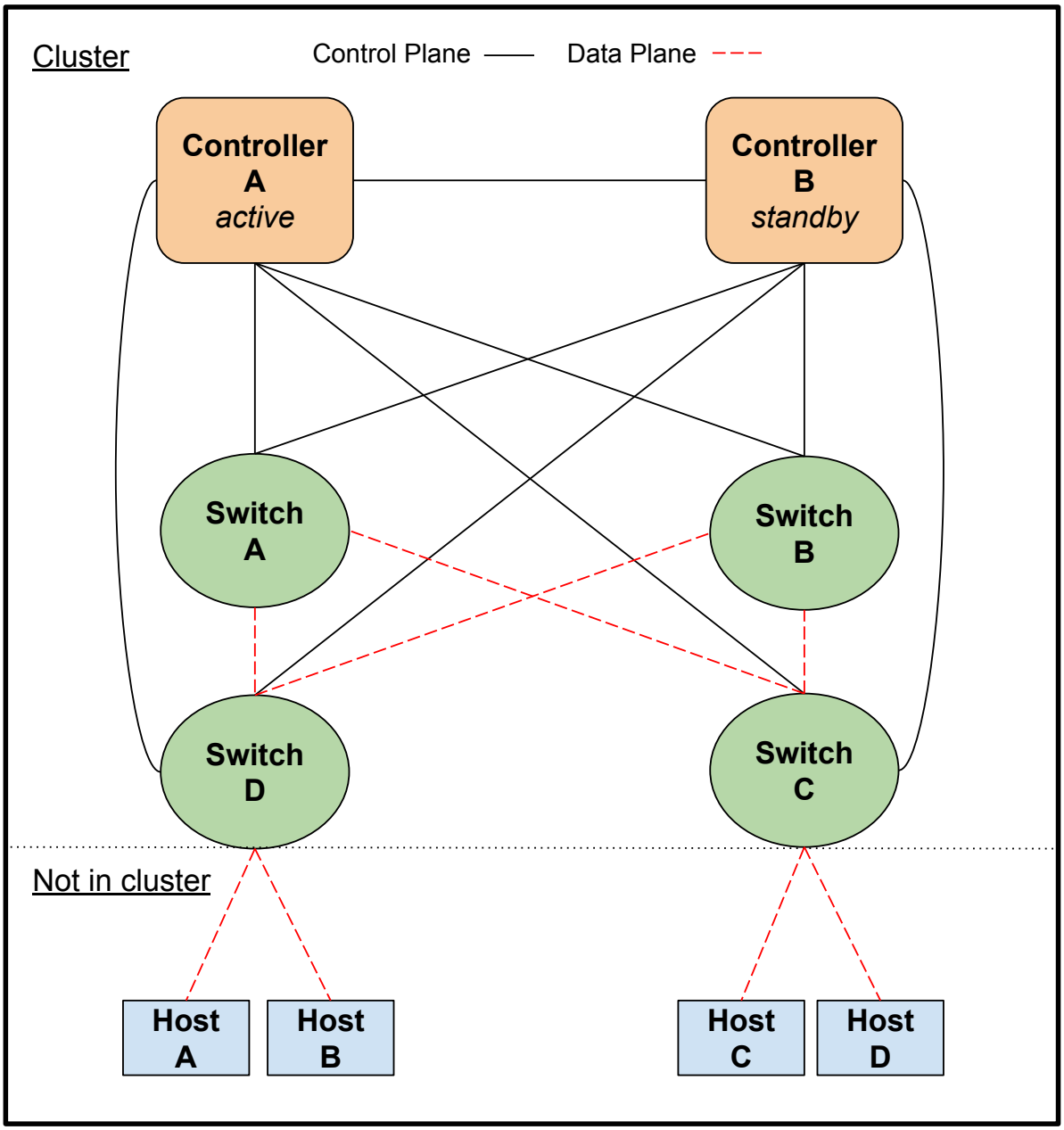

Figure 3.2: A cluster in a typical network environment

3.2 Cluster Composition and Properties

In this section, we will take a look at the actual cluster types and the data collected by its nodes used for visual analytics in the context of this thesis.

Clusters represent networks designed for the forwarding of messages between computers outside of the cluster. Figure 3.2 depicts a typical cluster as encountered for the visual analytics system. The cluster is logically split up into two parts, dictating which node can communicate with which other node:

1. **Data Plane**

   Nodes taking part in this part are usually switches. They are connected to each other directly or indirectly (usually forming a graph with full reachability). They are also connected to nodes outside of the cluster. These outside nodes are communicating with each other through the switches.

2. **Control Plane**

   The control plane consists of one or more controller nodes. These controllers are connected to each other, as well as to all switches. This means that the switches represent the border between the data plane and the control plane (using separate interfaces on the switches). Controllers themselves are servers which configure the switches to forward messages for the nodes outside the clusters. Generally, one controller is *active*, while all other controllers (if present) are on *standby*. The standby controllers are constantly kept up to date on the state that is managed by the active controller, and can take over at any time.

3.2.1 Support Bundles

Most relevant for our purposes are the controllers. The controller nodes are collecting data themselves, but they also collect data from the switches as well as other controllers. Controllers don't collect data from the switches and other controllers all the time.

The controllers will start a collection process when they are either triggered by an event or triggered manually by a user. The collection process aims to capture as much information as is possible, in order to provide the best possible snapshot of the current state of the cluster, as well as as much historic data as possible to understand how the cluster ended up in the specific state. The result of this collection process is a so-called *support bundle*.

Support bundles are big, compressed archives with a vast array of the inhomogeneous data that has been collected from the controllers, as well as all of the switches. Figure 3.3 shows how data in a support bundle is organized. One of the controllers is the initiator of this process, and all other controllers will attempt to provide their data to this controller. More specifically, each controller independently collects its own information, whereas the controller initiating the creation of a support bundle will also collect data

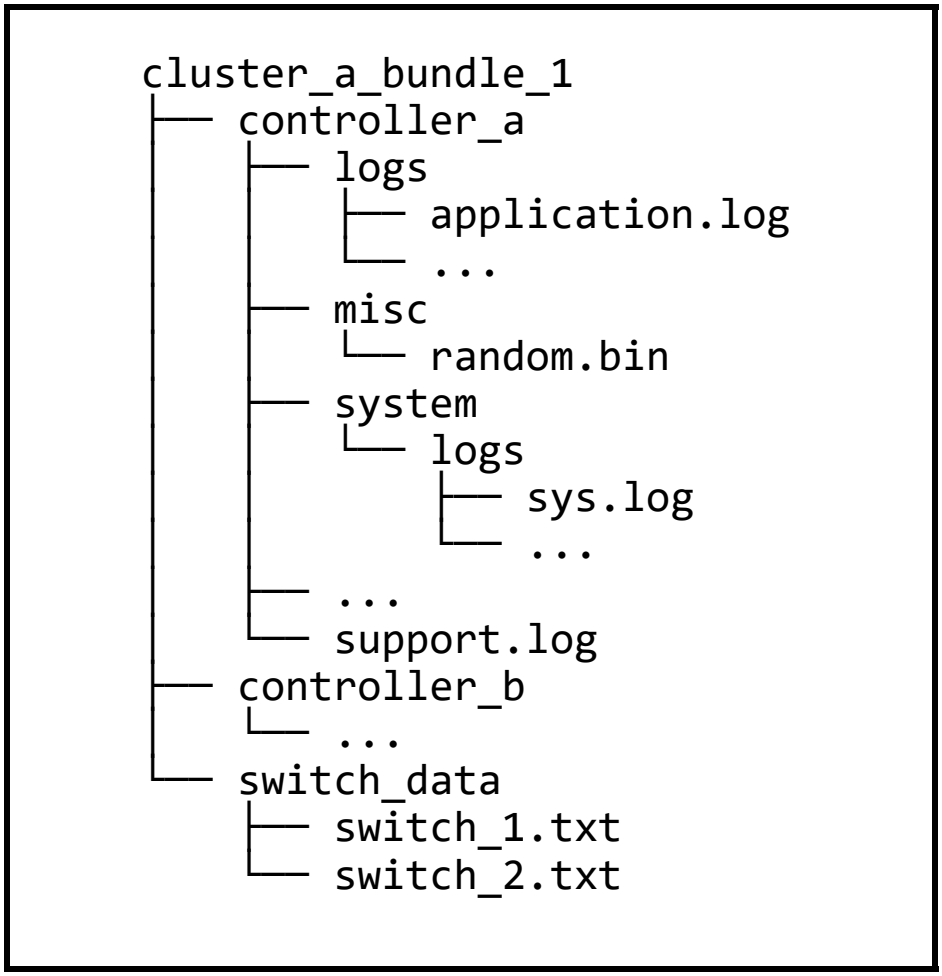

Figure 3.3: Data hierarchy in a support bundle

directly from the switches. All of these data collections are then combined into a single support bundle.

We need to take a look at issues that can occur when visualizing support bundle data to avoid problems during exploration and visualization of it.

3.2.2 Intra-Bundle Problems

The previous section went into some detail about the support bundle collection process. In a perfect world, a controller could take an instantaneous snapshot of every single node of the cluster as well as all switches, without any connection problems, delays or data corruption.

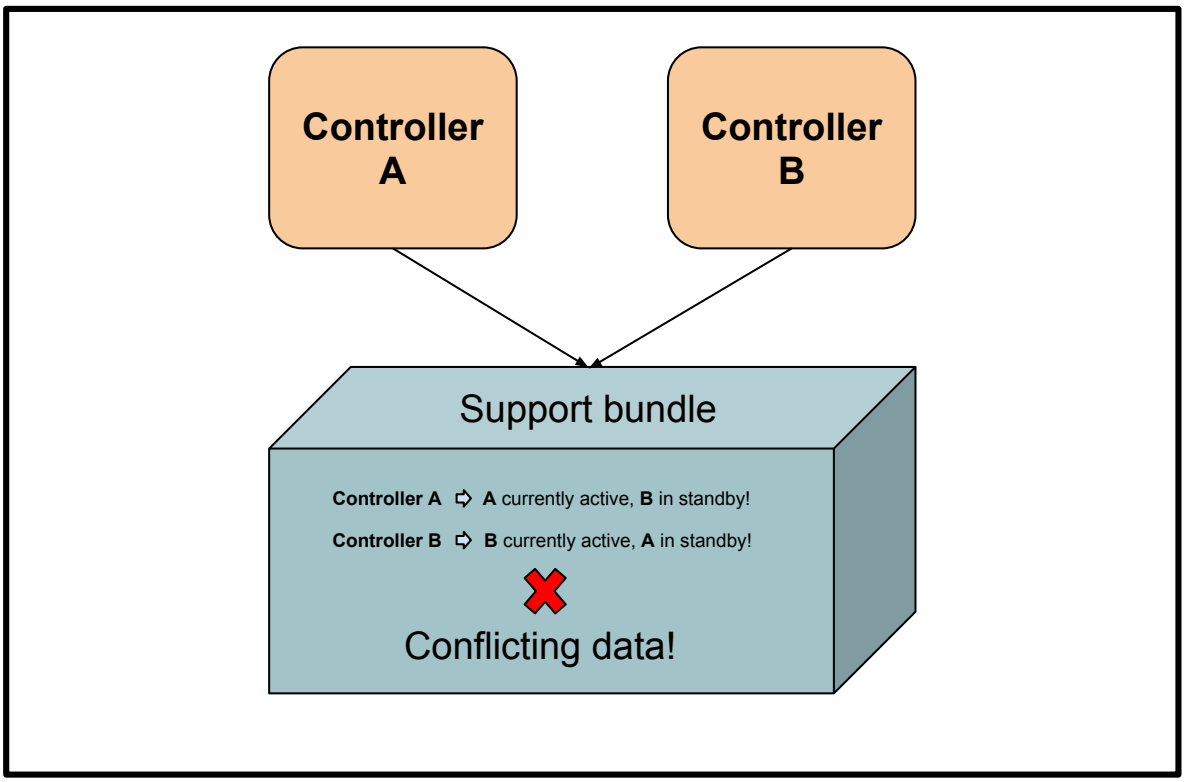

Figure 3.4: Conflicting view on state in a cluster

In practice, however, the collection process is not done instantly, and the collection of data by all nodes does not stop while the collection process is running.

The following problems can be observed in regards to the creation of support bundles, regarding the support bundle itself:

1. **Data inconsistency**

   Every single controller collects various types of data (see Section 3.3 for an example). Some of that data pertains to the controller itself and is thus free of overlaps, but some of the data collected can be redundant between controllers. An example for this would be information about the state of switches, from the perspective of the controllers (data collected directly from the switches is a separate source for inconsistency). Assuming that the collection of data by the controllers is working as intended, a change in the state of a switch might be represented in the information obtained by one of the controllers, but not by another. In a more extreme example, the data collected about the switches may differ vastly from the viewpoint of each controller and what the switches report themselves. If inconsistencies are detected, they have to be brought to the attention of the

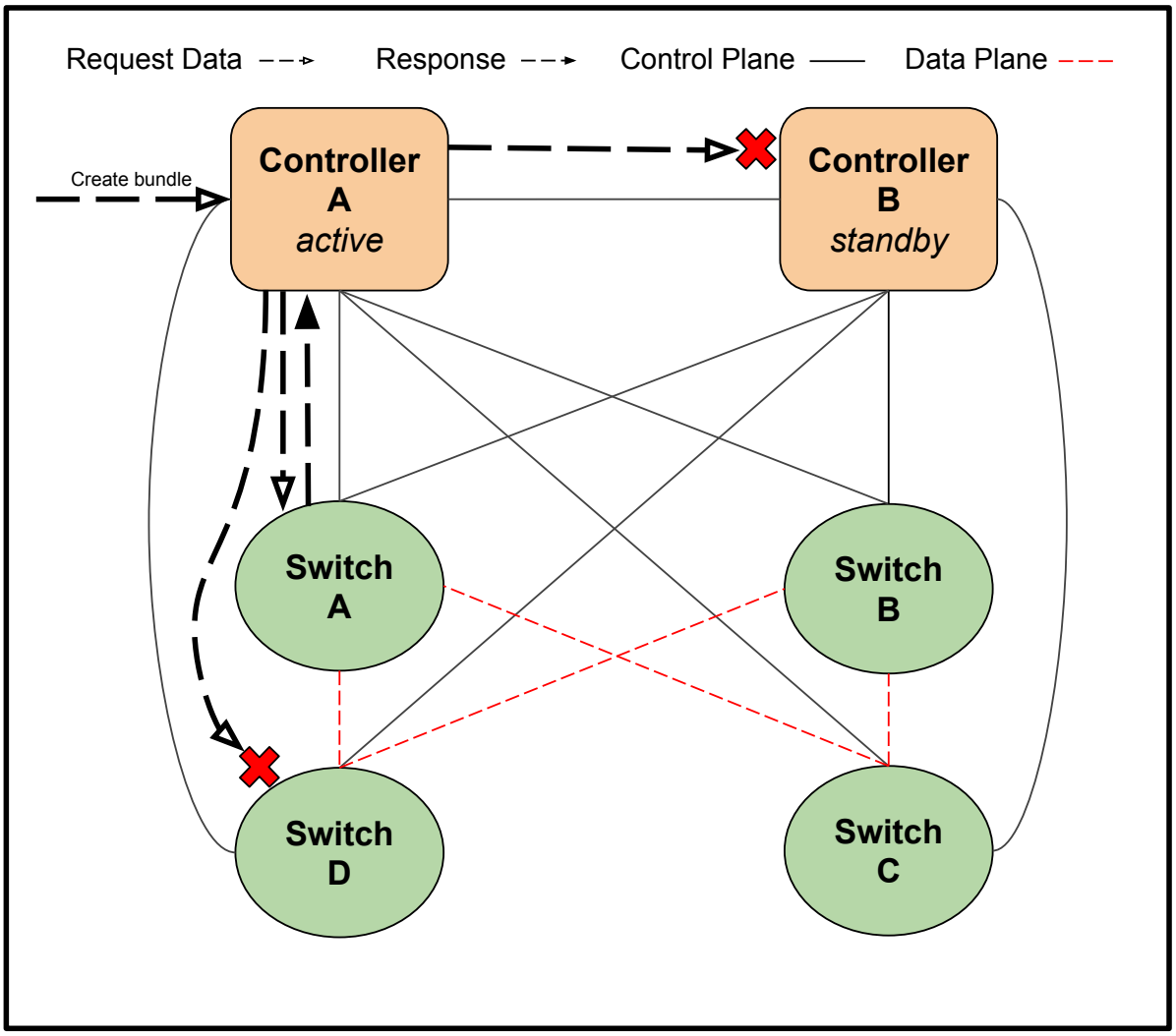

Figure 3.5: Unreachable nodes during support bundle creation

user of the visual analytics system. Figure 3.4 shows the conflict of a certain data point, resulting in an illegal invariant of the system (Although this invariant did not necessarily occur, and could be a side effect of the non-instant support bundle creation process). In cases like these, the differences should be highlighted, and the option to look at either data for the purpose of analysis should be present.

2. **Switches unreachable**

   If a switch cannot be communicated with during the collection of a support bundle, then its data cannot be included in the support bundle. Figure 3.5 shows a cluster, where during the creation of a support bundle the switch is not reachable. Since the switch is a node of the cluster, and thus of interest in the analysis, this means

that potential vital information for problem solving and visualization purposes can be missing. The user of the visual analytics system has to be made aware of the lack of information on this specific node.

3. **One or more controllers unreachable**

   Any controller, even the one initiating the creation of the support bundle, may not be able to relay its information to the process that creates the support bundle. Figure 3.5 shows a controller that can't be reached during the creation of a support bundle. The part of the controller that would be queried for data may not be responsive anymore, or the network connection to another controller might be broken. In any case, vital data about the state of one or more controllers might not be included in the resulting support bundle. Again, it should be made very clear to the user of the visual analytics system that a big set of expected data is not available for visual analysis.

4. **Collection influencing cluster**

   The creation of a support bundle is no quick task. A lot of information is collected from the file system of the respective node, which causes a lot of I/O activity for the employed storage, which may negatively influence the performance of the respective node in the cluster. Requesting data from the switches as well as from other controllers increases network utilization, which may interfere with other workloads in the network. The polling of controller data may stall it, or, in the worst case, cause it to fail completely. The creation of the support bundle might also slow down the system, e.g. by competing for the same resources (CPU, memory, storage space, network) as the main functionality of the respective node. If storage space is severely limited, the collection might also starve the node of storage space needed for function. Even if none of the more severe problems are observed, the act of data collection may influence the data itself, e.g. data about disk usage, memory, or CPU usage. As such, the moment of the collection might appear anomalous when investigated using the visual analytics system, and should be highlighted to let users know that any unexplained change in the data for that specific moment in time might be related to the act of collection.

