Identifying Relevant Data Center Telemetry Using Change Point Detection

Daniel Alves  
University of California, Santa Cruz  
dalves@ucsc.edu

Katia Oblaczka  
University of California, Santa Cruz  
katia@soe.ucsc.edu

Rick Lindberg  
Samsung SDS Research America  
r.lindberg@samsung.com

Abstract—In this paper, we focus on the problem of data center performance monitoring, more specifically, how to manage the large volume of data generated by data center telemetry tools. We propose a framework that uses Change Point Detection (CPD) to identify sources of useful telemetry and based on that information, filters incoming telemetry data in real-time as the data center operates. To evaluate our proposed CPD-based telemetry triage framework, we conducted experiments using a small emulated data center driven by different workloads. We also report results from experiments with telemetry data collected from a privately-owned, commercial, multi-tenant data center. Preliminary experimental results show that our CPD-based tool can filter out significant amounts of irrelevant telemetry while preserving most relevant telemetry sources.

Index Terms—data center, telemetry, feature selection, change point detection

I. INTRODUCTION

Data centers have become essential to support the information technology and computing needs of our society at large. As such, ensuring their proper operation is critical and has attracted considerable attention from both researchers and practitioners. In particular, there is significant interest in data center monitoring. More specifically, systems as complex as data centers generate considerable amounts of telemetry data and filtering out useful information becomes quite challenging.

In this work we propose a framework that uses Change Point Detection (CPD) [1] to identify relevant sources of data center telemetry in real time as the data center operates. Based on this information, our framework selects telemetry data being collected that should be monitored by data center operations. Our contribution is a systematic and scalable approach to identify in real time relevant data center telemetry sources and, as the data center operates, filter telemetry information based on what has been identified as relevant. We test our proposal on an experimental computing cluster using different workloads, including telemetry from a private data center.

II. BACKGROUND AND RELATED WORK

Data center performance must be continuously monitored in order to guarantee that data centers are operating adequately and, when it is not the case, identify indicators of performance degradation or resource wastage for timely diagnosis and response. However, monitoring modern data centers is quite challenging due to their scale and complexity [2]. This has prompted multiple efforts focusing on developing techniques for improved data center monitoring [3]–[6], [6]–[8].

Some such efforts deserve special attention. Jehangiri et al. [9] present a system for anomaly detection that identifies metrics responsible for Service Level Objective (SLO) violations and Xue et al. [10] propose a neural network based framework for prediction of future loads with feature selection provided by auto-correlation. Medel et al. [11] have also developed a reference neural network approach to characterize the behavior of Kubernetes systems for resource management and application design. Baig et al. [7] propose a method for reducing and reconstructing telemetry data from data centers with Markov Chain Models. Abuzaid et al. [8] present MacroBase, an engine for highlighting important events when monitoring large amounts of data. It focuses on event detection by analyzing data streams and presenting related information to users. Once relevant telemetry is identified, data center operations can use them for detailed diagnosis and/or future analysis. To the best of our knowledge, our work is the first to propose a method for filtering data center telemetry that uses CPD.

III. PROPOSED APPROACH

We propose a new framework based on Change Point Detection (CPD) [1] for identification of relevant sources of telemetry (e.g., server CPU usage, number of requests received by hosted web servers) that need to be closely monitored. The intuition behind using a CPD based approach is that we are interested in identifying metrics that reflect notable changes in data center behavior as the data center operates. To this end, we need to select metrics that themselves exhibit significant behavior changes. It is worth mentioning that in order to validate the proposed telemetry triage approach, we compare changes in the behavior of the metrics themselves against ground truth, i.e., the events that have generated those changes.

Our telemetry filtering framework consists of two main components, namely: the catalog builder and the filter. The former builds a catalog of relevant features out of observed measurements with labelled events of interest, effectively training the system to recognize those, while the latter uses that catalog to filter out non relevant telemetry sources from new incoming telemetry data. In the remainder of this section, we describe each component of our telemetry triage framework in detail.
A. Catalog Builder

The catalog building step of the proposed framework includes mechanisms to: (1) select relevant telemetry, which is described by Algorithm 1, and, (2) based on the metrics selected, build the catalog (Algorithm 2).

**Relevant Metric Selection:** Metrics relevant to an event are selected by comparing variations in their behavior over time to the observed events (ground truth). First the variations in behavior are identified through the application of Change Point Detection (CPD). Then we have two sets of points, one describing the occurrence of events, the other containing all identified changes. We compare both sets using the Hausdorff distance [12] (Algorithm 1). The next step is to normalize the measured distance such that a score of 1 indicates the smallest possible distance, and a score of 0, the maximum distance. Then we rank measurements according to this score, and use a cutoff score to select the most relevant metrics, where the cutoff score is a parameter of the telemetry triage mechanism.

