Identifying Relevant Data Center Telemetry Using Change Point Detection

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

Katia Obraczka
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].

This work was partially supported by CAPES (Coordination for the Improvement of Higher Education Personnel - Brazil) and by the US National Science Foundation under project CNS 1321151.

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 severs) 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

```
$O \leftarrow$ original set of time series
$R \leftarrow$ set of relevant time series (empty)
$E \leftarrow$ set of timestamps for events
$S \leftarrow$ set of scores for time series (empty)
$\ell \leftarrow$ threshold for selection
for all time series $ts \in O$ do
    $CP \leftarrow$ set of change points detected in $ts$
    $h_{ts} \leftarrow$Hausdorff$(CP, E)$
    add $h_{ts}$ to $S$
end for
normalize scores in $S$ 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 $\{x_1, x_2, \dots, x_n\}$, a change point c is a point $c = x_i$ such that the sets of points $\{x_1, x_2, \dots, x_i\}$ and $\{x_{i+1}, x_{i+2}, \dots, x_n\}$ 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 β 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

```
$R \leftarrow$ set of relevant time series
$C \leftarrow$ catalog (empty)
for all time series $ts \in R$ do
    $t \leftarrow$ type of $ts$
    $C \leftarrow C \cup \{t\}$
    for all pairs label-value $(l, v)$ of $ts$ do
        $C \leftarrow C \cup \{(t, l, v)\}$
    end for
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" approaching 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

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

our setup and one based on activity traces provided by the Wikibench project [19].

For the experiments using data from a commercial data center, we used telemetry measuring availability of different server resources for the same intervals and we established events according to fluctuations in the total number of virtual machine updates through time. This telemetry data was provided unlabelled beyond the type of the value being measured, so only the *simple* approach was used.

Using the telemetry measurements obtained and the events identified, we then proceed to identify which sources of telemetry best capture the events registered as ground truth. For each time series with measurements from one particular telemetry source, we run our CPD framework to identify change points and compare those to the ground truth, building the catalog. 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.

V. Results

Table I summarizes the results from the different experiments we conducted. Results are grouped by the type of workload executed, namely: different mixes of synthetic activity (*synth1* and *synth2A)*, wikibench traces (*wikibench1* and *wikibench2*), and telemetry data from a private company (*private*). Also listed are the type of filtering done, either *simple* or *full*, as well as the score threshold used for filtering. The remaining columns list the average ratio of telemetry sources that were filtered out and how many of the sources that would be considered relevant were filtered out.

TABLE I
Filtering results

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

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

[1] V. Guralnik and J. Srivastava, "Event detection from time series data," in *Proceedings of the Fifth ACM SIGKDD International*

Conference on Knowledge Discovery and Data Mining, ser. KDD '99. New York, NY, USA: ACM, 1999, pp. 33–42. [Online]. Available: http://doi.acm.org/10.1145/312129.312190

[2] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, Dec. 2008. [Online]. Available: http://doi.acm.org/10.1145/1496091.1496103

[3] D. Gmach, J. Rolia, L. Cherkasova, and A. Kemper, "Workload analysis and demand prediction of enterprise data center applications," in *2007 IEEE 10th International Symposium on Workload Characterization*, Sept 2007, pp. 171–180.

[4] C. Wang, K. Schwan, V. Talwar, G. Eisenhauer, L. Hu, and M. Wolf, "A flexible architecture integrating monitoring and analytics for managing large-scale data centers," in *Proceedings of the 8th ACM International Conference on Autonomic Computing*, ser. ICAC '11. New York, NY, USA: ACM, 2011, pp. 141–150. [Online]. Available: http://doi.acm.org/10.1145/1998582.1998605

[5] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," *SIGCOMM Comput. Commun. Rev.*, vol. 41, no. 4, pp. 242–253, Aug. 2011. [Online]. Available: http://doi.acm.org/10.1145/2043164.2018465

[6] M. Amaral, J. Polo, D. Carrera, I. Mohomed, M. Unuvar, and M. Steinder, "Performance evaluation of microservices architectures using containers," in *2015 IEEE 14th International Symposium on Network Computing and Applications*, Sep. 2015, pp. 27–34.

[7] S. ur Rehman Baig, W. Iqbal, J. L. Berral, A. Erradi, and D. Carrera, "Real-time data center's telemetry reduction and reconstruction using markov chain models," *IEEE Systems Journal*, vol. 13, no. 4, pp. 4039–4050, 2019.

[8] F. Abuzaid, P. Bailis, J. Ding, E. Gan, S. Madden, D. Narayanan, K. Rong, and S. Suri, "Macrobase: Prioritizing attention in fast data," *ACM Trans. Database Syst.*, vol. 43, no. 4, Dec. 2018. [Online]. Available: https://doi.org/10.1145/3276463

[9] A. I. Jehangiri, R. Yahyapour, P. Wieder, E. Yaqub, and K. Lu, "Diagnosing cloud performance anomalies using large time series dataset analysis," in *2014 IEEE 7th International Conference on Cloud Computing*, June 2014, pp. 930–933.

[10] J. Xue, F. Yan, R. Birke, L. Y. Chen, T. Scherer, and E. Smirni, "Practise: Robust prediction of data center time series," in *2015 11th International Conference on Network and Service Management (CNSM)*, Nov 2015, pp. 126–134.

[11] V. Medel, O. Rana, J. A. Bañares, and U. Arronategui, "Modelling performance resource management in kubernetes," in *2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, Dec 2016, pp. 257–262.

[12] F. Hausdorff, *Grundzüge der Mengenlehre*. Chelsea, 1978.

[13] S. Aminikhanghahi and D. J. Cook, "A survey of methods for time series change point detection," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 339–367, May 2017. [Online]. Available: https://doi.org/10.1007/s10115-016-0987-z

[14] R. Killick, P. Fearnhead, and I. A. Eckley, "Optimal detection of changepoints with a linear computational cost," *Journal of the American Statistical Association*, vol. 107, no. 500, pp. 1590–1598, 2012. [Online]. Available: https://doi.org/10.1080/01621459.2012.737745

[15] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[16] D. Bernstein, "Containers and cloud: From lxc to docker to kubernetes," pp. 81–84, 2014.

[17] K. Keahey, P. Riteau, D. Stanzione, T. Cockerill, J. Mambretti, P. Rad, and P. Ruth, "Chameleon: a scalable production testbed for computer science research," in *Contemporary High Performance Computing: From Petascale toward Exascale*, 1st ed., ser. Chapman & Hall/CRC Computational Science, J. Vetter, Ed. Boca Raton, FL: CRC Press, 2018, vol. 3, ch. 5.

[18] P. Authors, "Prometheus," https://prometheus.io, 2014, accessed: 2019-10-12.

[19] E.-J. van Baaren, G. Pierre, and G. Urdaneta, "Wikibench: A distributed, wikipedia based web application benchmark," Master's thesis, Vrije Univesiteit Amsterdam, Department of Computer Science, 2009.