5. **Corrupted data collected**

   In terms of its usefulness for visual analytics, the collected data in general can be partially or completely corrupted. In case of partial corruption, parts of the data may not be interpretable, or in a data type with a time dimension, the data points for one or more points in time are unreadable. Similarly, a log file with a well defined structure might have an unexpected, garbled line due to the crash of a node. This has to be taken into account when data is digested, as simply ignoring data because of a single problem might get rid of millions completely

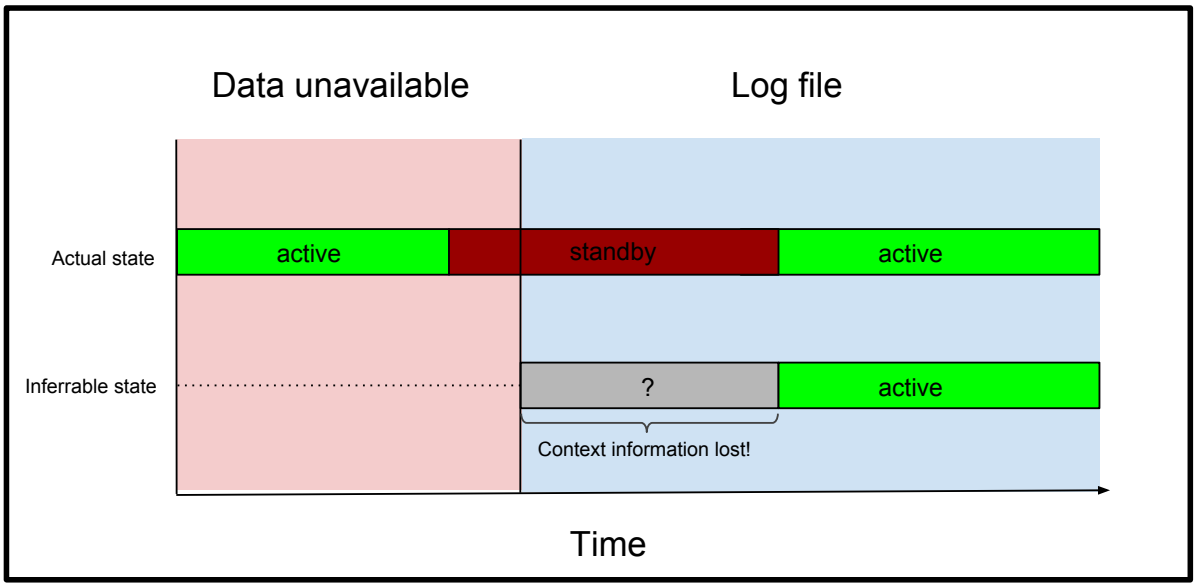

Figure 3.6: Loss of state information when data is truncated

intact data points. Ideally, the user should be made aware of corruption in data, even if the effect of the corruption can't be automatically inferred. The presence of data corruption might in itself be a hint for problems which might be discoverable through visual analytics and an expert user. In case of a total corruption of some data (e.g. if the data is in a specific format that is only interpretable as a whole), the user should also be made aware that certain data is not available due to corruption.

6. **State data not sufficient**

   Some data types carry causal state information, meaning that the current state is inferred from the chain of events that happened before. An example for this would be a controller, which can be in active or in standby state. The state might be inferred from a log, but if the log file does not extend all the way to the start of the controller anymore (or is corrupted), it may not be possible to immediately infer the state of the cluster from the remaining data. Figure 3.6 depicts this issue. It might be possible that for certain stateful data a certain data point might re-establish a clear state (similar to a key frame in a video codec), but this might not be true for all data. In an extreme example, every single data point for certain data is relevant to its state. Whenever state cannot be inferred due to lacking data, the user should be made aware of the time frame of uncertainty. Due to expert knowledge, the user of the visual analytics system may be able to refine the visualization to infer states from incomplete data.

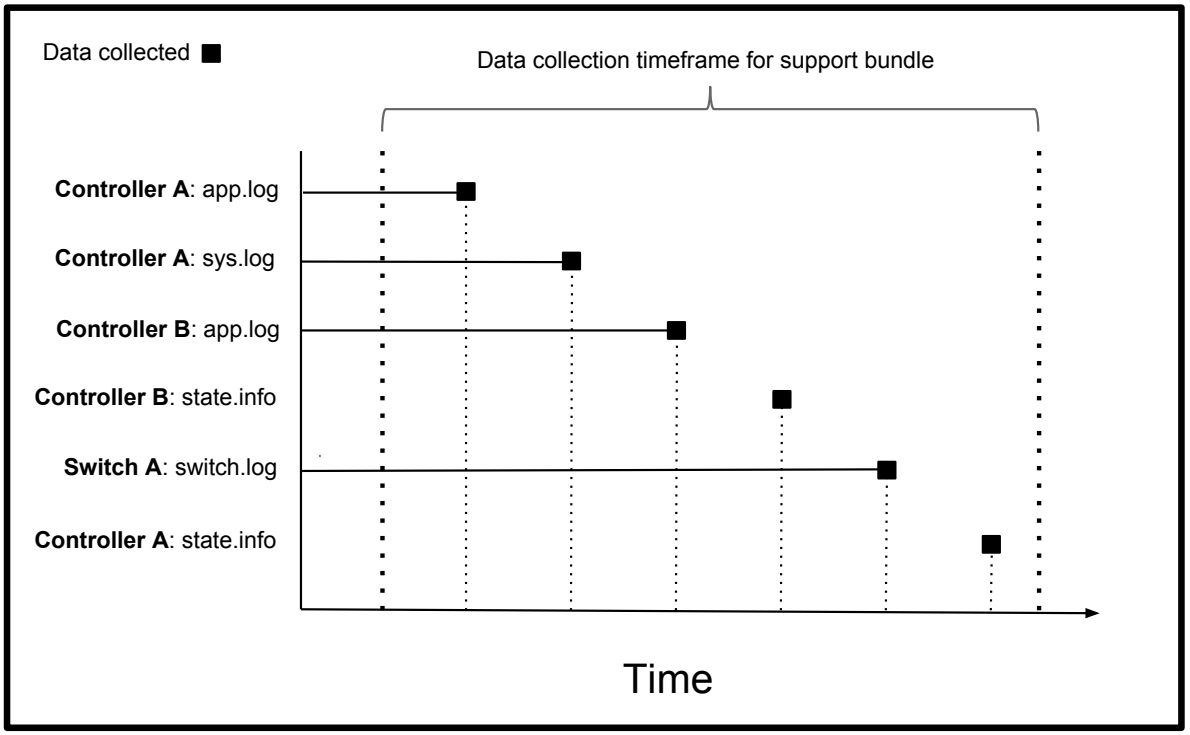

Figure 3.7: Time window of data collection

7. **Change in data types or format**

   Over the course of a clusters life, certain data might stop to be collected by nodes (e.g. updates disable the collection of certain data). This situation is relatively easy to deal with, as there is simply nothing to display. Similarly easy to deal with is the occurrence of a new data type, that has not been collected before - the visual analytics system simply has to display it. More complicated are changes in the structure of collected data. Heavily dependent on the particular case, changes like this may or may not be easy to pick up automatically. If possible, expert knowledge about the data might be necessary to identify the changes, and the expert knowledge should be employed retroactively to help the visual analytics system detect these changes in the future.

8. **Creation not instantaneous**

   As the creation of a support bundle may take several minutes (all while the cluster and its nodes keep performing their tasks), the data collected within a support bundle may stretch across different time frames solely for this reason. This also means that data which is representative of a single point in time is collected at different moments in time during the creation of the support bundle. Figure 3.7

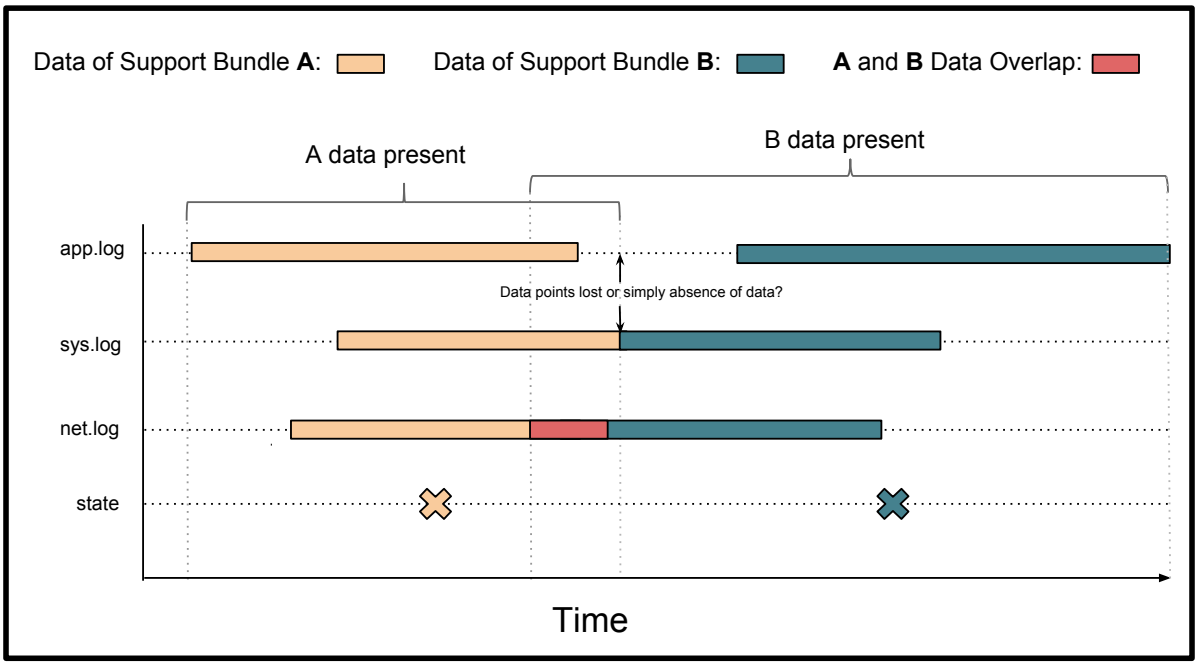

Figure 3.8: Potential data loss and overlapping of data points

depicts this issue. The creation time frame should be made aware to the user of the visual analytics system.

3.2.3 Inter-Bundle Problems

Beside the afore-mentioned problems and challenges that arise just within a singular support bundle, there are a new set of problems appearing when looking at two or more different support bundles that pertain to the same cluster.

The reason to look beyond individual support bundles is that support bundles are not the focus of any analysis: The real focus lies on the cluster, for which it collected data. Therefore, it is easier to think of the visual analysis in terms of clusters, and support bundles are just a means to an end to collect information about a cluster during its existence.

When looking at support bundles this way, many problems arise:

1. **Weakly defined and varying borders**

   Support bundles don't have a well-defined time window for which they contain data (even ignoring the fact that not all collected data has a time dimension, so some of the data only defines a single point in time). The window of a support

bundle could be loosely defined in many different ways, although either way does not change the fact that not all data contained within the bundle will actually be adhering to this time frame. Figure 3.8 shows the problem with unclear time frames for data. Even if two support bundles were seemingly perfectly consecutive, It would require extensive knowledge about all data to conclude with certainty that no information is missing. Without extensive knowledge about the collected data, only overlap of data points would allow the conclusion that no data has been lost. Otherwise, any gap, no matter how small, might have contained another data point that was lost. The user of the visual analytics system should be made aware of the individual time frame supplied by each bundle, as well as highlight sections of uncertainty where data points are either missing, or not occurring.

2. **Overlapping**

   Overlapping data is another problem in itself. Overlapping of data means that for data with a time dimension, two or more support bundles have data for the same time frame. In many cases, this overlap can be resolved by identifying duplicate data points and just eliminating the duplicates. For some data, however, this is not straightforward. If there are discrepancies where the overlap occurs, it is unclear which data should take precedence. This could occur because of data corruption, or a change in how data is being recorded. Overlap has to be determined on a data type basis, as for some data types overlap might be present, and for others not, depending on how big the present time frame for the respective data is.

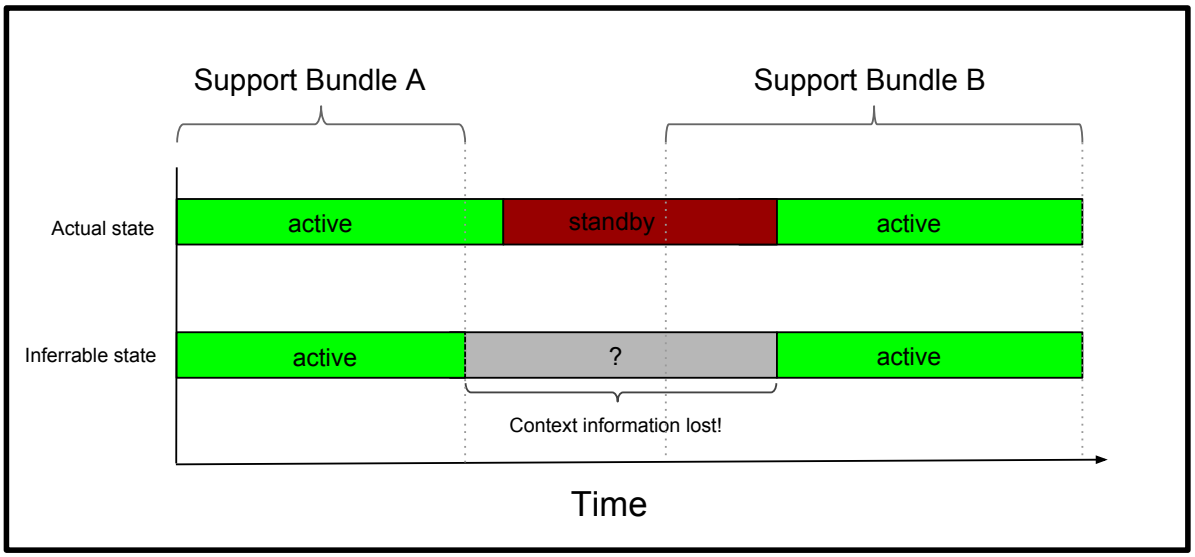

Figure 3.9: State information lost between support bundles

3. **Stateful data problems**

   Similar to the issues with state mentioned for Intra-Bundle problems, the absence of data points for stateful data can be problematic. If two or more support bundles are present with missing data points for certain data, then it might not be trivial to find out what the state is for the data present in the chronologically latter bundle. Figure 3.9 highlights this problem. A state transition might have happened in the data gap, which can cause state uncertainty well into, or covering all the data points in the latter support bundle. Uncertainty should be displayed to the user of the visual analytics system for that specific data. This makes the user aware of the issue, and also allows him to use his expert knowledge to infer the state himself, or ideally, to configure the visual analytics system in such a way that the state can be inferred automatically for subsequent, similar cases.

3.3 Data Types and Structure

So far we have talked about data and data types in a more abstract way. For most intents and purposes, the actual nature of the data does not matter as much, as long as its not highly specific, “complicated” data. To provide an example for the data encountered for the cluster type analyzed for this thesis, let’s take a look at what is being collected from a controller node.

Listing 3.1 Simple structured event, with the four always present fields: *name*, *currentTimeMillis*, *nanos* and *data*.

```
{
    "name":              "root.project.core.Main.startup",
    "currentTimeMillis": 1460581208105,
    "nanos":             84897412,
    "data":              null
}
```

1. **Dropwizard Metrics**

   These are metrics collected by Dropwizard [Dro17], a Java framework for application metrics collection. There are various collection categories (Meters, Gauges, Counters, Histograms, Timers). All of these essentially track a single value in regular intervals. An example for such a value is the length of a queue, the amount of free memory, or the frequency in which an event happens. Some categories automatically compute additional information (such as minimum, maximum or mean, as well as median and 75th, 90th, 95th, 98th, 99th, and 99.9th percentiles). Effectively, this provides values which can be easily visualized as a line chart, for example.

2. **Log files**

   Log files come from many different sources and in many different formats. Generally, they are human readable, but if they contain many lines, or multiple logs have to be read at the same time, filtering, highlighting or an altogether different method of representation should be chosen.

   Many log files are chaotic and don't follow any conventions. Even if they are structured and are following a well defined format, an expert user still needs to provide pattern information in order to store and visualize the log in a meaningful way. As lines in log files generally have a timestamp, it is important to know how to parse the timestamp. Furthermore, there might be other reliably used information, e.g. tags, which can be helpful for filtering and visualization reasons.

3. **Structured events**

   Events, in this context, are essentially lines in a very easily machine-readable log file in JSON [JSO13] format. They follow a well-defined pattern for their basic structure (see Listing 3.1 for a simple example):

   a) **name**

      The name of the event, which is essentially a fully qualified, dot separated type-string. The dots represent a hierarchy, so events exist in a global hierarchy,

which makes grouping easier and helps with understanding what information they contain for what part of the computer system using these events.

b) **currentTimeMillis**

This is the number of milliseconds that have elapsed since midnight, January 1, 1970 UTC and the moment that this event was created.

There are some noteworthy properties to this field, which are as follows:

i. Given that the time has been correctly set on all nodes for which events are collected, all events for a given cluster can globally ordered by time.

ii. If an event is created by the same node very shortly after another, this date might be the same for both events. In that case, the *nanos* field can be used to determine which event was created first.

iii. The timestamp does not necessarily carry the information a user might be interested in. Given two events A_1 and A_2, for which $currentTimeMillis(A_1) \leq currentTimeMillis(A_2)$ is true, we cannot assume that the actual order of execution of instructions leading up to the creation of the events happened in the same order, as this may be dependent on scheduling behavior on any given node. However, since this is an issue that cannot be easily be mitigated by the visual analytics system, the assumption is being made that the order the events are emitted in are as represented by the timestamp, and the expert user should be aware that this effect can occur.

c) **nanos**

This is a nanosecond counter which *does not* relate to any wall clock. It can be used to establish an order, in case two or more events have the same *currentTimeMillis* timestamp[1].

d) **data**

This is a field which is always present, but does not have to have any content. Any type of information can be put here, and expert knowledge is required to create visualizations which meaningfully interpret the information for any given event name. Listing 3.2 shows an event which has complex, nested data in its *data* field. One view in the visual analytics chapter will

[1] This is only possible if the events in question originate from the same Java process.

Listing 3.2 Complex structured event, with nested arbitrary information contained within the data field

```
{
    "name":"root.project.side.sync.complete",
    "currentTimeMillis":1460581270777,
    "nanos":238947789423,
    "data":{
        "__class":"root.project.side.Network",
        "networkType":null,
        "status":"DESIRED",
        "reports":[ 0, 2, "COMPLETED"],
        "startTime":384133414,
        "pushResults":{
            "duration":0,
            "result":"SUCCESS",
            "messages":0,
        },
        "createTime":38974189247189
    }
}
```

4. **Other/Metadata**

   Many other data types are collected. There is general context data, system data and version data. The data directly collected from the switch nodes, for example, is one big text file containing all information on that particular switch, in a non-structured format.

3.4 Digestion Pipeline

The inhomogeneous data in a support bundle is hard to work with without any transformation. A processing pipeline is necessary to have the data available for easy consumption by the visual analytics system, as the data is not stored in a way which allows for search, filtering, or random access in general. Figure 3.10 shows the pipeline.