**Algorithm 1** Relevance Selection

\[
O \leftarrow \text{original set of time series} \\
R \leftarrow \text{set of relevant time series (empty)} \\
E \leftarrow \text{set of timestamps for events} \\
S \leftarrow \text{set of scores for time series (empty)} \\
\ell \leftarrow \text{threshold for selection} \\
\text{for all time series } ts \in O \text{ do} \\
\quad CP \leftarrow \text{set of change points detected in } ts \\
\quad h_{ts} \leftarrow \text{Hausdorff}(CP, E) \\
\quad \text{add } h_{ts} \text{ to } S \\
\text{end for} \\
\text{normalize scores in } S \text{ so that 1 is best possible fitting and 0 is the worst} \\
R \leftarrow \{ts|h_{ts} \in S, h_{ts} \geq \ell\}
\]

**Change Point Detection:** In the context of statistical analysis of time series, Change Point Detection (CPD) techniques try to identify changes in the statistical behavior of the observations [1]. That is, for a time series \( t \) containing points \( \{x_1, x_2, \ldots, x_n\} \), a change point \( c \) is a point \( c = x_i \) such that the sets of points \( \{x_1, x_2, \ldots, x_i\} \) and \( \{x_{i+1}, x_{i+2}, \ldots, x_n\} \) follow distinct statistical distributions.

While there are a number of CPD techniques [13], our approach adopts the Pruned Exact Linear Time approach [14] (PELT). One of its parameters is a distance function to select candidate change points from the time series; it also uses a fixed penalty parameter \( \beta \) which indicates the cost of each additional change point selected. We opted for a univariate approach as the goal is to be able to evaluate numerous time series in isolation and multivariate approaches would require further techniques to prune possible candidates. For our experiments, we used \( \beta = 4 \) and adopted a Mahalanobis [15] based distance function, both determined empirically.

**Catalog Builder:** To build the catalog we use labelling information available from collected data center measurements. The purpose of the labels is to identify the type of metric being collected and provide additional identification information, such as the server’s IP address. The catalog keeps information on which types of metrics the catalog builder considered relevant by archiving labels and associated values describing those metrics.

We can then process newly obtained telemetry data with this information to filter out metrics that do not show indications of changing in response to the events that we are interested in monitoring. Note that, as the data center operates, the relevance of telemetry information being collected may change, which the catalog builder must capture.

**Algorithm 2** Catalog Builder

\[
R \leftarrow \text{set of relevant time series} \\
C \leftarrow \text{catalog (empty)} \\
\text{for all time series } ts \in R \text{ do} \\
\quad t \leftarrow \text{type of } ts \\
\quad C \leftarrow C \cup \{t\} \\
\quad \text{for all pairs label-value } (l, v) \text{ of } ts \text{ do} \\
\quad \quad C \leftarrow C \cup \{(t, l, v)\} \\
\text{end for} \\
\text{end for}
\]

**B. Filter**

The filtering step of our framework relies on the telemetry sources identified as relevant during catalog building. Those were sources that, when we analysed them using change point detection, showed variation in behavior that matches the events under investigation. We can then use their identifying labels, i.e., their metric id, to select which time series we should continue to monitor.

However, considering all labels may render the filtering process excessively selective. To address that, we opted for two different approaches: a “simple” approach considering only a subset of labels describing the type of measurement collect, and a “full” approach that consider all labels. Both approaches are described in Algorithm 3.

IV. Experimental Methodology

In this section, we describe our experimental methodology as well as the experiments we conducted to evaluate our CPD-based data center telemetry triage framework. We ran experiments using data collected from a test bed under our control with either synthetic or trace-based activity, and from a privately-owned, commercial, multi-tenant data center.

Our experiments consisted of three phases, namely: (1) testbed setup and telemetry generation; (2) telemetry collection and relevance identification; and (3) telemetry filtering.

For our testbed experiments we deploy a cluster of five nodes managed through Kubernetes [16] on Chameleon [17]. The cluster is composed of an orchestrator, three servers and a data storage node. We use Prometheus [18] for aggregation and collection of telemetry measurements. These experiments are driven by two different workloads, a synthetic one generated from sending requests designed to stress different parts of
Algorithm 3 Filter

\begin{align*}
O & \leftarrow \text{original set of time series} \\
C & \leftarrow \text{catalog} \\
F & \leftarrow \text{filtered set of time series (empty)} \\
\text{for all time series } ts \in O & \text{ do} \\
& t \leftarrow \text{type of } ts \\
& \text{if } t \notin C \text{ then} \\
& \quad \text{continue} \\
& \quad \text{if using “full” method then} \\
& \quad \quad \text{for all pairs label-value } (l, v) \text{ of } ts \text{ do} \\
& \quad \quad \quad \text{if } (t, l, v) \notin C \text{ then} \\
& \quad \quad \quad \quad \text{continue} \\
& \quad \quad \text{end if} \\
& \quad \text{end for} \\
& \text{end if} \\
& F \leftarrow F \cup \{ts\} \\
\text{end for}
\end{align*}

We then use this catalog to filter a similar set of telemetry collected at a different time.

