A Dataset of Performance Measurements and Alerts from Mozilla (Data Artifact)

Mohamed Bilel Besbes
m_besbes@live.concordia.ca
REALISE Lab @ Concordia University
Montréal, Québec, Canada

Diego Elias Costa
diego.costa@concordia.ca
REALISE Lab @ Concordia University
Montréal, Québec, Canada

Suhaib Mujahid
smujahid@mozilla.com
Mozilla
Montréal, Québec, Canada

Gregory Mierzwinski
gmierzwinski@mozilla.com
Mozilla
Potton, Québec, Canada

Marco Castelluccio
mcastelluccio@mozilla.com
Mozilla
London, United Kingdom

Abstract

Performance regressions in software systems can lead to significant financial losses and degraded user satisfaction, making their early detection and mitigation critical. Despite the importance of practices that capture performance regressions early, there is a lack of publicly available datasets that comprehensively capture real-world performance measurements, expert-validated alerts, and associated metadata such as bugs and testing conditions.

To address this gap, we introduce a unique dataset to support various research studies in performance engineering, anomaly detection, and machine learning. This dataset was collected from Mozilla Firefox's performance testing infrastructure and comprises 5,655 performance time series, 17,989 performance alerts, and detailed annotations of resulting bugs collected from May 2023 to May 2024. By publishing this dataset, we provide researchers with an invaluable resource for studying performance trends, developing novel change point detection methods, and advancing performance regression analysis across diverse platforms and testing environments. The dataset is available at https://doi.org/10.5281/zenodo.14642238.

CCS Concepts

• **Information systems** → **Decision support systems**; *Expert systems*; • **Applied computing** → **Investigation techniques**; • **Software and its engineering** → *Software maintenance tools*.

Keywords

Performance Regression Detection; Mozilla; Change Point Detection; Time Series Analysis; Perfherder; Performance Sheriffing

ACM Reference Format:

Mohamed Bilel Besbes, Diego Elias Costa, Suhaib Mujahid, Gregory Mierzwinski, and Marco Castelluccio. 2025. A Dataset of Performance Measurements and Alerts from Mozilla (Data Artifact). In *Companion of the 16th ACM/SPEC International Conference on Performance Engineering (ICPE Companion '25), May 5–9, 2025, Toronto, ON, Canada.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3680256.3721973

ICPE Companion '25, Toronto, ON, Canada

ACM ISBN 979-8-4007-1130-5/2025/05
https://doi.org/10.1145/3680256.3721973

1 Introduction

Software performance testing and monitoring have become crucial practices for ensuring the quality of modern software. Companies manage diverse platforms, and frequent updates to their products, and even small performance regressions can lead to significant financial losses and degraded user satisfaction. For instance, a study at Amazon revealed that a one-second delay in page load speed could result in an estimated $1.6 billion loss in annual revenue[1]. Also, Amazon stated that every 100 milliseconds in added page load time cost 1% in revenue in a study dating back to 2006[2]. On the other hand, software performance improvements, even if slight, help businesses with increasing performance indicators such as customer retention. For example, in 2011, a study showcased that a page load time reduction by 2.2 seconds of the download page for Mozilla's famous browser, Firefox, resulted in an additional 10 million Firefox downloads in a single year[17].

Maintaining efficient software performance has driven the adoption of robust performance engineering practices in the industry. Numerous companies spend significant time and resources to devise practices for detecting and mitigating performance regressions before they impact production [4]. There have been efforts in publishing industry datasets for performance measurement [5, 7, 18]. MongoDB's performance dataset that was published by Daly [5], in the ICPE Data Challenge of 2022. However, we believe that there is a lack of comprehensive datasets that capture these industrial practices that researchers can study. We introduce a dataset that contains one year of performance testing and monitoring data from Mozilla. This dataset includes 5,655 time-series performance measurements, 17,989 performance alerts with expert validation, and their associated performance bugs.

The development of this dataset involved an important effort in collecting, cleaning, and annotating performance measurements from diverse software systems of Mozilla. To enhance its utility, the performance measurements data was cross-referenced with the alerts. Also, the industrial effort of conducting the performance testing and manually annotating it was very notable as performance alerts are carefully validated and linked to corresponding bugs. This would enable researchers to trace the root causes of regressions effectively. By providing high-quality, annotated data, this dataset

serves as a valuable resource for analyzing performance trends, detecting performance anomalies, and improving platform-specific workflows.