Depending on the type of the underlying data, different data storage structures are useful to maximize the accessibility of the data. Time series data can be very saved very space efficiently in a time series database, such as InfluxDB [Inf17]. More unstructured but still time based data, such as log files or events, could be saved in a database which allows efficient full text search on entries, such as Elasticsearch [Ela17]. The rest of the data, specifically the data that has been collected at a specific point in time, could also be saved in Elasticsearch to benefit from the full text search capability inherent to that

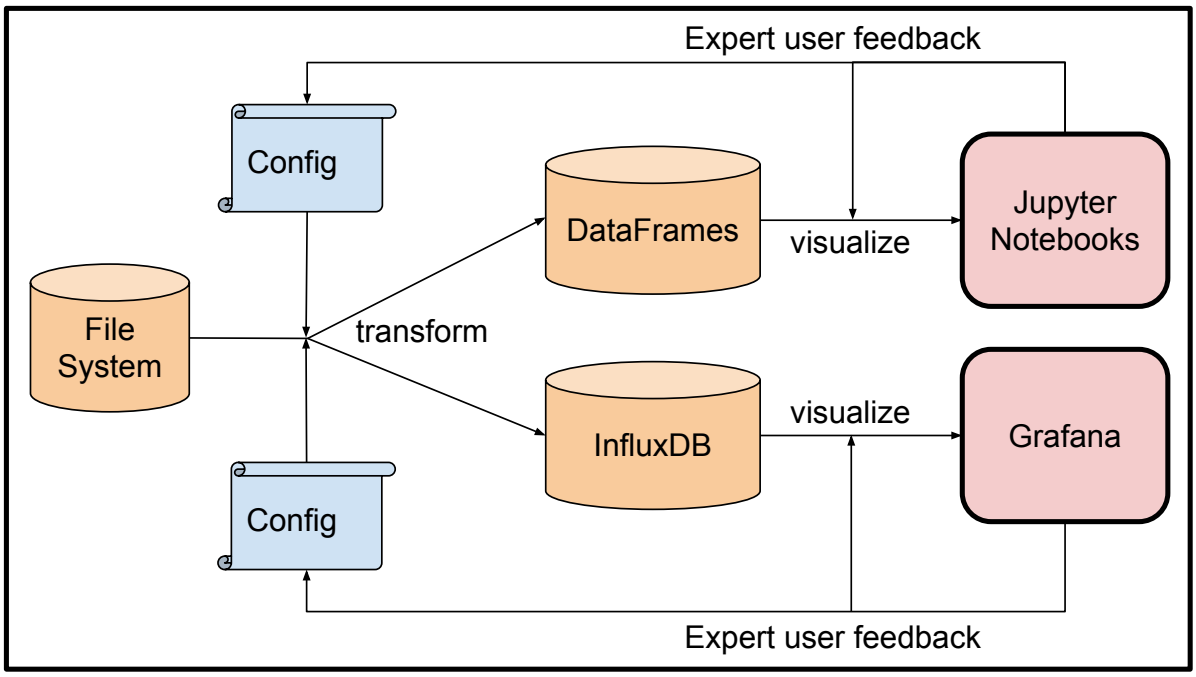

Figure 3.10: Digestion pipeline for the visual analytics system

database. The capabilities of Elasticsearch have not been required yet for the data of interest to experts, but are likely needed in the future. For the purpose of exploring the data using the Jupyter notebooks (see Chapter 4.1), time-series and structured event data is also stored using *Pandas DataFrames* [Pan17], a relational database format that can be easily handled by python code in notebooks.

A pipeline which can be configured also allows expert users to make changes regarding what data is processed, and how it is processed. This is very similar to the visual analytics pipeline, as described by Keim et al. in *Visual Analytics: Definition, Process, and Challenges* [KMS+08].

3.4.1 Type Inference

For any given data in a support bundle, we have to decide if we need to transform it, how we transform it, and how we store the result.

A simple default, which can be applied to basically any file, is to either ignore it, or to save it as a big blob in a database. If line breaks can be discerned in a file, it can also be saved line-by-line to allow for random access to the unstructured data.

As the number of encountered formats and encodings can vary at any given time, it is unfeasible to hope to correctly detect and store each file. Moreover, if a file were to be consumed automatically in a wrong way, the user of the visual analytics system may come to conclusions based on distorted data.

As such, a mapping is employed that defines which file, if present, has to undergo which transformation, and in what way the final result should be stored. During transformation, data may be added, removed, or transformed into other data. Potentially complex processing may be done on the data, resulting in derived data (e.g. inferred state). The result of this digestion can be accessed by the visual analytics system introduced in Chapter 4.

3.5 Context

Support bundles are a simple solution to the question:

How to collect as much data as possible from my cluster with the least amount of effort?

Some distinct issues with the collection procedure have been highlighted in this chapter. Some of these issues warrant the question:

Why not change the way data is collected?

3.5.1 Continuous Digestion

In Chapter 7, a solution is proposed for employing the same visual analytics system introduced in Chapter 4 based on continuous digestion as opposed to batch processing of support bundles. The reason that this is not done in the first place is that it requires extensive changes in the way clusters operate in the way they collect data. It also raises security questions, as potentially sensitive data has to be streamed to a destination where it can be collected. Furthermore, it turns the passive visual analytics tool, which works on data passively collected in the past, into an active component that can have a bigger impact on the inner workings of the cluster.

3.5.2 Automatic Type Inference

As opposed to the opt-in way of digesting files, which can be described as maintenance-intensive, efforts can be made to automate as much of the preprocessing as possible. In Chapter 7, we will take a look at a proposed mechanism to make the processing less

labor intense, and discuss the pros and cons of that approach as compared to the manual definition of the processing pipeline.

4 Visual Analytics Techniques

In this chapter, we will take a look at two different parts of the visual analytics system. Both are based on the outputs of the processing pipeline shown in 3.4. Section 4.1 will talk about the use as Jupyter notebooks as platform for visual analytics, while section 4.2 will talk about the use of Grafana for data exploration for time series. Furthermore, various visualizations and interaction concepts will be introduced in section 4.4, which have either been prototyped or were worked out together with expert users and deemed useful additions to be integrated in either Jupyter notebooks, or Grafana.

The main goals for the visual analytics system achieved using the techniques explained in this section are:

1. Allow easy exploration of all available data for a given cluster
2. Make correlation of time-series data with structured events easy
3. Augment event and time series visualizations with state information of the system
4. Allow expert users to look at data with no other requirements than a browser
5. Allow exploration data from multiple support bundles on a single timeline for a given cluster. Users can arbitrarily select a window on that timeline, and get to explore the data across boundaries of the underlying support bundles.
6. Enable side-by-side comparison of data from separate support bundles or clusters, e.g. in order to find patterns in otherwise temporally unrelated data

We will first look at the most fleshed out system, the Jupyter notebooks, followed by the Grafana Data exploration panel, and conclude this chapter with a look at the prototypes and potential improvements for the former two systems.

4.1 Jupyter Notebooks

According to expert users, pure visualization of data for the purpose of looking for issues in a distributed system has limited use. This stems from the fact that, as explained in Chapter 3.3, data exists in a multitude of formats, and these formats, as well as the types of data collected, keeps changing. Since the expert users tasked to find the issue with a distributed system often have intimate knowledge about this data, they can often look at it in its raw form, or with simple, often console based tool, and eventually extract the information they are interested in. A platform for the visualization and exploration of this data may be present, but is often ignored, either due to expert users not being familiar with the technology, or due to the limited scope in which the data can be viewed.

While the former problem is hard to overcome and requires expert users being introduced to a new workflow (and in doing so making them put aside proven methods of searching for problems in distributed systems they have become familiar with), the latter problem needs to be addressed in order to actually offer a visual analytics system which has advantages over those existing, mostly console based debugging approaches.

In dialogue with expert users the idea was formed that the inherent heterogeneity of the data in combination with vastly varying needs of expert users of the data makes it prohibitively difficult to build an all-encompassing solution. Since effectively all expert users working on finding issues in distributed systems have advanced knowledge in programming, a system was built that combines visualization, interaction and expert feedback.

Figure 4.1 shows a Jupyter Notebook. Jupyter notebooks are web-based environments in which an interactive python environment runs on a server and is shown in the browser. Users can enter and execute valid python code. The state of the python environment is persistent, which means that any code executed can alter the state of the environment, e.g. a variable created is persisted and can be accessed in another part of the notebook.

The data of each support bundle that has been processed by the digestion pipeline in 3.4 is accessible in an SQL-database-like format (*Pandas DataFrames* [Pan17]) from any notebook that is part of the visual analytics system. The data can be programmatically explored, and can be rendered in the web-browser, where multiple views are linked in their time axis. A feedback loop can be established by interacting with the resulting visualizations, and by altering and re-executing parts of the code in the notebook, which can transform the data, or change the nature of the rendered result. Given that all data a given expert user is interested in is present in the dataframes, any kind of transformation can be done directly in place, and the result can be textually or graphically presented.

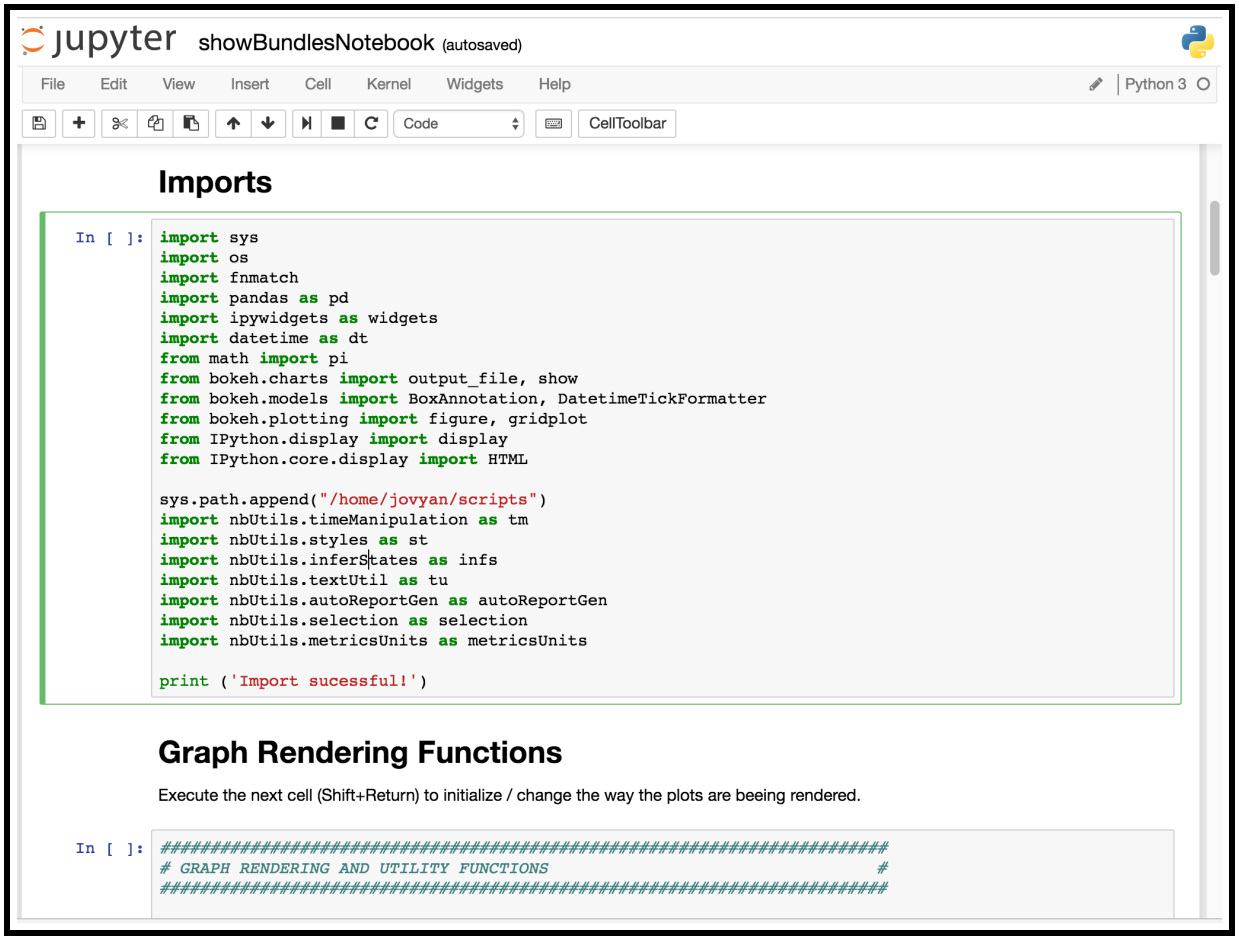

Figure 4.1: A Jupyter notebook with python code

The only part of the visual analytics pipeline which cannot be immediately affected from this system is the transformation of the raw support bundle data into the dataframes. For this, the user has to make changes to the underlying processing pipeline (see chapter 3.4), and then trigger the re-processing for this particular data set. After this processing is done, the new dataframes have replaced the old ones, and can simply be accessed from the notebook as before.

4.1.1 A Basic Notebook

The system was introduced with a single notebook, which served as both a help for expert users to understand how they could interact with the data, as well as provide some useful visualizations to start with. Figure 4.2 shows how support bundle data is first being selected to be loaded into the python environment. After the selection of

Figure 4.2: Data selection in the basic notebook

the dataframe, the user can either programmatically explore the data, or selects any number of time series from the shown list to render them. Figure 4.3 shows how, using this basic notebook, all selected time-series data, as well as all structured event data (see chapter 3) encountered in events.log, which is collected by every controller, can be rendered together and aligned on the time axis. Additionally, either data is put into context of the state of the system with the use of highlighting to show the state that the controllers have been in at a specific point in time. Its possible to pan and zoom into any view arbitrarily, and all other views will stay aligned on the same time axis. More notebooks have been created by the expert users to highlight more specific problems, and will be presented in chapter 5.

The biggest advantage of the visualization with this basic notebook is the ability to correlate events with both the state of the system, as well as metrics collected at the same point in time, across multiple nodes. To access the same information without this system, expert users would have to extract the necessary information from many different sources, and painstakingly sift through them, trying to keep the state of the system in the back of their head, while progressing through multiple files in global chronological order.

The technology used to visualize the data and allow for data exploration with python in the notebook is Bokeh [Bok17]. Bokeh allows the creation of various visualizations and supports interaction with the data in real-time. Since the python environment in the browser could not handle presenting all data on the client-side (the browser) as the amount of data is magnitudes too large to be handled there, Bokeh is used to present lightweight front-end visualizations, while all the heavy lifting is done in the back-end (the python environment running on the server that hosts the notebook).

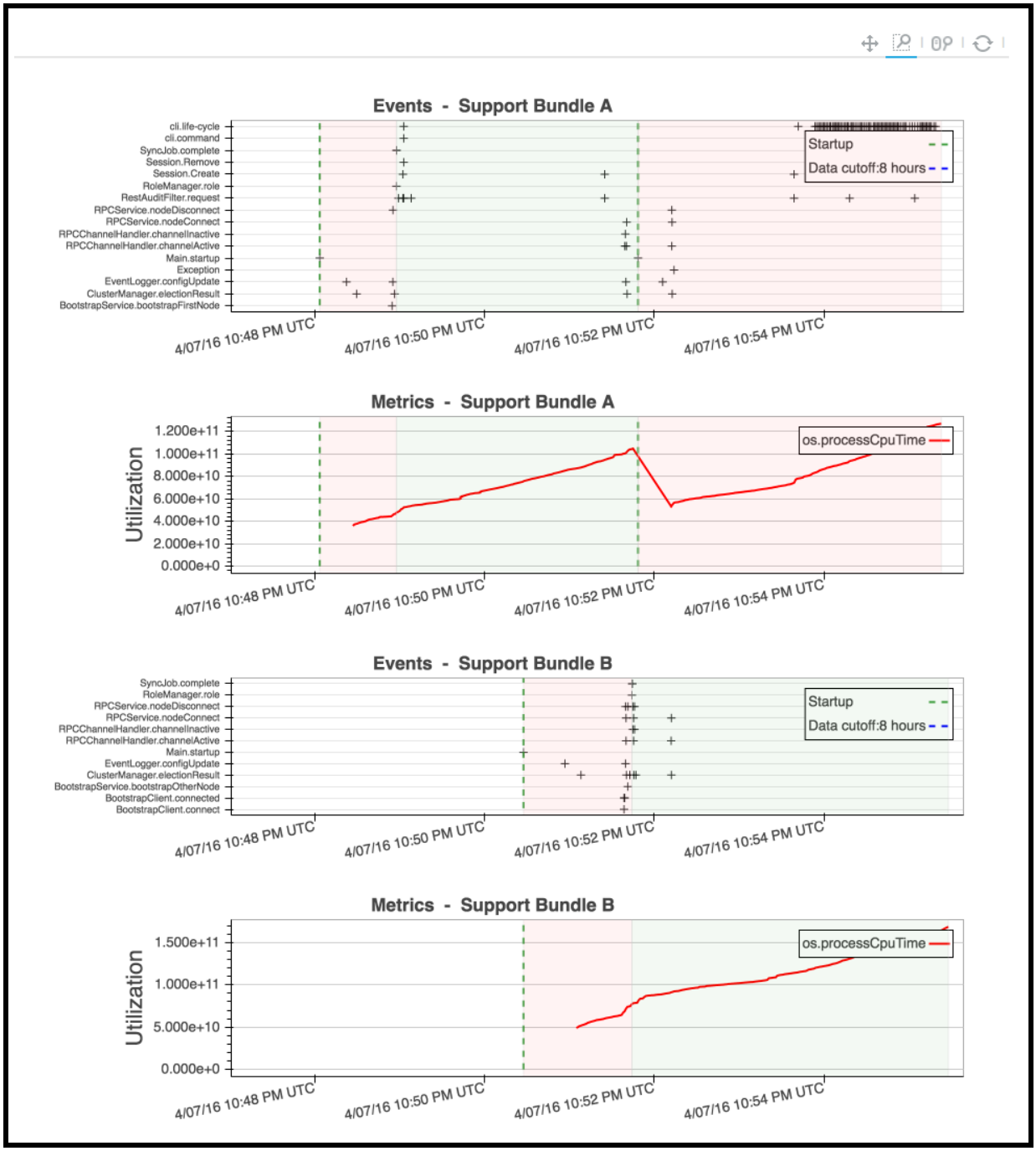

Figure 4.3: Multiple coordinated views of cluster data

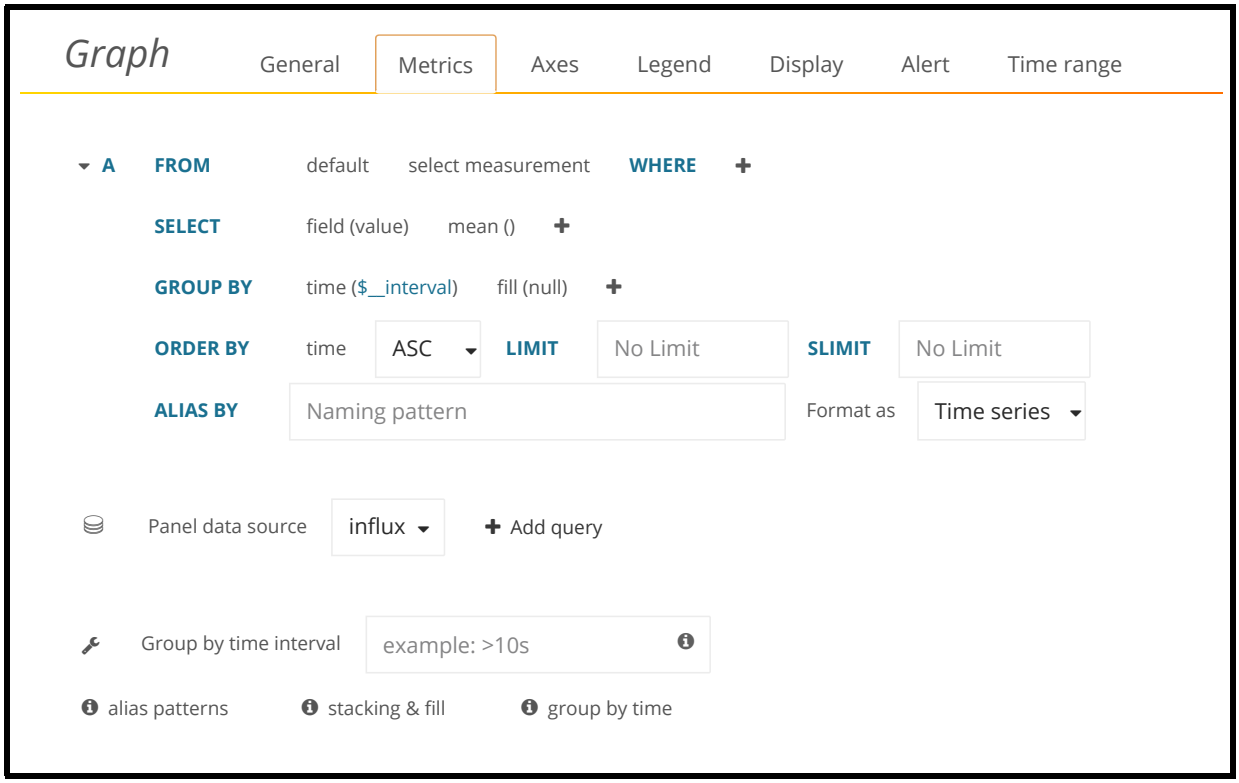

Figure 4.4: Query definition in Grafana

4.2 Data Exploration with Grafana

While the approach from chapter 4.1 using Jupyter notebooks and dataframes has been the most tried, and was used for most of the case studies in chapter 5, effort was invested to streamline the selection of data to be rendered. The result is a prototype which uses Grafana, a web-based data visualization tool, to allow intuitive data exploration *across multiple support bundles pertaining to the same cluster*, something not possible with the notebook based solution.