After this filtering process, we counted how many time series were discarded from the original total, as well as how many were discarded and would have been considered relevant if we had the corresponding ground truth. The purpose of the first metric is measuring how much we save by filtering out irrelevant telemetry sources, while the second metric shows us the loss of relevant telemetry incurred by the filtering.

### Table I: Filtering results

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Filtering Type</th>
<th>Score</th>
<th>Total Filtering</th>
<th>Relevant Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>synth1</td>
<td>simple</td>
<td>0.7</td>
<td>0.73</td>
<td>0.16</td>
</tr>
<tr>
<td>synth1</td>
<td>simple</td>
<td>0.8</td>
<td>0.98</td>
<td>0.79</td>
</tr>
<tr>
<td>synth1</td>
<td>full</td>
<td>0.9</td>
<td>0.99</td>
<td>0.87</td>
</tr>
<tr>
<td>synth1</td>
<td>full</td>
<td>0.7</td>
<td>0.94</td>
<td>0.24</td>
</tr>
<tr>
<td>synth1</td>
<td>full</td>
<td>0.8</td>
<td>0.99</td>
<td>0.85</td>
</tr>
<tr>
<td>synth1</td>
<td>full</td>
<td>0.9</td>
<td>1.00</td>
<td>0.91</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.7</td>
<td>0.68</td>
<td>0.14</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.8</td>
<td>0.97</td>
<td>0.86</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.9</td>
<td>0.99</td>
<td>0.94</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.7</td>
<td>0.70</td>
<td>0.15</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.8</td>
<td>0.97</td>
<td>0.87</td>
</tr>
<tr>
<td>synth2</td>
<td>full</td>
<td>0.9</td>
<td>0.99</td>
<td>0.94</td>
</tr>
<tr>
<td>wikibench1</td>
<td>simple</td>
<td>0.7</td>
<td>0.36</td>
<td>0.05</td>
</tr>
<tr>
<td>wikibench1</td>
<td>simple</td>
<td>0.8</td>
<td>0.39</td>
<td>0.04</td>
</tr>
<tr>
<td>wikibench1</td>
<td>full</td>
<td>0.9</td>
<td>0.46</td>
<td>0.02</td>
</tr>
<tr>
<td>wikibench1</td>
<td>full</td>
<td>0.7</td>
<td>0.83</td>
<td>0.56</td>
</tr>
<tr>
<td>wikibench1</td>
<td>full</td>
<td>0.8</td>
<td>0.92</td>
<td>0.74</td>
</tr>
<tr>
<td>wikibench1</td>
<td>full</td>
<td>0.9</td>
<td>0.99</td>
<td>1.00</td>
</tr>
<tr>
<td>wikibench2</td>
<td>simple</td>
<td>0.7</td>
<td>0.10</td>
<td>0.08</td>
</tr>
<tr>
<td>wikibench2</td>
<td>simple</td>
<td>0.8</td>
<td>0.22</td>
<td>0.10</td>
</tr>
<tr>
<td>wikibench2</td>
<td>simple</td>
<td>0.9</td>
<td>0.54</td>
<td>0.06</td>
</tr>
<tr>
<td>wikibench2</td>
<td>full</td>
<td>0.7</td>
<td>0.50</td>
<td>0.23</td>
</tr>
<tr>
<td>wikibench2</td>
<td>full</td>
<td>0.8</td>
<td>0.61</td>
<td>0.63</td>
</tr>
<tr>
<td>wikibench2</td>
<td>full</td>
<td>0.9</td>
<td>0.93</td>
<td>1.00</td>
</tr>
<tr>
<td>private</td>
<td>simple</td>
<td>0.7</td>
<td>0.62</td>
<td>0.21</td>
</tr>
<tr>
<td>private</td>
<td>simple</td>
<td>0.8</td>
<td>0.74</td>
<td>0.32</td>
</tr>
<tr>
<td>private</td>
<td>simple</td>
<td>0.9</td>
<td>0.94</td>
<td>0.53</td>
</tr>
</tbody>
</table>

Our experimental results provide insight into the effectiveness of our CPD-based telemetry triage approach as a way to filter out irrelevant data center telemetry sources as well as how to adjust the CPD tool’s parameters for effective filtering. Regarding our tool’s effectiveness, we can identify configurations that filter out between 40% to 80% of total telemetry source while preserving between 70% and 90% of relevant telemetry. The type of workload seems to be one of the main factor influencing results due to variation of behavior between workloads. As for the type of filtering and threshold score, simple and 0.7 seems to provide adequate results for most workloads.

### VI. Concluding Remarks and Future Work

This paper outlined a framework for filtering data center telemetry data using Change Point Detection. We described our approach as well as how it was implemented and evaluated using both an emulated data center driven by different workloads as well as telemetry collected from a privately-owned commercial data center. Preliminary results show that our tool can filter out significant amounts of irrelevant telemetry while preserving most relevant telemetry sources. Among open questions and future work directions, we highlight: (1) the need for a reliable method to establish ground-truth for our framework; (2) investigation of how to best set parameters used by our framework; (3) investigation of a multivariate CPD approach.

### References