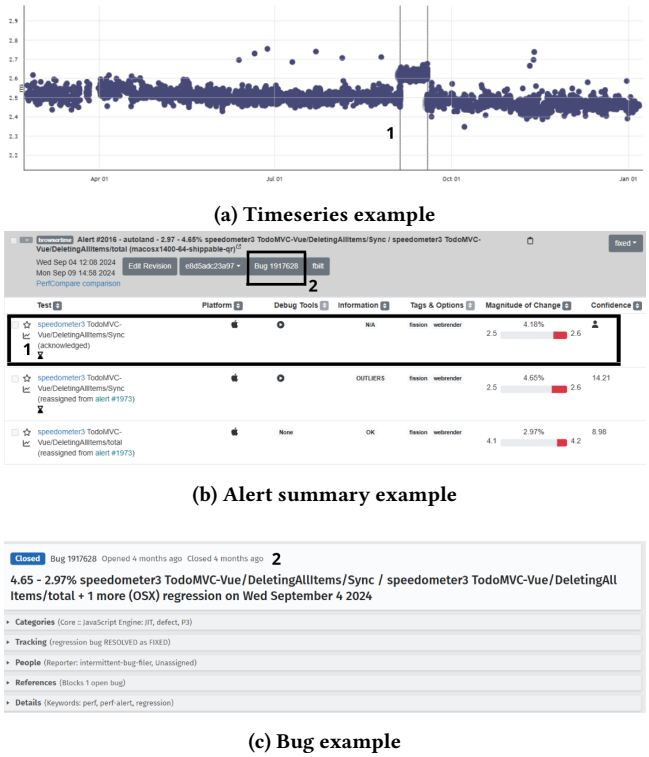
2 Context and Background

Mozilla Firefox is a popular open-source internet browser, known for its emphasis on privacy and customization, used by millions of users worldwide [15]. As responsiveness is a critical quality of internet browsers, the development team employs robust performance testing and monitoring strategies to capture potential regressions before they reach their end users. As new code is pushed to the codebase, multiple performance tests are executed on periodic basis to assess the performance of the new code. For each code revision [10] under test, performance tests run on various *platforms* (e.g., a specific operating system running on desktop hardware). The result of a single performance test is a *performance measurement*, which can represent execution time, memory consumption, and other tested performance characteristics. A platform is a collection of software and hardware setup

Given the inherent variability of software performance [9], performance tests could be repeated multiple times to increase the robustness of the results, a common practice in software performance engineering [3]. A *performance time series* is a sequence of performance measurements of the same performance test on a single platform throughout multiple revisions, and is commonly identified as a *signature* in Mozilla's terminology.

Mozilla's performance anomaly detection system is called Perfherder, and it is developed to identify performance anomalies that need further investigation. Mozilla's Perfherder periodically runs performance checks by analyzing one of the recent revisions by computing its T-test score from the two-sample student T-test[19] to contrast the measurements of generally 12 preceding revisions minimum that had performance measurements with the current plus a maximum of 11 subsequent ones. The resulting T-value is used as a confidence score showcasing the likelihood of the occurrence of a performance anomaly. In case the T-test shows a significant change detected by comparing it to a fixed threshold, Perfherder proceeds with measuring the change magnitude between both measurement groups, and if it surpasses a certain threshold, an alert is triggered on Perfherder [14]. Perfherder is the full system used in Mozilla to handle the performance workflow.

Alerts related to the same software revision are grouped in an *alert summary* (as presented in **1** in Figure 1). These alert summaries are then manually evaluated by a *Performance Sheriff*, a member of Mozilla's performance team, who assesses whether the alert should be further investigated. Performance Sheriffs can also create new alerts manually, if they notice a performance anomaly that was missed by Perfherder. In case the investigated alert summary presents an actual regression, a bug is created and associated with the given alert summary as shown in **2** in the same figure.



(a) Timeseries example

(b) Alert summary example

(c) Bug example

Figure 1: Illustrative example of the data as seen by Performance Sheriffs

3 Dataset Collection and Processing

3.1 Artifacts & dataset description

Our collected dataset contains four main entities: the performance time series, the alerts data, their associated alert summaries, and their associated bugs' data. Figure 2 showcases the relationships between the entities.

Performance time series: Performance time series contain series of performance measurements of a performance test and platform across multiple software revisions (x scale). The data contains all the measurement values, the measurement unit, the platform, its related test suite, and more. All time series collected contain at least one performance alert.