As explained in chapter 3.4, the time-series data output of the data digestion pipeline is also fed into InfluxDB. From there, it can be very easily consumed by Grafana. Traditionally, (and as shown in Figure 4.4) users of Grafana would select a type of visualization (e.g. a Line Chart), which will place it on their Grafana dashboard. From there, they would open the options of the newly created panel, navigate to the options, and select a data source, e.g. InfluxDB. Following this, they would set up the query to create the desired visualization.

Since the visualization based on Grafana is based on clusters, and not on support bundle as the driving element, the data of two support bundles can be shown at the same time,

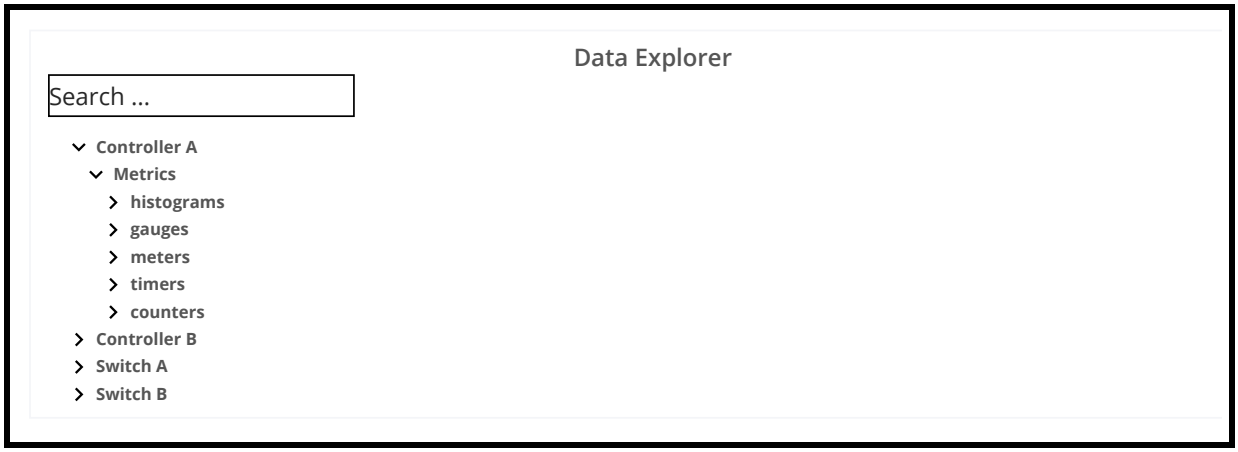

Figure 4.5: Data exploration panel for cluster data

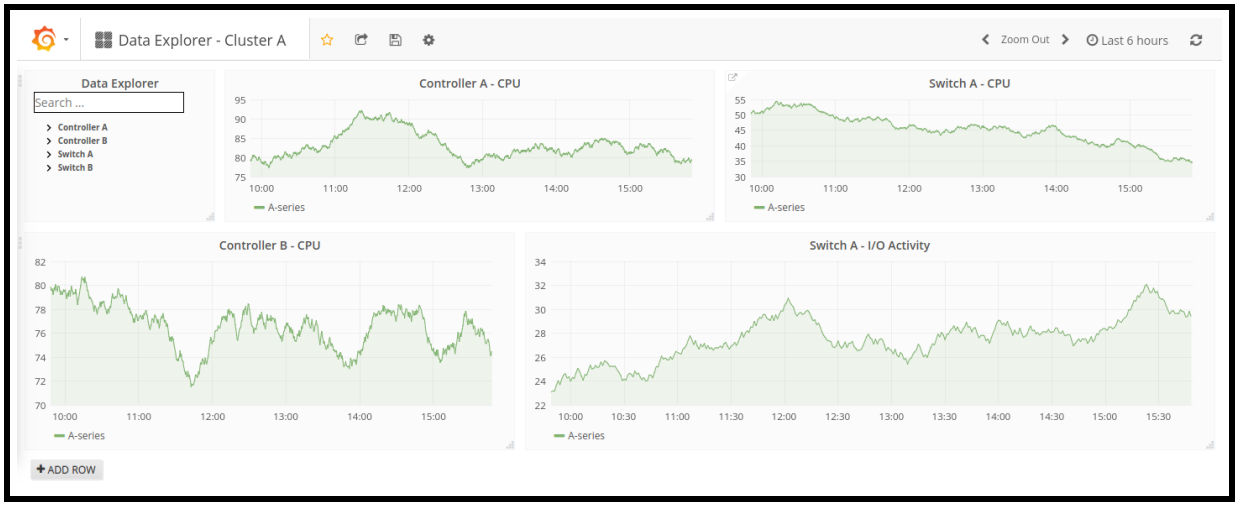

Figure 4.6: A Grafana dashboard with coordinated multiple views

simply as the concatenation of the data present in both bundles. If there is a gap in the data between the two bundles, the system simply highlights this gap (e.g. by stopping to render a lineplot where data is missing, as opposed to interpolating missing data points).

4.3 Exploration Panel

Given that we know about the nature of our data, and what visualizations are possible, this workflow seems unnecessary complex. In order to streamline data exploration, an exploration panel (see Figure 4.5) was developed. An exploration panel is a panel

added to a dashboard, which acts as the centerpiece for all consecutive visual analytics activities. The panel consists of a Tree displaying all data available for a given cluster. The tree's structure resembles the hierarchy of data encountered in all support bundles pertaining to the particular cluster. The hierarchy in which most data is encountered is directly related to their position in the directory tree, although for some data, such as the time-series data collected by Dropwizard (which would otherwise be encountered in the directory hierarchy), the data is shown as a direct child node of the respective node. The tree can be navigated by expanding the children of each tree-node, up until the leaves of the trees are reached. The leaves, once clicked, open a view showing the data to which they refer, skipping all the unnecessary steps traditionally necessary to populate a panel with data. Figure 4.6 shows a dashboard with multiple panels, showing various time-series data that originated from a cluster. Panning and zooming in any panel will influence the visualization of any other panel as well, since all views are linked.

The panel also contains a search field, which allows the expert user to filter the potentially vast amounts of different data types associated with this cluster. All data not matching the terms entered into the search field are hidden, while all tree nodes matching the term are revealed up until the depth where the keywords match. The total number of matching results is aggregated for each non-leaf tree-node next to its name, but children are not unnecessarily expanded to keep the results uncluttered.

As of the time of this writing, only standard time-series data, as encountered in the Dropwizard data described in Chapter 3.3, can be explored this way.

4.3.1 Advanced Search Field

Given that a cluster may contain hundreds of thousands of data sets, more advanced filtering might be desirable to find the relevant data set with the search. While not implemented in this prototype, the search field could employ a more sophisticated algorithm to return the most relevant results first. What is considered the most relevant could be automatically determined, e.g. by recording what choices expert users usually make, and in turn more prominently present those choices.

4.4 Further Concepts

In order to provide users with a useful visual analytics system, we have to visualize both the data of interest (As described in Section 4.2), but also highlight all the non-intuitive data inaccuracies described in Section 3.2.2 and 3.2.3. Many of the subsections in these sections go into detail about how metadata about the collection process can be

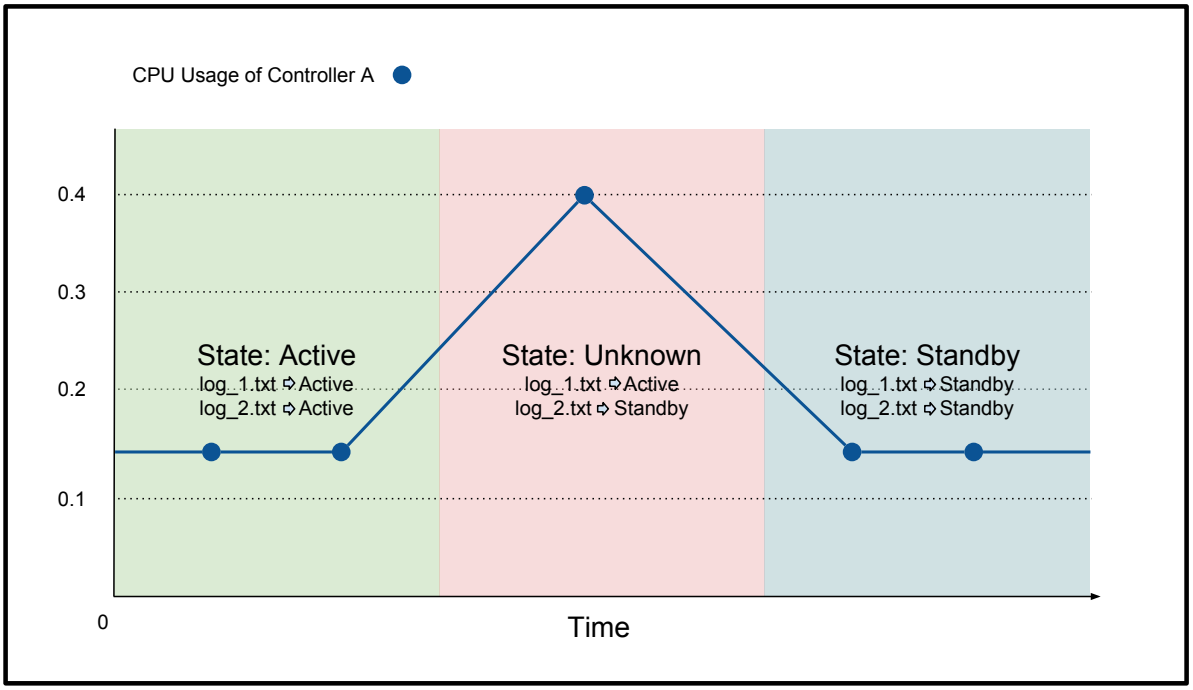

Figure 4.7: State conflict between two data sources

highlighted along the actual data of interest in order to point out potential inaccuracies with the data.

4.4.1 Mitigation of Bundle Problems

In Chapter 3.2.2 and 3.2.3 we discussed how the data collected within a support bundle has some issues attached to it, as well as how problems can arise when more than one support bundle exists for a cluster. In this section, we propose solutions for how these issues can be brought to the attention of the user to help him understand the data he or she is exploring better.

1. **Data inconsistency**

   As explained in Chapter 1, some of the data contained within a support bundle can be redundant. The redundancy in this case is a potential source of inconsistency. Figure 4.7 shows how state-unrelated data is contextualized by highlighting the background based on state that has been inferred from data with state information. Depending on the needs of the user, more than one state defining data source should be used, if it is present. This serves two purposes:

a) In a correctly working system, components are in valid states. If due to a bug, however, any component is in an invalid state (or a global illegal invariant exists), then highlighting this conflict directly helps the user to understand what went wrong in the system, without even having to explore more data in detail. Figure 4.7 shows such a scenario, where for a certain time period, it is unclear what state Controller A is in, since two state sources claim opposite states. The highlighting as part of the line plot points out the unclear state situation and helps the user to interpret the data correctly.

b) State transitions might not be instantaneous all across a distributed system. If the state of every component is inferred from a single source, and in turn used to highlight the state as context for other data, then other data might be shown in the wrong context for short periods of time. If different state data sources cause slight state uncertainty for the whole system during transitions, then for a certain time frame, the data of interest should be highlighted with this uncertainty, as shown in Figure 4.7.

Since the notion of state (or other type of invariant highlighting) is highly dependent on the actual data and the use case, it has to be added to the visual analytics system by an expert user. Highlighting is used to show the state of controllers in the Jupyter Notebook [Jup17] in Chapter 4.1.

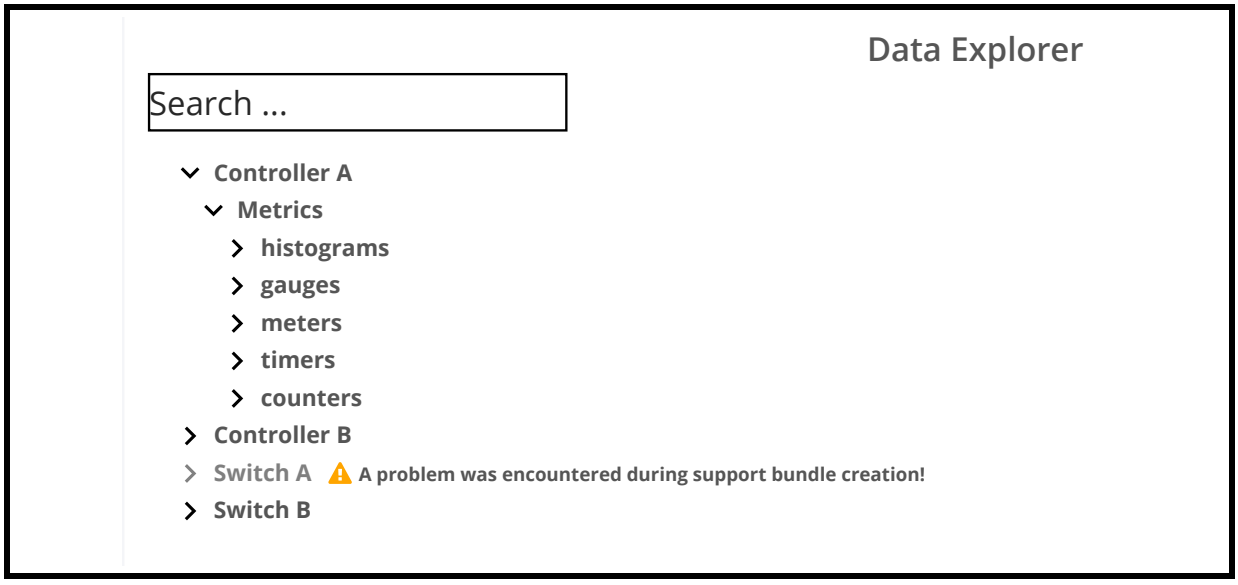

Figure 4.8: Highlighting the absence of a switch

2. **Switches unreachable**

   In chapter 2, we discussed how there can be a situation where a cluster during the creation of a support bundle has a switch which is not reachable. Since the switch is a node that if of interest for the analysis, we need to make the user aware that a specific switch was not reachable. This requires another set of data that informs us of the existence of the switch in the first place, but this data is generally present and taken from the controller that starts the collection process (Otherwise, the bundle creation would not which switches to contact to begin with). Figure 4.8 shows how the absence of a switch and its data can be visualized based on the data exploration panel introduced in chapter 4.2. Data that is only partially missing can be highlighted differently to bring the issue to the attention of the user, as shown in Section 5.

3. **One or more controllers unreachable**

   Very similar to the issue of a switch not being reachable is the issue of not being able to reach a controller. While the absence of a switch can often be mitigated by a cluster by routing messages over alternative switches, the absence of the data of a controller hints at a much more serious issue, and thus should be brought to the attention of the user in a more prominent way. Figure 4.9 shows how the data explorer can highlight the absence of a controller.

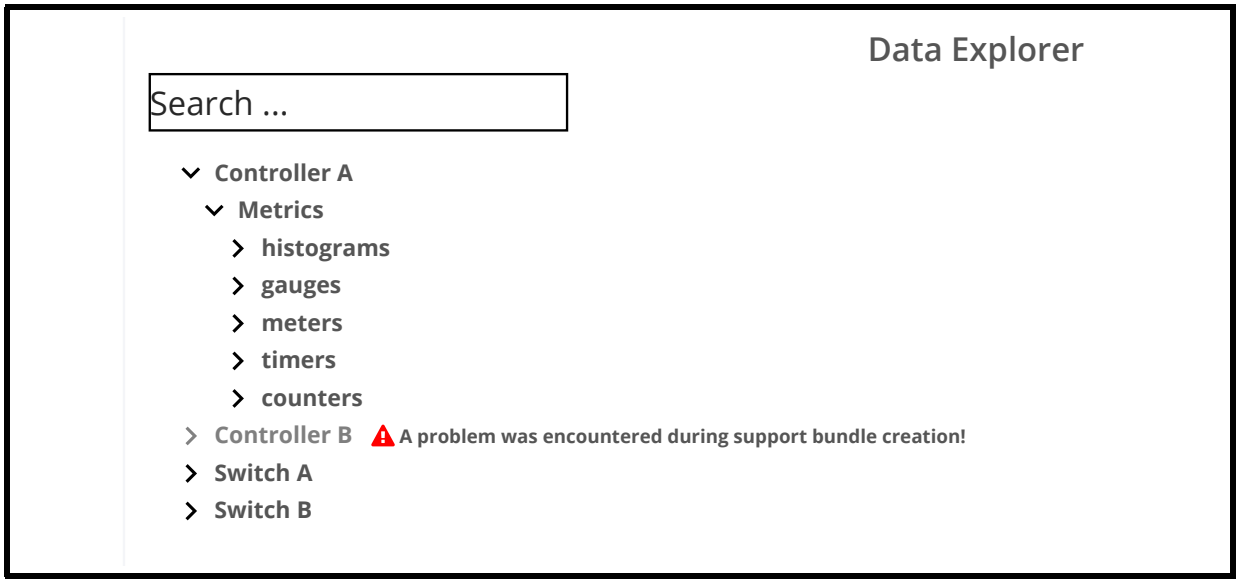

Figure 4.9: Highlighting the absence of a controller

4. **Collection influencing cluster**

   In chapter 4, we went into detail about how the creation of a support bundle can influence the data it collects in various ways. This is undesirable but not easy to prevent. To help the expert user understand that the data he sees may have been influenced by the support bundle creation, we need to highlight the time window during which the collection happened, whenever the data collected by the bundle is being explored. Figure 4.10 shows how such data is being highlighted with the time frame of the bundle creation, thus potentially explaining the spike in CPU usage at that specific point, which might otherwise point towards a problem with the cluster that does not exist and is purely a side effect of the collection.

5. **Corrupted data collected**

   Having explored the issue with corrupted data in chapter 5, we now present a solution for highlighting such corrupted data during exploration. Figure 4.11 shows how a single data point, as well as an unknown amount of data points that are corrupted can be highlighted. Some data follows a predictable pattern (e.g. a line in a file represents a single data point at a specific point in time), which could be used to more precisely point out missing data points. This assumption, however, requires extensive knowledge about the data, whereas simply highlighting the time frame from the last non-corrupted data point before a corruption occurs to the next data point after the corruption is general-purpose. It cannot be applied to data without a time axis, however that data often does not have a structure which

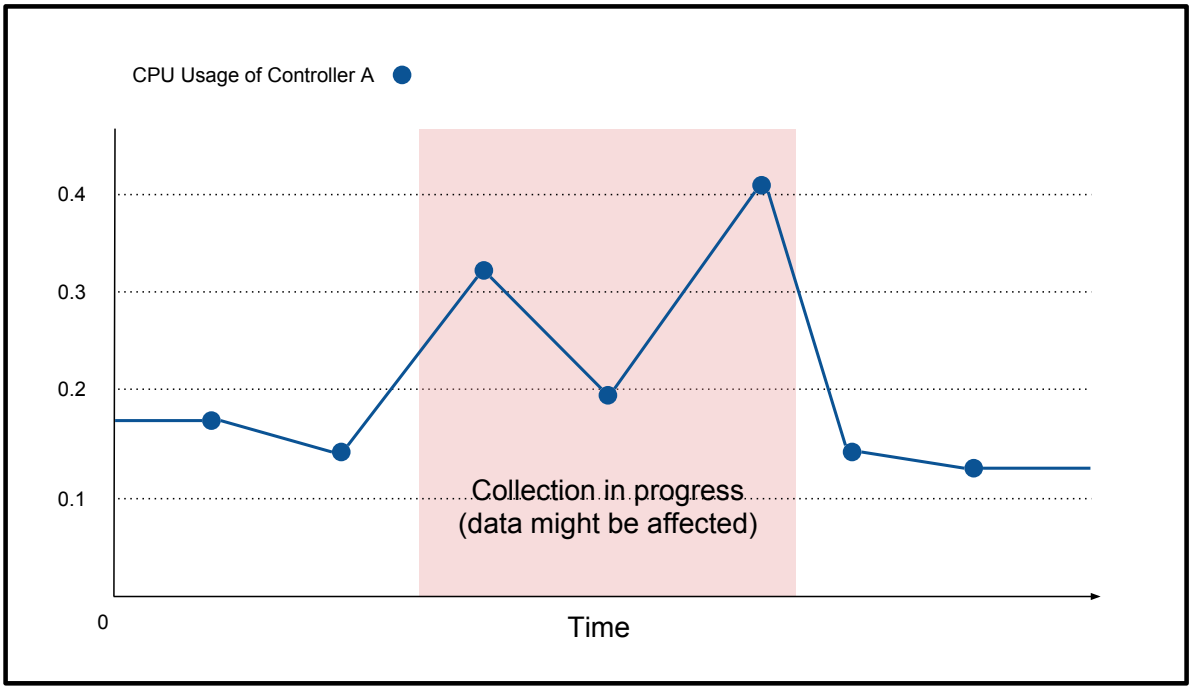

Figure 4.10: Highlighting the bundle creation time frame

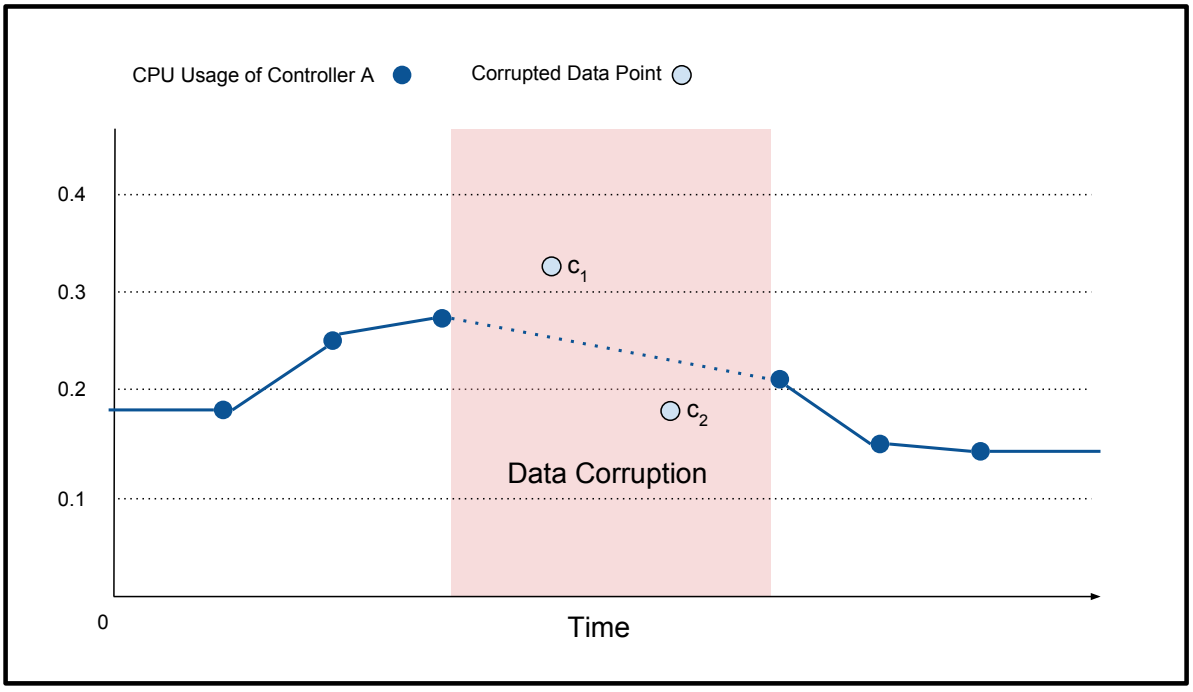

Figure 4.11: Highlighting data corruption

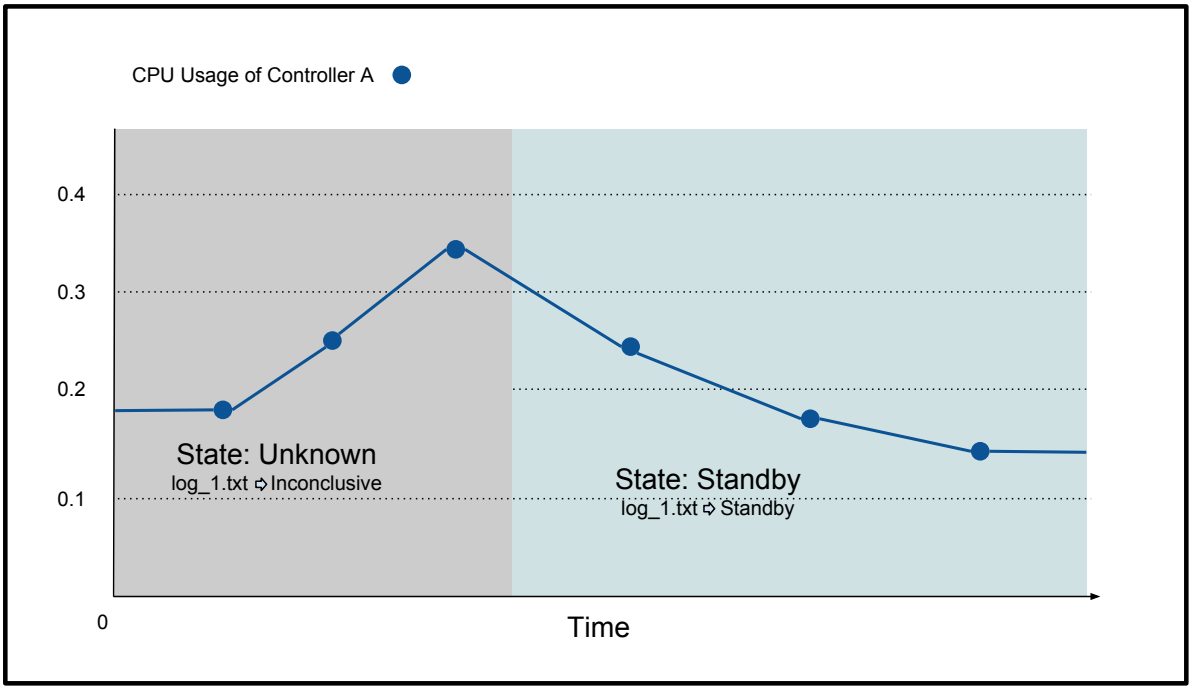

Figure 4.12: Highlighting insufficient state information

can be considered corrupted, or corruption would make the whole file unreadable (e.g. a file with a single, broken JSON object).

6. **State data not sufficient**

   As explained in chapter 6, some data carries state information. If the data in question is very verbose or redundant, it might be possible to infer the state of a system with any single data point. More often than not however, there is a causal element to the data, meaning events in the past influenced the state, but without access to that data, we cannot infer the state with every set of data in the present. In some cases, state can be inferred again once some key data point appears. Figure 4.12 shows how other data is shown in the context of the state, and how during a time frame where the state is unknown based on the knowledge present, the state is described as unknown. Once a data point reveals the state, the highlighting indicates the current state and helps giving the explored data come context. The same technique can be applied in case there are gaps of stateful data between multiple support bundles. In this case, for all data missing between two support bundles, highlighting is applied (e.g. state: *unknown*).

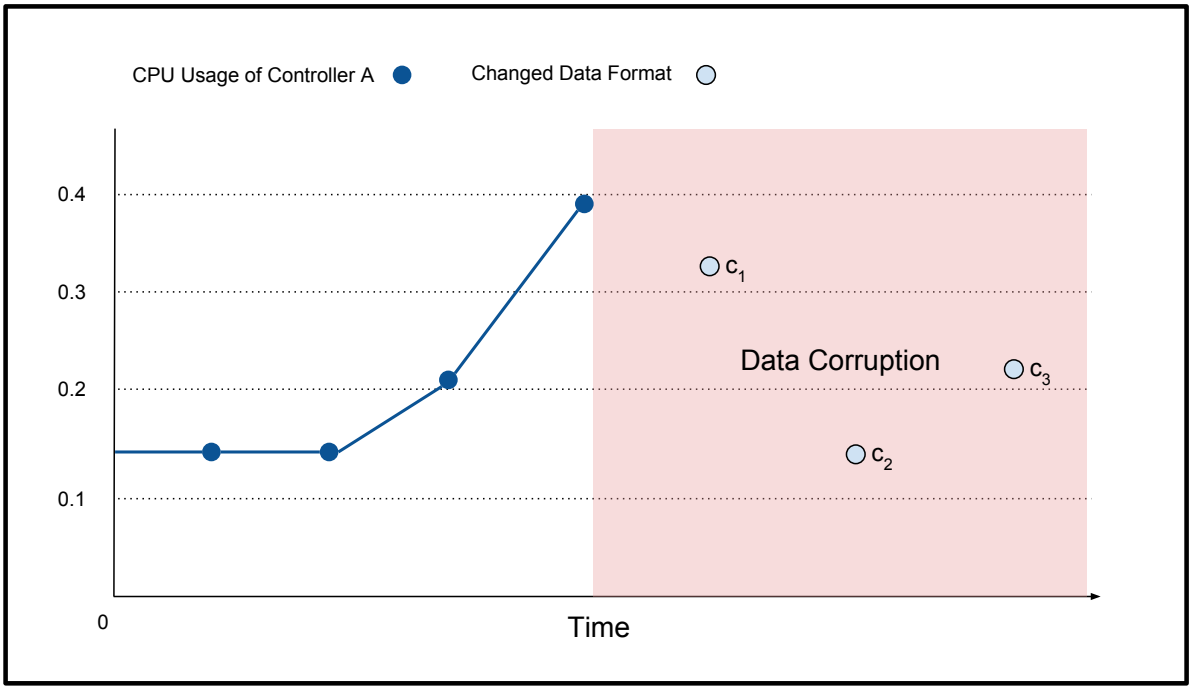

Figure 4.13: Highlighting change in data types

7. **Change in data types or format**

   In chapter 7, we discussed how data may start or stop to be collected by a node, or how the structure of the collected data may change over time. The disappearance or reappearance of data is relatively trivial, as there is generally no assumption being made about the presence of any data. Much more complicated is the change of the data structure. If information is simply added or removed to a structured data format (e.g. JSON), then this is generally not complicated. However, if the data format itself is changed, or any other change takes place which causes whatever technique was used to parse the file to not be applicable to the data anymore, then the remaining data might simply be considered corrupted (see section 5). In this case, the visual analytics system relies on the user to identify such a case as depicted in Figure 4.13, where starting from a certain point onwards, all data is considered to be corrupted.

8. **Creation not instantaneous**

   As mentioned in chapter 8, the creation of a support bundle does not happen in an instance, but rather over a period of time. The data present in a support bundle is directly affected by this, as the collection of some data concludes later than some other data. There are two separate approaches how this issue should be addressed.

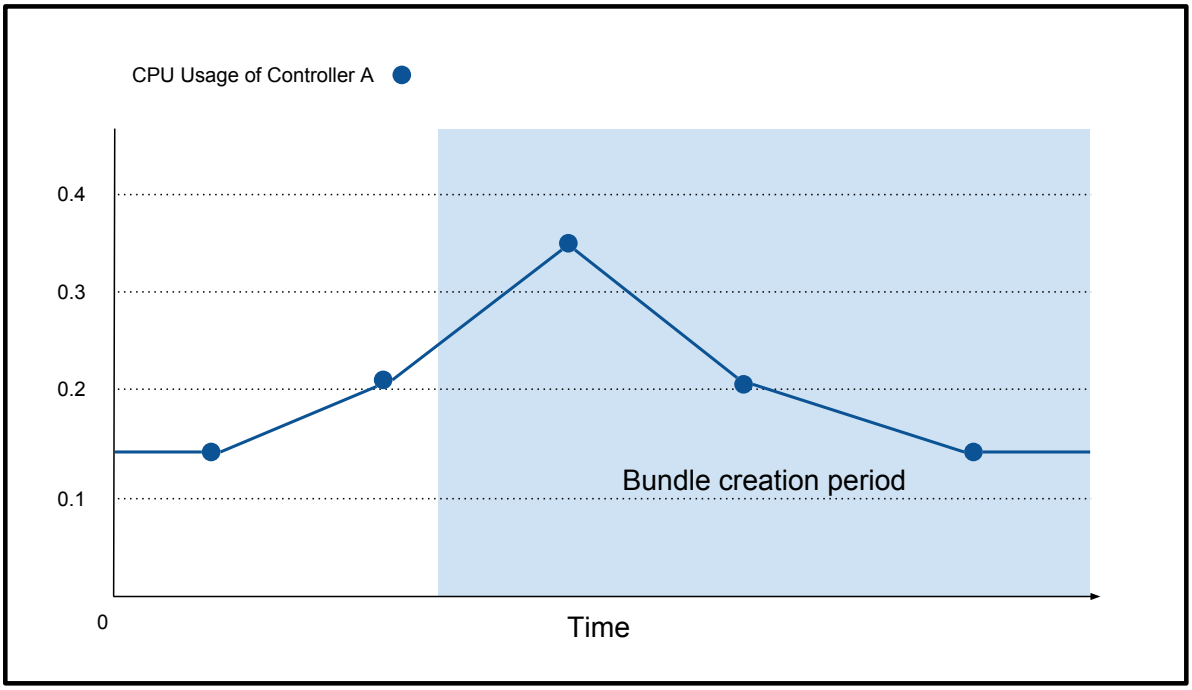

Figure 4.14: Highlighting the bundle creation period

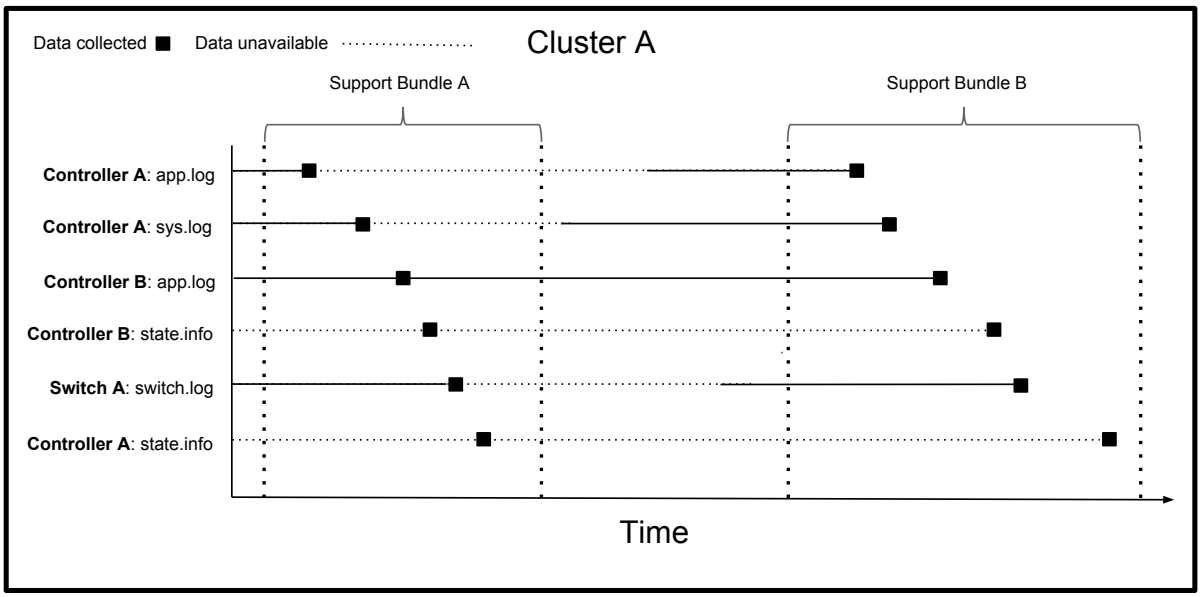

Figure 4.15: Visualizing the time frames of availability of all data

First, the user needs to be able to look at any data in the context of when the support bundle was created, as shown in Figure 4.14, to help understand that some data points might have been influenced by the collection procedure.

Secondly, a meta-view which has the sole purpose of allowing the user to see at what point which data has been viewed. Figure 4.15 shows how all data available for a cluster could be visualized in a meta view showing when the data has been recorded. To make this possible, the information when what data has been collected needs to be extracted from metadata about the collection. Every support bundle has a log file which contains information about when what data was collected, and can thus be used to infer relatively accurately the time specific data has been collected. The visualization should be filterable, and a selection of any time frame for any data shown should open a view on the respective data and set the observed time frame in all views to that time frame.

4.4.2 Collaborative Markers

Expert users suggested that providing the ability to comment on specific data points with markers would be a useful feature, so that other users are able to see directly what another user pointed out. Figure 4.16 shows a proposed visualization of markers for various data. This approach works for any type of data visualization used by the visual analytics system, and the following marker types are suggested:

1. **Area markers**

   Area markers could be put down by clicking and holding the mouse over any data with a time axis. As a result, a highlighted area would be displayed over the data similar to the other suggested highlighting techniques, together with a comment left by the author of the marker.

2. **Point markers**

   Point markers could be set by a single click, and would represent a single point in time in any time series data. They can be represented by a vertical line in any time series visualization, and show text by the author.

3. **Data markers**

   Data markers would be applied to a whole data type for a cluster, and could be shown in the data exploration panel suggested in section 4.2. Similar to the other markers, text written by the authors is shown alongside the marker.