Alerts: An alert is a potential performance anomaly that was triggered by Mozilla Perfherder (automated) or by performance Sheriffs (manual). The alerts data characterizes the performance alert and contains the related alert summary (grouping), the alert's status, the time series ID, the result of the t-test, whether the alert was created manually or not, the noise profile of that alert if there is any, and other attributes.

Alert Summaries: An alert summary is a grouping of alerts on the same software revision. A revision could have multiple associated alert summaries. It is important to note that, alert summaries are the ones usually validated by Performance Sheriffs. The alert summary data contains details about its related software revision, the triage due date, the assignee among the performance Sheriffing team that will take a look at the alert summary, its associated bug, and other attributes.

Performance Bugs data: If a performance alert summary is validated by Performance Sheriffs, usually a performance bug is created to prompt the development team to action. The performance bug

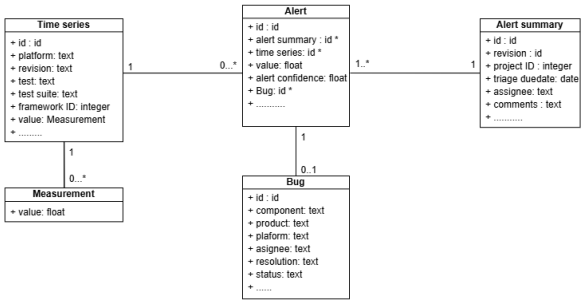


Figure 2: Structure of the relationships between the data entities

entity holds information about bug creation metadata (e.g., time of the creation, author), bug severity, bug status, creator, assignee, its associated product and component, comments from Sheriffs, its status of replication, and other attributes.

3.2 Data collection process

We collect the entire dataset using the Mozilla Perfherder API [12]. This particular API retains the historical data of alerts and performance measurements for one year, hence, we collect the data from May 2023 to May 2024. We proceed to collect the data using the following steps:

(1) We started by collecting performance alerts present in the Mozilla API, relevant to all Mozilla systems.
(2) Using the collected performance alerts, we identified the unique time series related to these alerts and we extracted their features. This means that all of our performance time series are associated with at least a single alert, and we have also not included time series that have not exhibited at least one single performance alert.
(3) We cleaned the dataset by removing empty time series for example. We also cross-referenced the alerts features with the time series measurements.
(4) Similar to the time series, we identified the unique bugs associated with the collected alerts and we extracted their features.

3.3 Data Labeling

Alert summaries group multiple alerts from the same software revision and are validated by Performance Sheriffs. These alerts could be created by Perfherder or by Performance Sheriffs. Upon validation, Performance Sheriffs update the alert summary status, which indicates whether an alert summary was deemed to be false (e.g., environmental change unrelated to the software) or needs further action from their team. We wanted to facilitate the use of this dataset by the research community, hence, we provide a custom labeling system, as follows:

- **True Alert summaries:** True alert summaries represent validated performance anomalies or regressions. They include alerts from summaries labeled as *reassigned*, *improvement*, *fixed*, *backedout*, *downstream*, or *wontfix*.
- **False Alert summaries:** False alert summaries do not represent real performance issues. They have the *invalid* status. They result from noise or irrelevant factors, for example.
- **Uncertain Alert summaries:** These are alert summaries whose validity remains undetermined. They are labeled as *investigating* or *untriaged*, requiring further review to classify them as true or false.

3.4 Dataset Characteristics

Table 1 showcases some of the statistics of our dataset, including performance time series, their associated alerts and performance bugs. Our dataset contains a total of 5,655 performance time series, covering performance tests of 186 different test suites across 5 different software platforms. On average, each performance time series contains 2,124 measurements and tests, and 1,867 software revisions. In total, the dataset contains ≈12 million measurements, and only 0.35% of performance measurements are associated with a performance alert.

Analyzing the performance alerts, we report a total of 17,989 alerts, 8,788 of which correspond to alerts from Speedometer3/TP6 test suites, the two of the most important Mozilla performance test suites. When grouped, the alerts correspond to 3,912 alert summaries. Most of the alert summaries come from two repositories, Autoland [11] and Mozilla Beta [13], respectively, representing 84.4% and 12.06% of the total alert summaries. Autoland is the first testing stage, followed by Mozilla Beta, which justifies the high occurrence of performance alerts in Autoland versus Mozilla Beta. The breakdown of the repository distribution is detailed in Table 1.

The total of 3,912 alert summaries are categorized as follows according to the logic stated in 3.3. True alerts represent 56.31% of the alerts, Uncertain alerts represent 31.21%, and False alerts represent 12.47%. Table 1 contains the alert summaries count per status for every single project.