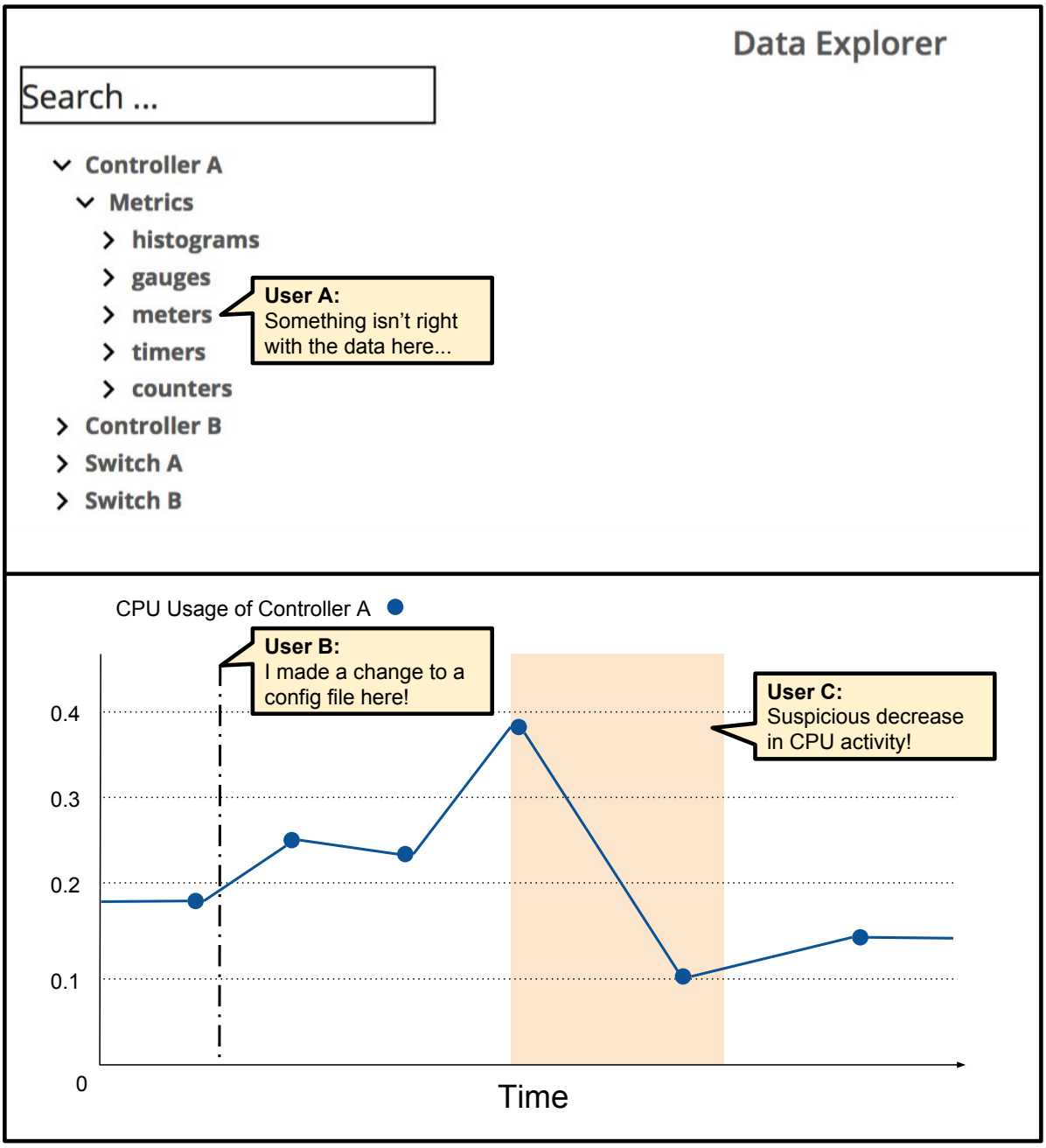

Figure 4.16: Different types of collaborative markers

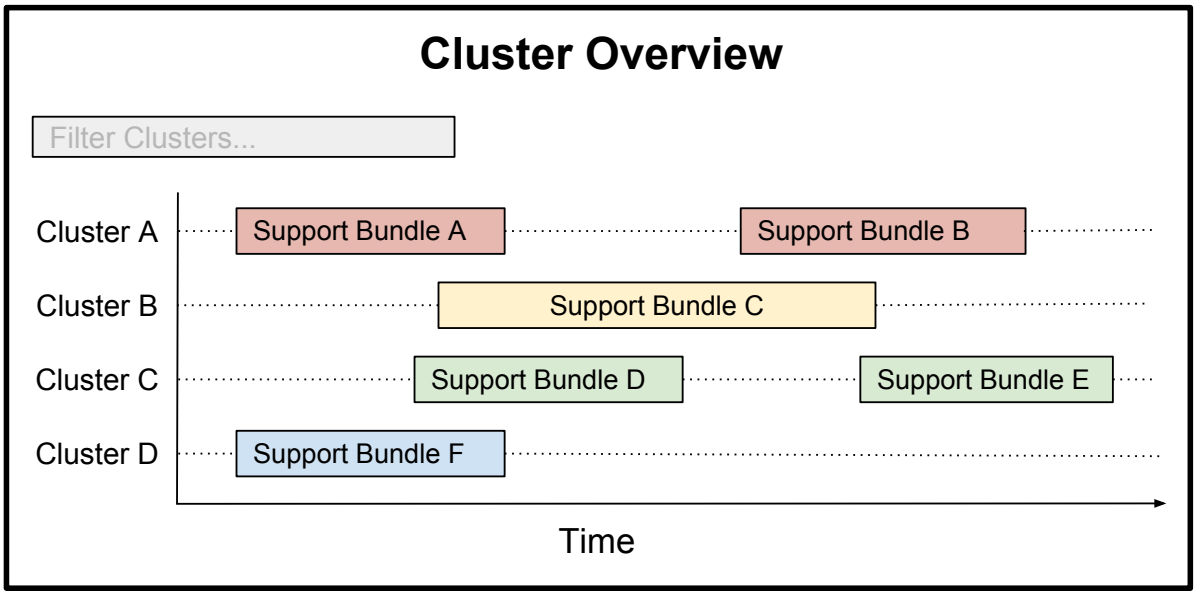

Figure 4.17: Top level visualization of cluster data

4.4.3 Visualization of Clusters

In the Jupyter notebooks, data selection is based on concrete support bundles. However, as prototyped with the Grafana data exploration, a more suitable solution would be to think of the highest hierarchy level, the clusters to which individual support bundles pertain. Taking the (albeit weakly defined, see chapter 1) borders of support bundles into account, a top-level visualization as depicted in Figure 4.17 can be employed to give expert users an overview over *all* clusters for which support bundles have been processed ever. A search field would assist the user in finding the correct cluster, while selecting a part of the time axis for a specific cluster would open a dashboard, focused on the selected cluster, during the selected timeframe. From there, the data could be explored using the data exploration panel, enabling users to very quickly and effectively delve into the data.

5 Case Studies

This chapter contains case studies in which experts at a Data Center Networking company have been using the visual analytics system to help resolve issues from their bug tracker. The components of the visual analytics system introduced in Chapter 4.1 are used for this. For specific issues, the analytics system automatically digested the support bundles provided by the bug reporters. The system would then provide a link to the basic notebook in a comment in the bug tracker. Experts were encouraged to use the system to investigate the uses.

The relevant environment for all the cases are a cluster with one active and one standby controller, as well as a varying amount of switches.

5.1 Case A: Delay in Echo Replies and Disconnecting Switches

In this case study, a bug report points out that for a certain cluster the switches, which are supposed to stay connected via a tcp connection to the controllers, fail to timely reply to ECHO messages with an ECHO-REPLY message, and eventually drop their connection to the controllers.

While finding the reason for this behavior is the end goal of every analysis like this, a very important step in this case is finding out whether the problem lies with the switch, or the controller, since different experts exist for each domain.

Since the controllers collect time series data on various variables, the expert user uses the visual analytics system and displays the 99.9th percentile of the round-trip times between the controllers and the switches (see Figure 5.1), during the time-frame where the issues occurred, as inferred from error messages from the controller log.

The expert users points out that most switches have low round-trip times, but a few of the switches show significantly higher times. He concludes that, based on the fact that only a few switches experience slow round trip times, and that the round-trip times are

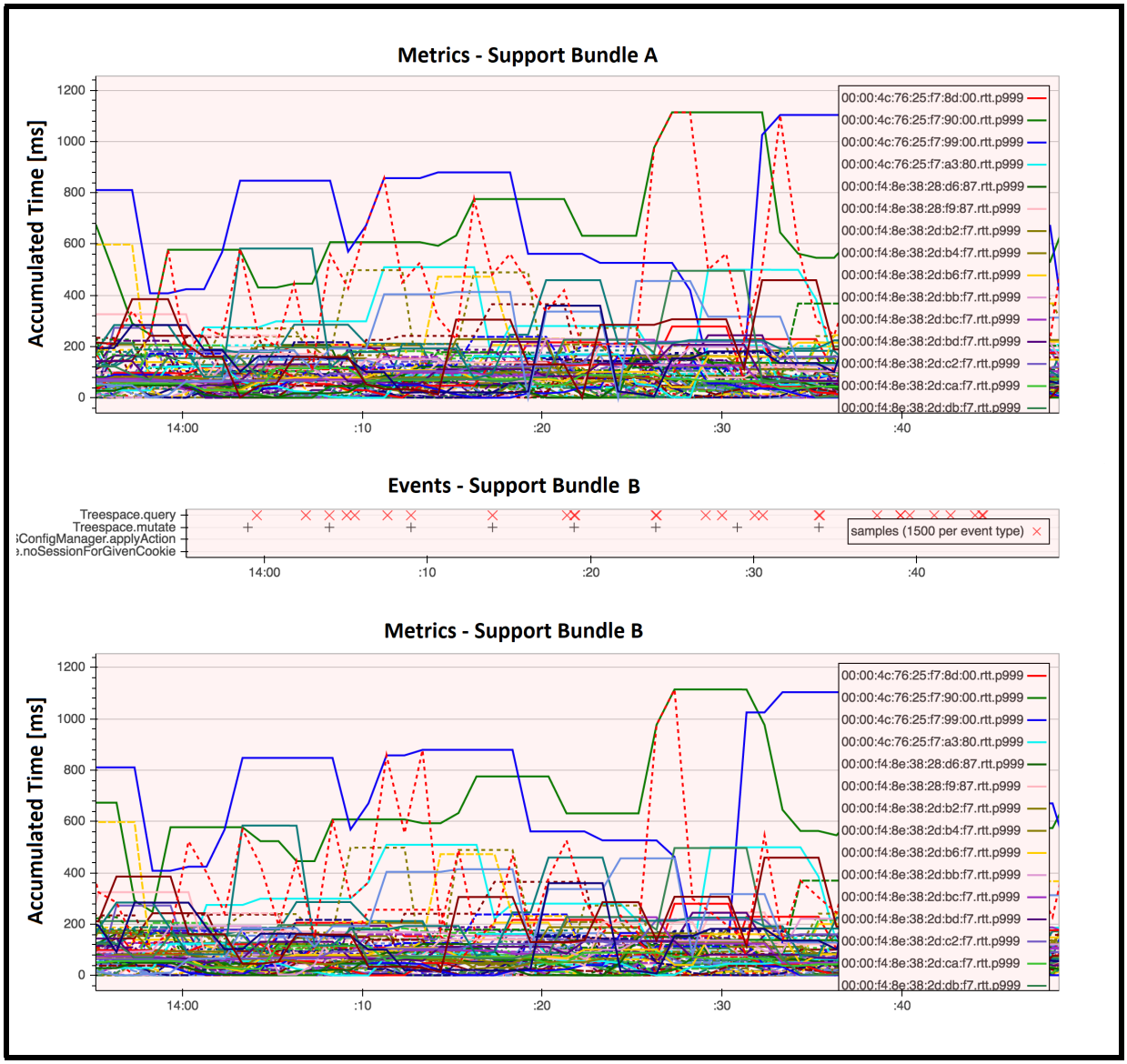

Figure 5.1: Round-trip times between switches and controllers over time

very similar independent of the controller, that the issue is most likely with the switches, and can therefore delegate this issues to the correct expert.

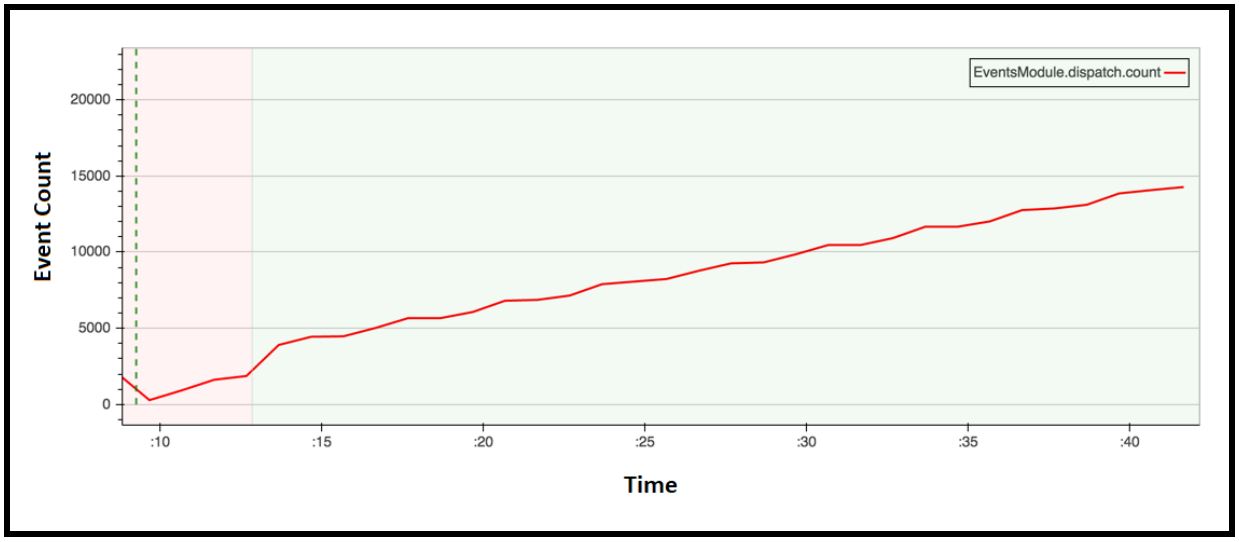

Figure 5.2: Number of events dispatched by controller

5.2 Case B: Controller Throws IllegalStateException During Config Change

In this case study, an exception is thrown in a controller when a config change event occurs. Expert users discuss in the bug tracker how this could happen, until one expert suggests that it could be related to a certain thread blocking. The data necessary to directly confirm or deny this assumption failed to record for the cluster and is thus not available. An expert user uses the visual analytics system to visualize the number of events that the controller has dispatched (see Figure 5.2), in the time window where the issue occurred.

The expert user argues that the thread presumed blocked could not be blocked, since the number of dispatched events is increasing steadily, and the thread responsible for the dispatch must be working. In addition to that, the highlighted areas transition from red to green, implying that the controller changed its state from standby to active.

While not responsible for the eventual discovery of the core issue, the visual analytics system helped to quickly check and reject an assumption that a certain thread was blocked, saving experts from expending time on finding other proof that the thread was active during the time period. Therefore, the visual analytics system helped in guiding the expert discussion to its eventual conclusion.

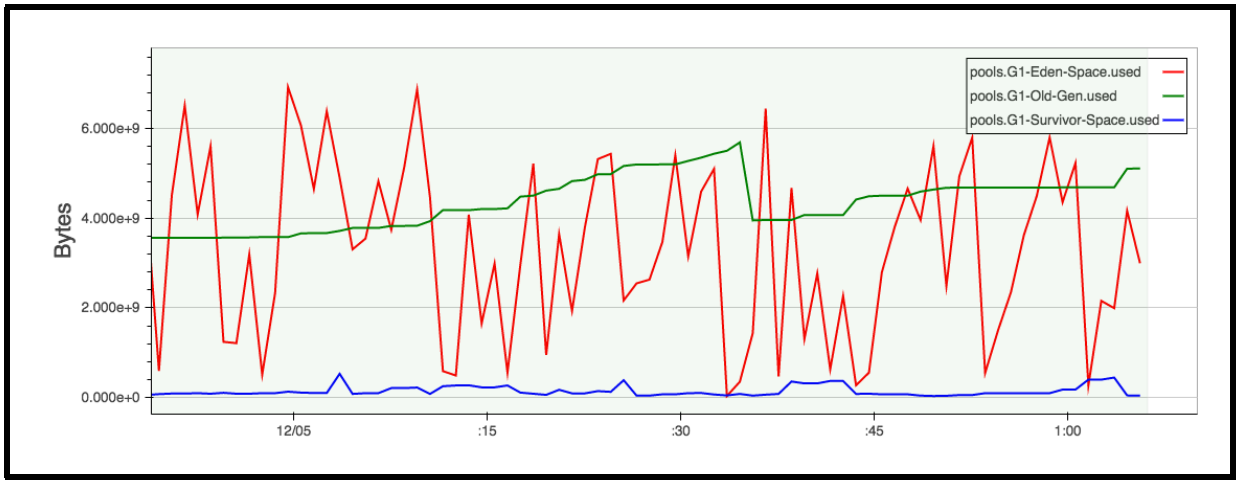

Figure 5.3: Garbage collector activity in controller over time

5.3 Case C: Cluster Takes Too Long to Start

In this case study, the time it takes to fully start a cluster and have it configured correctly is longer than expected. An expert user begins investigative work with the visual analytics system. The expert user suspects the controller to run out of memory, and visualizes the activity of the Java garbage collector for the controller (see Figure 5.3). While the expert user points out that the activity of the garbage collector is higher than anticipated, it cannot be the reason for the observed issues, as the system is never experiencing an actual memory bottleneck.

Having ruled out memory and the garbage collector at the controller as the issue, the expert user focuses his attention on the switches. He modifies the basic notebook in order to show a scatter plot of the average duration switches take to complete the startup. Figure 5.4 shows this scatter plot, where the x-axis represents the time axis (and when the respective switch started), while the y-axis shows how long the respective switch took to start. Since a mix of virtualized as well as physical switches was used, the expert users groups chooses a different color to represent each switch based on which of those two categories it falls under.

The expert user notes how the plot immediately shows how the KVM (virtual) switches take a significantly longer time to start than the physical switches, which start up in a mostly similar time frame. The expert user therefore concludes that the issue must be a performance problem related to the virtual switches in this cluster, and delegates the issue to the experts for that problem.

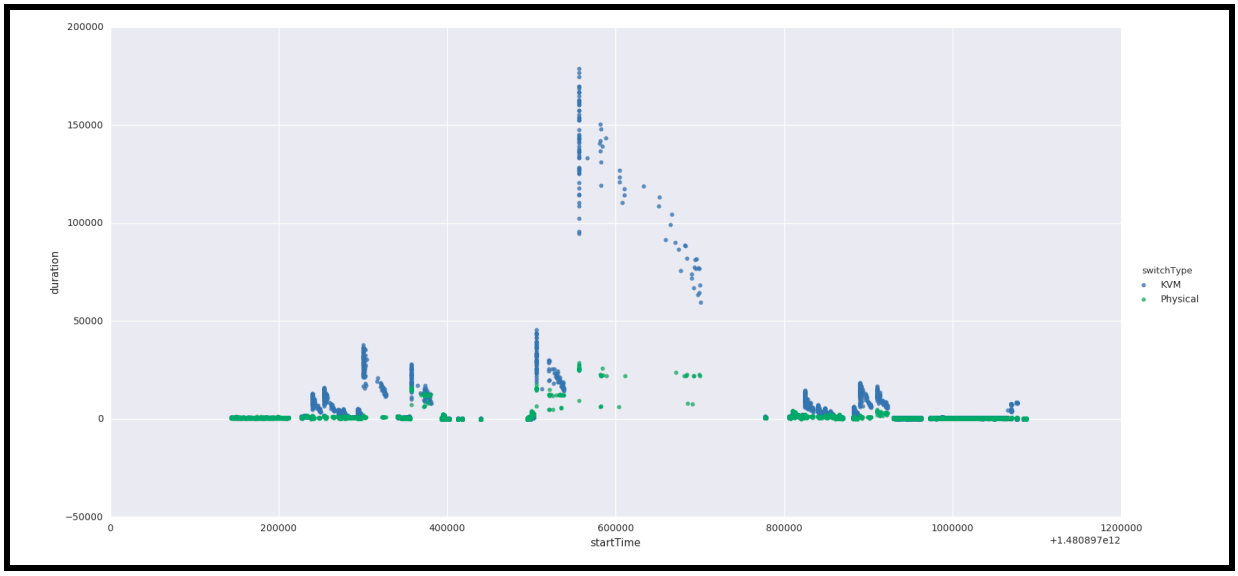

Figure 5.4: Startup duration of physical and virtual switches

Figure 5.5: Garbage collector activity in controller

5.4 Case D: Controller-Switch Connections Unstable During Bundle Creation

In this case study, a bug was reported on the bug tracker where the connections between controllers and switches would randomly fail while a support bundle was created. This serves as a good example for why the effects of collecting data from a cluster have to be considered for both a production system, as well as the visual analytics system.

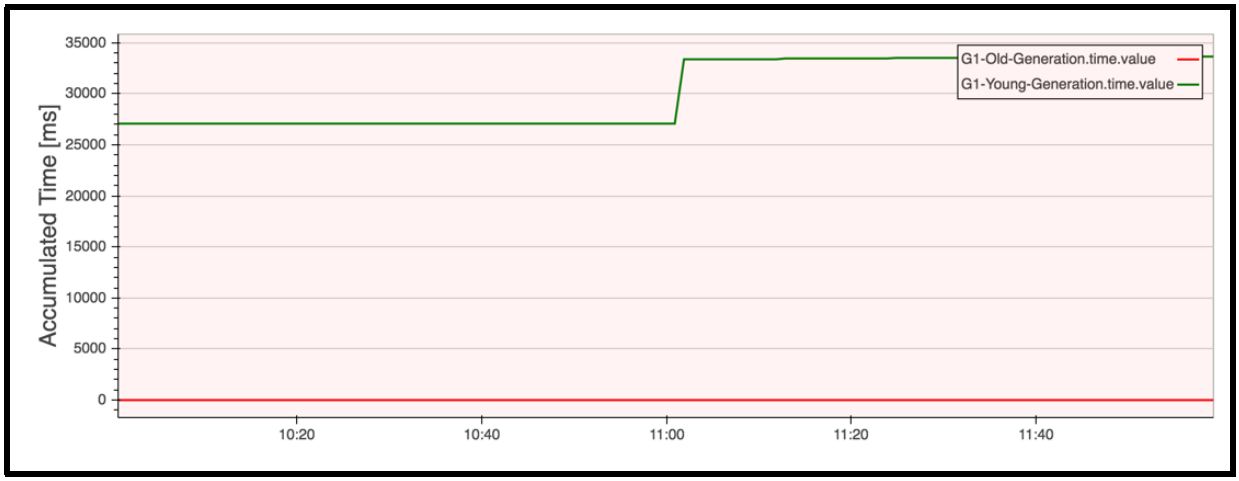

Figure 5.6: Garbage collector activity in controller over time

The expert user suspects that the memory on the controller could be the problem (similar to the assumption made for the issue in Section 5.3), and plots data about the garbage collector (see Figure 5.5). The expert user is quickly able to show that during the time where the switch connections become unstable, no relevant garbage collection activity took place, which guides the discussion between the experts into another direction.

Eventually, it is revealed that the I/O-intensive support bundle creation competed with the I/O activity of the controller, leading to a bottleneck within the controller. This caused the switches to disconnect randomly.

5.5 Case E: Switch Connections Unstable in Controller Due to Garbage Collection

In this example, switches would randomly loose and regain their connection to the standby controller. Just as with the case studies in Sections 5.3 and 5.4, an expert user wants to investigate whether memory could be an issue here. The expert user visualizes the garbage collection data (accumulative time spend executing garbage collection) again (see Figure 5.6), and notices that the garbage collection ran for 6 seconds uninterruptedly. The expert user argues that, while this is not ideal, its an isolated occurrence, and the garbage collection was done for the so-called young generation, and not the old generation (which, according to the expert user, means that it is unlikely to be the cause of the issue).

After having looked at the accumulative time spent on garbage collection, the expert user investigates whether the actual amount of used memory could have been problematic

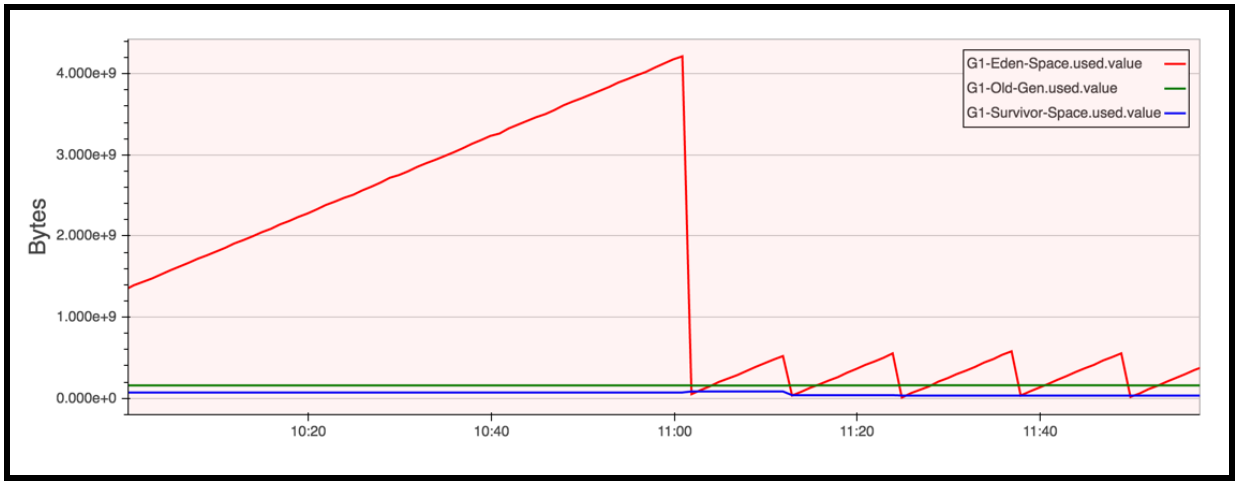

Figure 5.7: Memory usage in controller over time

(see Figure 5.7). Based on this visualization, the expert user is able to argue that, while memory usage was increasing linearly, the garbage collection successfully collected almost all of the used up memory. All following growth and collection periods were much shorter, meaning that the garbage collection was triggered much earlier. None of these observations seem suspicious to the expert user, prompting him to investigate other potential sources for the issue.

Suspecting that the problem might be related to threads (a fluctuation of threads could indicate that a thread stopped or that a number of threads could have been started erroneously), the expert user plots all thread-related information with the visual analytics system (see Figure 5.8). The visualization shows counters pertaining to the state of switches. Since the data indicates that nothing of interest happened at the time of the issue (around 11:00, where all lines are stable), the expert user rules out thread-related fluctuations as the root cause.

Wondering if structured event data (see Chapter 3.3) might yield any clues, the expert user visualizes which events have occurred while the switch connections became unstable (see Figure 5.9). The expert user selected the time frame where the issue occurred, and realizes that Treespace queries (database queries) are so common during that time frame that the visual analytics system is only showing samples as to not slow down the interactive visualization.

As many components of the controllers could be issuing database queries, the expert user uses the notebook to explore the *payload* of the respective structured events, which yields information about what part of the system is issuing this many requests (see Figure 5.10). An overwhelming majority of database requests are issued by three components, which are related. The expert user conclude that, while it cannot be concluded with

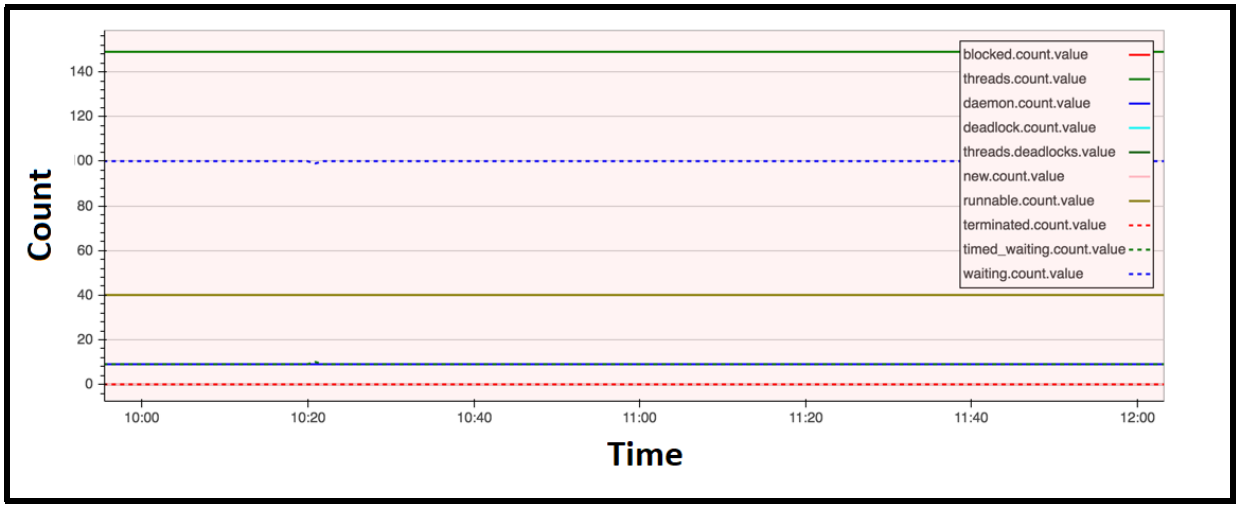

Figure 5.8: Thread information from the controller

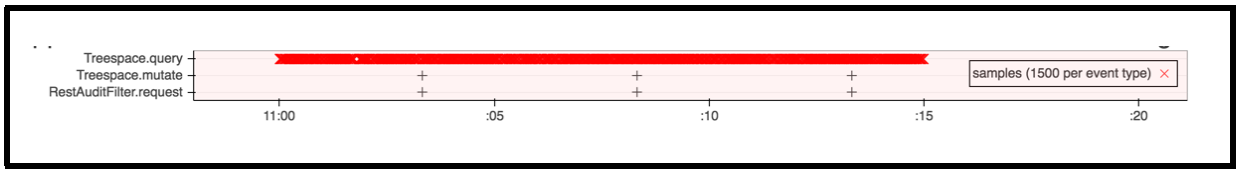

Figure 5.9: Structured event data collected by controller over time

absolute certainty that the underlying issue is a flood of database queries, it is highly likely, and an expert for the respective component should investigate why it is issuing these high amounts of database requests, even though the controller is in standby role.

After an improvement to the respective component, the problem did not resurface, and the issue was deemed solved.

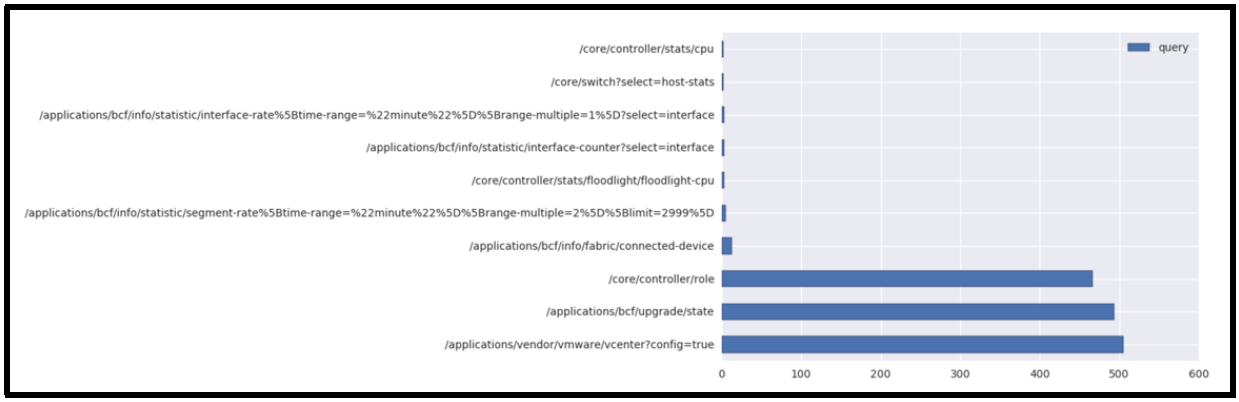

Figure 5.10: Distribution of database query events in controller

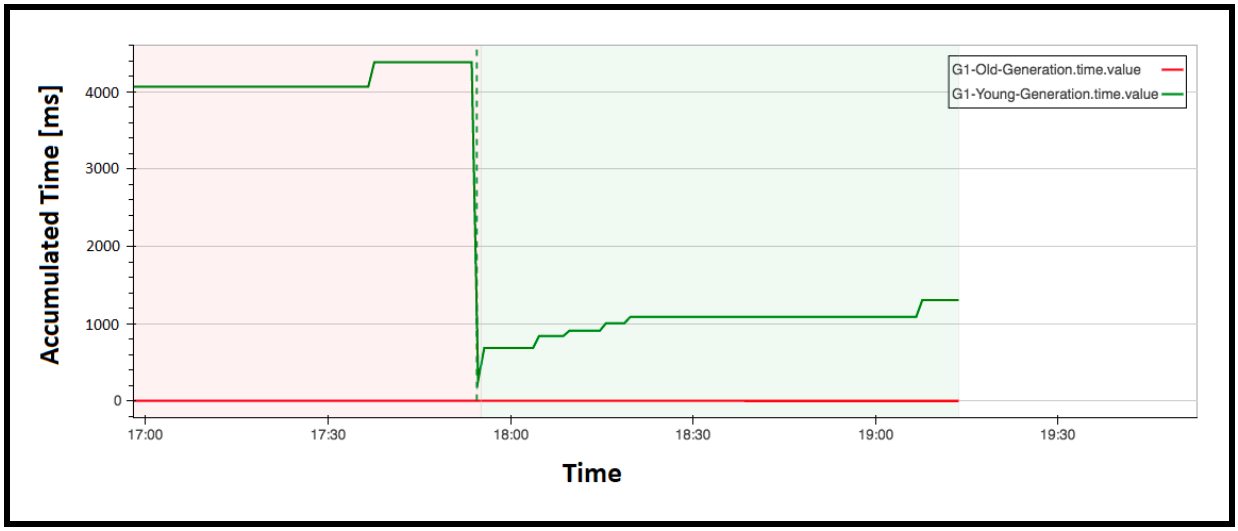

Figure 5.11: Garbage collector activity in controller

5.6 Case F: Some Switches Disconnect and Reconnect During Bundle Creation

In this case study, some of the switches disconnect, and then reconnect to the controller, but only when a support bundle is being created (similar to the case in Section 5.4). The expert user notes that switch disconnects have been caused by garbage collector activity in the controller before. The expert user explores the garbage collection data (see Figure 5.11), and remarks that there is no full garbage collection going on (as the old generation value does not change).

Another possible reason for a disconnect could be a CPU bottleneck, but the expert user argues, based on the visualization of CPU information (see Figure 5.12), that the controller has not been particularly busy.

The expert user explores the structured event data (see Figure 5.13), which show a number of exceptions, which correlate with other events. The expert user suggest investigating these exceptions to help find the underlying issue.

5.7 Summary

The case studies show that the visual analytics system can be introduced in an existing workflow for bug tracking and fixing. A reoccurring pattern observed in all presented

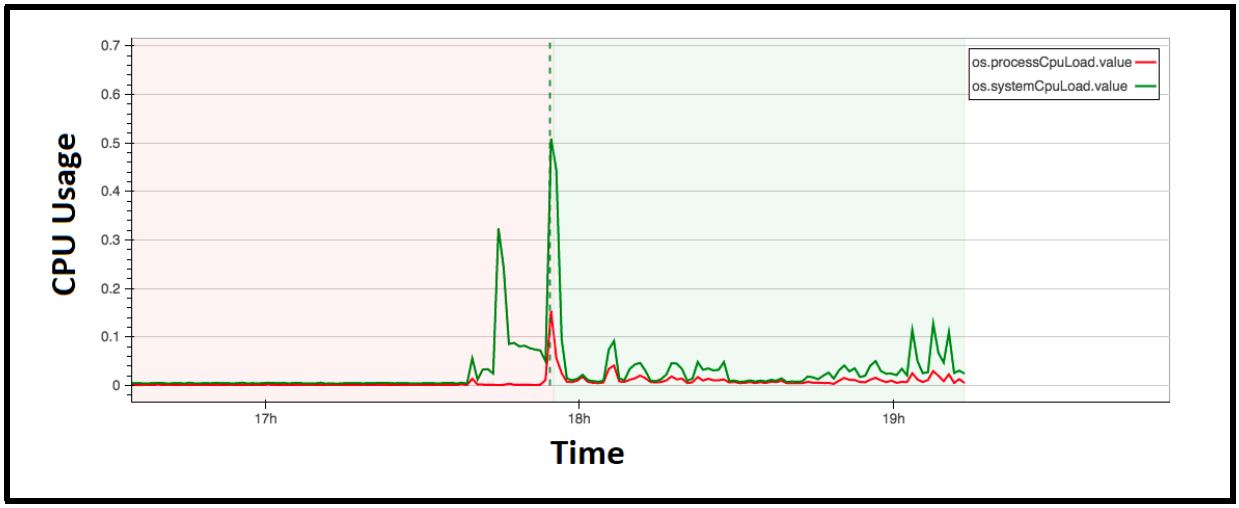

Figure 5.12: CPU usage for the controller

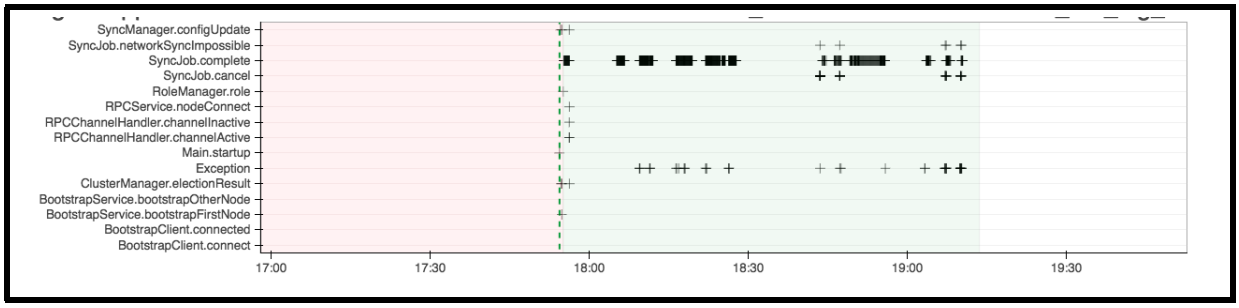

Figure 5.13: Structured event data collected by controller over time

cases is that in the vast majority of cases, the visual analytics system is used to test a hypothesis, as opposed to finding clues for an issue through random exploration.

Based on this realization, the visual analytics system was extended in such a way that visualizations which would often be created by the expert users are automatically inserted into the comments of each issue on the bug tracker. The idea behind this approach is to make expert users who have not used the system yet more aware of the system, and to provide them with a starting point for their own exploration.

The results of this approach are at the time of this writing not conclusive yet, but initial feedback has been positive.

6 Discussion and Limitations

In this chapter, we want to take a step back and take a critical look at the introduced visual analytics system. Thanks to the opportunity to evaluate the visual analytics system with real data provided by a data center networking company (see the case studies in Chapter 5), we will be able to draw conclusions based on the feedback of expert users.