To test the performance of a software revision, performance tests are executed across different software platforms. We report the alerts per tested platforms in Table 2. We also count the presence of a given platform at least once in alert summaries. There is a large presence of alerts from tests that run on Windows (6,241 alerts), macOS (5,298 alerts), Linux (4,210 alerts), and Android (2,217 alerts).

In case a performance regression is identified, it gets associated to a bug to be fixed. It is worth noting that out of the 3,912 existing alert summaries, only 633 have one associated bug out of the 482 unique bugs. Table 3 contains the breakdown of bugs by severity and status. The severity of the bug decrease in magnitude as the numerical value increases.

Structure of the dataset artifact. The dataset artifact [8] is organized as follows:

- **Scripts:** A folder that contains the scripts used to collect, clean and label the data. The scripts can also be used to recollect and update the dataset to include newer performance measurements and alerts from Mozilla systems.
- **Data:** A folder containing the performance time series data, alerts and alerts summaries, and bugs. Performance time series are further organized into their respective repositories, and we store a single CSV per performance time series. Alerts and bugs, on the other hand, are stored on one CSV file each.

Table 1: Dataset statistics by repository.

Repository	Time Series	Alerts	Custom Alert Labels			Alert Summaries	Bugs
			True	False	Uncertain		
Autoland	3833	13593	1779	431	1109	3319	438
Mozilla Beta	1477	3597	317	45	110	472	25
Firefox Android	342	796	105	12	1	118	25
Mozilla Central	2	2	2	0	0	2	2
Mozilla Release	1	1	0	0	1	1	1
Total	**5655**	**17989**	**2203**	**488**	**1221**	**3912**	**482**

Table 2: Breakdown of the Alerts/Alert Summaries by Platform

Platform	Alerts	Alert Summaries
Windows	6241	1754
macOS	5298	1615
Linux	4210	1302
Android	2217	330
Other	23	23
Total	**17989**	**3912**

Table 3: Characteristics of the 482 performance bugs in the dataset

Characteristic	Level	# of Bugs
Bug Severity	S2	26
	S3	125
	S4	33
	Unknown	298
Bug Status	NEW	35
	ASSIGNED	3
	REOPENED	1
	RESOLVED	427
	VERIFIED	16

4 Areas of data potential usage

Given that the dataset includes performance time series, validated performance alerts, and their associated bugs, we envision that it can be used to conduct research on different areas in performance engineering.

Performance characterization The performance time series can be used to best understand the performance profile of industrial software. Researchers can use this dataset to characterize the performance evolution of Mozilla systems, identify performance trends, characterize performance measurement variation, and correlate measured performance across different testing platforms.

Performance Regression Prediction The dataset includes performance measurements, expert-validated performance changes (alerts), and our custom label that makes it ideal for predicting performance regression. This dataset can be used to test different approaches, such as change point detection methods [20], time-series forecasting [6] or regression models [16]. The dataset also contains other metadata attributes (such as the noise profile), which can further help classify performance measurements.

Characterization of Performance Bugs We envision researchers using the published dataset as a starting point to characterize performance bugs. The dataset contains metadata that can be used to assess how long bugs take to be created and fixed, how many professionals are directly involved in the related discussion, and how much debate goes into fixing a performance bug.

Performance regressions root cause analysis Given Mozilla develops open-source software, it is also possible to extend this work to extract the code-related features. The meta-data included in the dataset targets both the repository, test suite, and the revision ID, that can be used to mine the exact commit and to analyze what are the code modifications that caused regressions. This is especially promising because the dataset covers multiple repositories/projects.

5 Limitations

Bias in the data collection: we have collected performance measurements associated with at least a single performance alert. Performance time series that do not yield any alert is not present in the dataset. So, the dataset might not fully represent the entire spectrum of performance measurements, particularly normal or less 'anomalous' measurements that do not trigger alerts. Researchers can, however, use our data collection scripts to expand the dataset to include times series that do not exhibit any anomalous behavior.

Potential threats in data collection: Data alerts and associated performance measurements were collected on different dates. This temporal separation could lead to minor inconsistencies in the dataset. We mitigated this threat by cross-verifying the alerts and measurements alignment.

Generalizability: While the dataset focuses on Mozilla-tested products, including browsers and other software, it may not generalize seamlessly to products outside the Mozilla ecosystem. However, we believe that many practices and processes used by Mozilla's performance team (such as time series monitoring, alerting mechanisms, and regression identification strategies) can offer transferable insights to other use cases.