As extensively explained in Chapter 3, and as witnessed in some of the presented case studies (see Chapter 5), the collection of data from a cluster can influence the cluster itself. However, this data is collected whether or not it is used for a visual analytics system. In the example of the data center networking company, the support bundles have always been used as the main means of finding and eradicating bugs that appear in the distributed system. A visual analytics system built on top of this data therefore is only another benefactor of this data.

The visual analytics system itself can also be set up in such a way that it does not interfere with any of the clusters, nor with the work flow of experts working on solving issues with the bug tracker. The visual analytics system interacts with the expert users via the discussion found in every bug tracking system (see Chapter 5). The visual analytics system posts links to itself to encourage expert users to explore the data of a particular issue. This is in contrast to approaches where a system is supposed to completely replace the current work flow.

Now we will discuss the key advantages and disadvantages of the system.

1. **Resources**

   The visual analytics system is web-based, and as such, users only need a web browser to access it. However, the system has to be hosted somewhere. Since this system will have to handle many data transformations and store and process terabytes of data, server-grade hardware should be used to provide a smooth experience for users. An advantage is that the expert users don't require the processing power needed to explore the data on their computers, which allows expert users to explore the data using commodity hardware.

2. **Preservation of history**

   While the visual analytics system did not play a role for resolving bugs in a bug tracker in the past, once it is employed and the insights generated are referenced in the discussion between experts, it becomes a part of the history of a specific issue. In case an issue is ever revisited, the insight generated by the visual analytics system should still be accessible. This has an effect on various things. Data has to be retained potentially indefinitely, and updates to the system should not negatively influence past insights. Storage space requirements can easily reach multiple terabytes in a month.

3. **New insights, new requirements**

   Once the visual analytics system is employed to visualize a certain set of data, it can inadvertently drive the demand for more data that can be visualized. The visual analytics system has to be built and deployed in such a way that it is scalable, as data density per unit of time will likely increase after the system has been deployed.

4. **New tools**

   The introduction of a visual analytics system vastly changes the way data is approached. Experts have to expend time to learn how to use such a system, and can be reluctant to switch away from a proven method of solving an issue. Expert users who used the system suggested a lead-by-example approach, where issues are solved with the help of the system to encourage other experts to try out the system themselves.

5. **Status quo**

   Solving issues with visual analytics in distributed software has not seen much use in the past. However, even relatively simple visualizations and interaction models are often sophisticated enough to change the way experts look at the data of such a system, since even simple visual tools contrast strongly with the mostly text and console-based traditional approach for issue troubleshooting.

6. **Intelligent behavior**

   So far, all the data digested by the visual analytics system has been observed independently, and experts suggested that taking advantage of all the combined observations could provide experts with more useful insights. An example for this would be the ability to detect similar patterns across bundles, which could allow the visual analytics system to point out similarities between issues. This idea will be explored in Chapter 7.

The case studies in Chapter 5 provided anecdotal evidence of the usefulness of the visual analytics system. The next section intends to provide an analysis which shows the advantages and disadvantages of traditional and visual analytics-based issue solving.

6.1 Time to Access Data

Given that the visual analytics system is set up to automatically digest data (as done for the case studies in Chapter 5), an interesting metric can be studied:

How much time does it take an expert user to finish his investigation?

For this evaluation, we assume an issue that has been reported to a bug tracker, and was assigned to an expert for investigation. The issue has a total of 4 support bundles, which measure 10GB each, for a total of 40GB. The extracted data occupies 100GB on the disk. We also assume that in total, 3 experts will be involved in solving the issue, and that the second and third expert become involved when their predecessor asks them for help after failing to identify the cause. Experts could be working off-site, and would need to retrieve the data fully in order to investigate it. The focus of this evaluation is to determine the variable factors that contribute to the total time it takes to solve an issue. It is also assumed that experts can investigate the data with similar speed using the traditional and the visual analytics approach, as the case studies suggest that experts were able to timely draw conclusions using the visual analytics system.

6.1.1 Traditional Approach

After the expert user is made aware of the issue, all data associated with the issue has to be retrieved by the expert, and extracted to his work environment. The time it takes the expert to retrieve the data from the server shall be known as $t_{download}$. After the data has been downloaded, it needs to be extracted. The extraction also takes some time, and shall be represented as $t_{extract}$. The total space in use for the issue is now 140 GB, with 40GB of support bundles represented as $s_{archived}$ and 100GB from the extracted data represented as s_{raw}.

Depending on the amount of experts that are involved in solving this issue (and assuming they are consulted by the previous expert whenever they are stuck and ask for help), we need to add a factor for the amount of experts working on the issue, which shall be referred to as $n_{experts}$. The total time it takes all experts combined to solve the issue will be known as t_{solve}.

In summary, the time it takes to find the cause of the issue is:

$t_{total} = (n_{experts} * (t_{download} + t_{extract})) + t_{solve}$

According to our scenario, we assume $n_{experts} = 3$. Assuming a 100 megabit connection from the experts (who may be working off-site) to the server with the data, 40 gigabyte of data would result in $t_{download} = 57\,minutes$.

The extraction of the data requires will be estimated at $t_{extract} = 3\,minutes$.

We will not assume any particular time to solve the issue with the traditional approach, and leave t_{solve} as a variable. This leaves us with the following formula:

$t_{total} = (3 * (57\,minutes + 3\,minutes)) + t_{solve}$

$\implies t_{total} = 3\,hours + t_{solve}$

$\implies t_{total} - t_{solve} = 3\,hours = t_{overhead}$

We conclude that for our scenario, the overhead caused by experts having to handle large amounts of data is 3 hours, and the overhead is linearly growing with the number of experts working on an issue.

The peak disk space needed is:

$s_{total} = n_{experts} * (s_{archived} + s_{raw})$

which, given the size of the archived and raw data leads us to:

$s_{total} = 3 * (40\,GB + 100\,GB)$

$\implies s_{total} = 420\,GB$

To summarize, the traditional approach to solve an issue is linearly related in both time and space requirements to the number of experts working on the issue. The time to resolve an issue has a constant element of t_{solve}.

6.1.2 Approach Using Visual Analytics

The general assumption is that the visual analytics system will retrieve the data of issues it encountered, process them, and then offer expert users the ability to explore the data related to an issue.

First, and just like with the traditional approach, the visual analytics system retrieves all data for the issue ($t_{download}$), and extracts them ($t_{extract}$). While it is likely that the server which the visual analytics system is hosted on has a much better connection to the server holding the data, and its performance in decompressing the archives

is superior to the expert users' personal computers, we will still assume the same conditions as encountered in the traditional approach, as to not distract from the factors that contribute in a more serious way to the total time needed.

Since the visual analytics system needs to preprocess the data for exploration ($t_{process}$), the formula as compared to the traditional approach slightly changes:

$t_{total} = t_{download} + t_{extract} + t_{process} + t_{solve}$

While a new variable was added to the formula as compared to the traditional approach, the factor for the number of experts working on the issue has been eliminated, as the number of experts working on the system is not relevant because the data has been centrally processed once, and can be explored by the experts with a browser with negligible amounts of data being transferred between the browsers and the server. For the sake of simplicity, we assume that it takes 60 minutes to digest the data. The exact number depends on the types of transformations applied to the raw data, among other factors.

This brings us to the following formula:

$t_{total} = 57\,minutes + 3\,minutes + 60\,minutes + t_{solve}$

$\implies t_{total} = 2\,hours + t_{solve}$

$\implies t_{total} - t_{solve} = 2\,hours = t_{overhead}$

In contrast to the traditional approach, the overhead is lower and the only relevant variable is the time it takes to solve the problem.

The space requirements are different as well, and are also no longer dependent on the number of experts working on an issue. However, data is transformed into different representation and stored for easier exploration, which causes additional storage overhead (s_{db}). We will assume a space requirement as big as the original raw data (s_{db}).

This leads us to the following:

$s_{total} = s_{archived} + s_{raw} + s_{db}$

Which, similar to the formula for time, has an added variable for the processed data size, but is similarly lacking a linear relation with the number of experts, as the data has to only be stored once on a central server. The peak space requirement in our example therefore is:

$s_{total} = 40\,GB + 100\,GB + +100\,GB$

$\implies s_{total} = 240\,GB$

6.1.3 Conclusion

The example was chosen to highlight that, while the visual analytics system has an initial overhead, it scales much better as more experts work on a given issue.

An example where the traditional approach would prove superior is an issue of high importance (meaning it needs to be solved as quickly as possible). In that case, the overhead introduced by the need for the visual analytics system to preprocess its data is high enough to potentially deter from its usage.

However, the majority of issues are not so critical that an expert would drop whatever he or she was doing and immediately start working on it. More likely, an issue is assigned to a particular expert, who will take a look at it hours, potentially even days later. During this time, the visual analytics system is already processing the data, meaning that by the time the expert investigates the issue, the overhead has been made irrelevant, and the expert can immediately solve the issue. if this is the case, the formula for the time needed to solve the issue is simply:

$t_{total} = t_{solve}$

In case an expert needs access to the raw data because he is unable to investigate using the visual analytics tool alone, the expert can still explore the data with the traditional approach. Alternatively, the already extracted data could be selectively retrieved from the server where the visual analytics system is running, assuming that it retains the raw data in its form. While transferring uncompressed data might cause more network traffic than transferring the compressed version of a particular file, the overall traffic should still be highly reduced, as data can be retrieved selectively (as opposed to retrieving the full support bundle).

In addition to that, the volume of data may simply be impractical to explore locally on a computer. Running on a server, the visual analytics system has access to very large amounts of computational power, disk space, and memory.

If we take into account:

1. A growing number of experts working on a particular issue, e.g. in a company that is growing
2. A never-ending increase in data volume
3. The possibility to take advantage of centrally stored data (e.g. useful for data mining, machine learning)

We come to the conclusion that centralizing the visual analytics system in favor of experts analyzing issues in a distributed and local manner reduces the overhead of each expert, and allows experts to do complex data analysis and exploration using only commodity hardware.

7 Conclusion and Future Work

The goal of this master thesis was to apply visual analytics to the domain of problem solving in distributed systems. After putting the topic in the context of related work, the data model and challenges related to the data were explored. A visual analytics system was introduced which takes inhomogeneous data from a multitude of sources and transforms it into more suitable formats for quick access. Expert users can then use this system to interactively explore the data as well as manipulate the data and visualization to their liking.

The system was used by expert users in a data center networking company to help isolate the root cause for issues from a bug tracker. In all cases presented, the system assisted experts in their problem analysis to some extent. While many additional concepts could be employed to assist the experts even more, the presented solution could provide helpful feedback with relatively simple visualization and interaction techniques already.

In the following sections, we will discuss topics which could be explored in future work.

Duplicate Detection

According to expert users, a lot of time can be wasted on determining if two issues in a bug tracker have the same root cause. While the visual analytics system already supports visual comparison between different support bundles, the expert user has no knowledge about what other support bundle might express signs of the same bug, since a bug in code might manifest itself in different ways, and it is hard to quickly determine if two distinct bugs are present, or only one. Since all the data for all issues is available to the visual analytics system, pattern recognition or machine learning techniques could be used to find similarities between issues (e.g. similar exceptions, or similar patterns in time-series data). If a new issue appears and analysis reveals a similarity between the issue and another issue, an expert user could be notified when exploring the data of the new issue, and explore the similarities between the two issues.

Continuous Digestion

Currently, all data is processed in batch after it has been collected. If instead clusters would stream all their data continuously to the visual analytics system, the need for support bundles could be eliminated. The visual analytics system would then be in the position to determine what data is retained, and for how long.

There are many advantages, but also disadvantages to this approach. If data is continuously streamed to the visual analytics system, then a failure of the analytics system would cause the loss of data. On the other hand, streaming data away from a node to a central location that can easily be backed up reduces the risk and allows debugging in case that a node suffers a catastrophic failure.

From a security standpoint, streaming all kinds of data constantly from a node in a production environment is problematic or even impossible altogether, as nodes may run in isolation with no way to communicate with a centralized analytics server. Nevertheless, exploring possible hybrid solutions would be a possibility for future work.

Automatic Type Inference

Collected data for support bundles include the logs of the distributed system itself, as well as all logs of other applications running on the system. While most of these logs contain timestamps, the format of these logs and timestamps may differ. Manually defining which file in which directory uses which log format would lead to a very maintenance-intensive system. Automatic detection of log formats (or absence thereof) in the context of distributed systems could be a topic for future work.

Easier Expert User Feedback

Currently, changes to how the raw data is processed and stored are not as easy as they could be from the user interface of the visual analytics system. The visual analytics system could be extended in a way that makes data transformation easier, and provide the expert user with an estimate for when the reprocessed data will be available if the transformation will take some time to completed.

Closing Words

The results of this thesis suggest that visual analytics in the domain of distributed systems issue analysis can provide an advantage over traditional–often text and console-based investigation methods–as long as challenges associated with visualization and interaction are properly taken into account.

Acknowledgement

I would like to thank Michael Burch, Andreas Wundsam and Big Switch Networks, Inc. for their continuous support of this thesis.

Bibliography

[AMST11] W. Aigner, S. Miksch, H. Schumann, C. Tominski. *Visualization of Time-Oriented Data*. 1st. Springer Publishing Company, Incorporated, 2011. ISBN: 0857290789, 9780857290786 (cit. on p. 13).

[Bok17] Bokeh. *Bokeh*. 2017. URL: https://bokeh.pydata.org/en/latest/ (cit. on p. 38).

[Dro17] Dropwizard. *Dropwizard Production-ready, out of the box.* 2017. URL: http://www.dropwizard.io/1.1.4/docs/ (cit. on p. 28).

[DSP+17] F. Du, B. Shneiderman, C. Plaisant, S. Malik, A. Perer. "Coping with Volume and Variety in Temporal Event Sequences: Strategies for Sharpening Analytic Focus." In: *IEEE Transactions on Visualization and Computer Graphics* 23.6 (June 2017), pp. 1636–1649. ISSN: 1077-2626. DOI: 10.1109/TVCG.2016.2539960 (cit. on p. 13).

[Ela17] Elasticsearch. *Open Source Search and Analytics Elasticsearch*. 2017. URL: https://www.elastic.co/ (cit. on p. 30).

[FDPH16] M. Feng, C. Deng, E. Peck, L. Harrison. "HindSight: Encouraging Exploration through Direct Encoding of Personal Interaction History." In: *IEEE Transactions on Visualization and Computer Graphics* (2016) (cit. on p. 14).

[Hol13] A. Holzinger. "Human-Computer Interaction and Knowledge Discovery (HCI-KDD): What Is the Benefit of Bringing Those Two Fields to Work Together?" In: *Availability, Reliability, and Security in Information Systems and HCI: IFIP WG 8.4, 8.9, TC 5 International Cross-Domain Conference, CD-ARES 2013, Regensburg, Germany, September 2-6, 2013. Proceedings*. Ed. by A. Cuzzocrea, C. Kittl, D. E. Simos, E. Weippl, L. Xu. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 319–328. ISBN: 978-3-642-40511-2. DOI: 10.1007/978-3-642-40511-2_22. URL: https://doi.org/10.1007/978-3-642-40511-2_22 (cit. on p. 13).

[Inf17] InfluxDB. *InfluxData (InfluxDB) | Time Series Database Monitoring and Analytics*. 2017. URL: https://www.influxdata.com/ (cit. on p. 30).

[JSO13] JSON. *The JSON Data Interchange Format*. Tech. rep. Standard ECMA-404 1st Edition / October 2013. ECMA, Oct. 2013. URL: http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf (cit. on p. 28).

[Jup17] Jupyter. *jupyter*. 2017. URL: http://jupyter.org/ (cit. on p. 44).

[KH13] J. Kehrer, H. Hauser. "Visualization and Visual Analysis of Multifaceted Scientific Data: A Survey." In: *IEEE Transactions on Visualization and Computer Graphics* 19.3 (Mar. 2013), pp. 495–513. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.110. URL: http://dx.doi.org/10.1109/TVCG.2012.110 (cit. on p. 13).

[KKE10] E. D. Keim, J. Kohlhammer, G. Ellis. *Mastering the Information Age: Solving Problems with Visual Analytics, Eurographics Association*. 2010 (cit. on p. 13).

[KMS+08] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, H. Ziegler. "Visual Data Mining." In: ed. by S. J. Simoff, M. H. Böhlen, A. Mazeika. Berlin, Heidelberg: Springer-Verlag, 2008. Chap. Visual Analytics: Scope and Challenges, pp. 76–90. ISBN: 978-3-540-71079-0. DOI: 10.1007/978-3-540-71080-6_6. URL: http://dx.doi.org/10.1007/978-3-540-71080-6_6 (cit. on pp. 13, 31).

[Pan17] Pandas. *pandas.DataFrame*. 2017. URL: https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.html (cit. on pp. 31, 36).

[RKW+06] P. Reynolds, C. Killian, J. L. Wiener, J. C. Mogul, M. A. Shah, A. Vahdat. "Pip: Detecting the Unexpected in Distributed Systems." In: *Proceedings of the 3rd Conference on Networked Systems Design & Implementation - Volume 3*. NSDI'06. San Jose, CA: USENIX Association, 2006, pp. 9–9. URL: http://dl.acm.org/citation.cfm?id=1267680.1267689 (cit. on p. 14).

[SBC+14] L. D. Silvestro, M. Burch, M. Caccamo, D. Weiskopf, F. Beck, G. Gallo. "Visual Analysis of Time-Dependent Multivariate Data from Dairy Farming Industry." In: *Proceedings of the 5th International Conference on Information Visualization Theory and Applications, IVAPP 2014, Lisbon, Portugal, 5-8 January, 2014*. 2014, pp. 99–106. DOI: 10.5220/0004652600990106. URL: https://doi.org/10.5220/0004652600990106 (cit. on p. 13).

[SSK+16] D. Sacha, H. Senaratne, B. C. Kwon, G. P. Ellis, D. A. Keim. "The Role of Uncertainty, Awareness, and Trust in Visual Analytics." In: *IEEE Trans. Vis. Comput. Graph.* 22.1 (2016), pp. 240–249. DOI: 10.1109/TVCG.2015.2467591. URL: https://doi.org/10.1109/TVCG.2015.2467591 (cit. on p. 14).

[SWLL13] G. Sun, Y. Wu, R. Liang, S.-X. Liu. “A Survey of Visual Analytics Techniques and Applications: State-of-the-Art Research and Future Challenges.” In: *J. Comput. Sci. Technol.* 28.5 (2013), pp. 852–867. URL: http://dblp.uni-trier.de/db/journals/jcst/jcst28.html#SunWLL13 (cit. on p. 13).

[TLCV15] C.-W. Tsai, C.-F. Lai, H.-C. Chao, A. V. Vasilakos. “Big data analytics: a survey.” In: *Journal of Big Data* 2.1 (Oct. 2015), p. 21. ISSN: 2196-1115. DOI: 10.1186/s40537-015-0030-3. URL: https://doi.org/10.1186/s40537-015-0030-3 (cit. on p. 13).

[ZSB+12] L. Zhang, A. Stoffel, M. Behrisch, S. Mittelstadt, T. Schreck, R. Pompl, S. Weber, H. Last, D. Keim. “Visual analytics for the big data era 2014; A comparative review of state of the art commercial systems.” In: *2012 IEEE Conference on Visual Analytics Science and Technology (VAST)*. Oct. 2012, pp. 173–182. DOI: 10.1109/VAST.2012.6400554 (cit. on p. 14).

All links were last followed on October 4th, 2017.

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

Sunnyvale, 4.10.2017, A. Kubzdela

place, date, signature