6 Conclusion and Future work

We present a novel dataset of performance time series, alerts and bugs, tailored to support research on software performance engineering. This dataset is collected from real-world software performance monitoring systems in Mozilla industrial settings. By providing this resource, we hope to enable further studies in a variety of performance engineering topics.

In the near future, we aim to extend the dataset by including the latest performance measurements and potentially include code-related features. We encourage the community to build on this artifact in order to push the boundaries of performance engineering research and explore new methodologies for ensuring software reliability and efficiency.

References

[1] Fast Company. 2012. *How One Second Could Cost Amazon $1.6 Billion In Sales.* https://www.fastcompany.com/1825005/how-one-second-could-cost-amazon-16-billion-sales

[2] CONDUCTOR. 2024. *Amazon Study: Every 100ms in Added Page Load Time Cost 1% in Revenue.* https://www.conductor.com/academy/page-speed-resources/faq/amazon-page-speed-study

[3] Diego Costa, Cor-Paul Bezemer, Philipp Leitner, and Artur Andrzejak. 2021. What's Wrong with My Benchmark Results? Studying Bad Practices in JMH Benchmarks. *IEEE Transactions on Software Engineering* 47, 7 (2021), 1452–1467. doi:10.1109/TSE.2019.2925345

[4] David Daly. 2021. Creating a Virtuous Cycle in Performance Testing at MongoDB. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering* (Virtual Event, France) *(ICPE '21)*. Association for Computing Machinery, New York, NY, USA, 33–40. doi:10.1145/3427921.3450234

[5] David Daly. 2022. *International Conference of Performance Engineering 2023 Data Challenge Track.* https://icpe2022.spec.org/tracks-and-submissions/artifact-evaluation-track/

[6] Federico Di Menna, Luca Traini, and Vittorio Cortellessa. 2024. Time Series Forecasting of Runtime Software Metrics: An Empirical Study. In *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering* (London, United Kingdom) *(ICPE '24)*. Association for Computing Machinery, New York, NY, USA, 48–59. doi:10.1145/3629526.3645049

[7] Michaela Hardt, William R. Orchard, Patrick Blöbaum, Shiva Kasiviswanathan, and Elke Kirschbaum. 2024. The PetShop Dataset – Finding Causes of Performance Issues across Microservices. arXiv:2311.04806 [cs.DC] https://arxiv.org/abs/2311.04806

[8] REALISE Lab. 2024. *Dataset and replication package.* https://zenodo.org/records/14642239

[9] Aleksander Maricq, Dmitry Duplyakin, Ivo Jimenez, Carlos Maltzahn, Ryan Stutsman, and Robert Ricci. 2018. Taming performance variability. In *Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation* (Carlsbad, CA, USA) *(OSDI'18)*. USENIX Association, USA, 409–425.

[10] Mercurial. 2024. *Mercurial Code Revision.* https://wiki.mercurial-scm.org/Revision

[11] Mozilla. 2024. *Autoland Code Repository.* https://hg.mozilla.org/integration/autoland/

[12] Mozilla. 2024. *Mozilla API Documentation.* https://treeherder.mozilla.org/docs/

[13] Mozilla. 2024. *Mozilla Beta Release Notes.* https://www.mozilla.org/en-US/firefox/129.0beta/releasenotes/

[14] Mozilla. 2024. *Mozilla's Perfherder Dashboard.* https://treeherder.mozilla.org/perfherder/alerts?hideDwnToInv=1&page=1

[15] Mozilla. 2025. *Mozilla's Firefox.* https://www.mozilla.org/en-US/firefox/

[16] Reza Ghanbari Baghnavi Omid Bushehrian. 2011. Article:A Regression-based Method for Software Performance Engineering. *International Journal of Computer Applications* 31, 7 (October 2011), 46–50. doi:10.5120/3840-5341

[17] SITESPECT. 2011. *Case Study: Mozilla Optimizes Web Site Content and Performance through Multivariate Testing.* https://info.sitespect.com/hubfs/Case%20Studies/Mozilla%20x%20SiteSpect_Case%20Study.pdf

[18] Per Erik Strandberg and Yosh Marklund. 2023. The Westermo test system performance data set. arXiv:2311.14510 [cs.SE] https://arxiv.org/abs/2311.14510

[19] Student. 1908. The probable error of a mean. *Biometrika* 6, 1 (1908), 1–25. doi:10.2307/2331554

[20] Charles Truong, Laurent Oudre, and Nicolas Vayatis. 2020. Selective review of offline change point detection methods. *Signal Processing* 167 (2020), 107299.