Change Point Detection for MongoDB Time Series Performance Regression

Mark Leznik
Ulm University
Germany

Md Shahriar Iqbal
University of South Carolina
USA

Igor Trubin
Capital One
USA

Arne Lochner
Ulm University
Germany

Pooyan Jamshidi
University of South Carolina
USA

André Bauer
University of Würzburg
Germany

ABSTRACT

Commits to the MongoDB software repository trigger a collection of automatically run tests. Here, the identification of commits responsible for performance regressions is paramount. Previously, the process relied on manual inspection of time series graphs to identify significant changes, later replaced with a threshold-based detection system. However, neither system was sufficient for finding changes in performance in a timely manner. This work describes our recent implementation of a change point detection system built upon time series features, a voting system, the Perfomalist approach, and XGBoost. The algorithm produces a list of change points representing significant changes from a given history of performance results. We are able to automatically detect change points and achieve an 83% accuracy, all while reducing the human effort in the process.

CCS CONCEPTS

• **General and reference** → **Measurement**; • **Computing methodologies** → **Machine learning**; • **Information systems** → **Data management systems**.

KEYWORDS

Anomaly detection, Change point detection, Performance regression

ACM Reference Format:
Mark Leznik, Md Shahriar Iqbal, Igor Trubin, Arne Lochner, Pooyan Jamshidi, and André Bauer. 2022. Change Point Detection for MongoDB Time Series Performance Regression. In *Companion of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22 Companion), April 9–13, 2022, Bejing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3491204.3527488

1 INTRODUCTION

The performance testing infrastructure at MongoDB is central to detecting performance degradation and resolution of performance regressions to ensure the quality of the released software. It is essential to prompt isolation of performance bugs and locking in performance improvements. However, due to inherent noises and the sheer number of experiments needed to run to test each benchmark in the testing platform, performance testing, e.g., detection and/or confirmation of performance changes, remains a challenging task [5]. Hence, an automated approach for detection of performance regressions on time without the introduced costs of repeated tests is highly desirable [4].

MongoDB has open-sourced a large data set of performance test logs, allowing a deeper insight into the testing process and the underlying failures. The current approach involves change point detection using E-divisive means algorithm [9] with a subsequent triage by an engineer. Unfortunately, change points that are too close in the time series may go unnoticed with the existing approach [5]. Additionally, there is no way to verify the correctness of the detected change points without manual investigation by an engineer. The occurrence of system noise in the data, which is flagged as a change point, leads to a higher workload for the support team and increases the risk of missed actual failures.

To address these issues, we develop a hybrid change point detection system as shown in Figure 1. Our approach uses an unsupervised multivariate adaptive statistical filtering (MASF) based

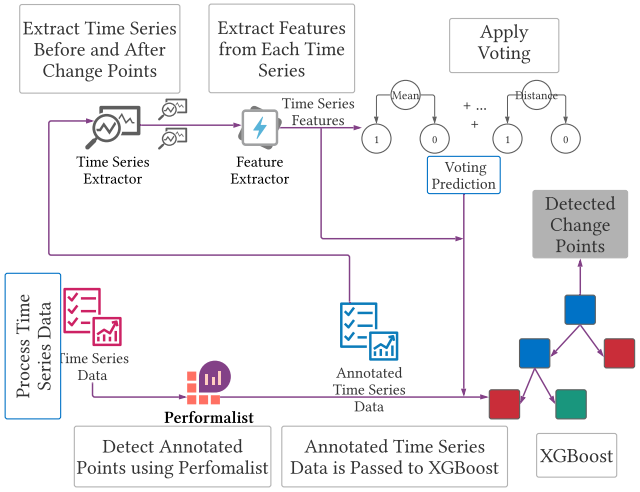

Figure 1: Overview of the proposed change point detection technique.

ICPE '22 Companion, April 9–13, 2022, Bejing, China

ACM ISBN 978-1-4503-9159-7/22/04…$15.00
https://doi.org/10.1145/3491204.3527488

habitual change point detection algorithm, Perfomalist [1], in combination with XGBoost [3] and a time series feature-based voting classifier. The supervised classification system corrects the mistakes made by the automated change point detection system, thereby increasing the accuracy of the annotated change points and reducing the human workload in this process.

The rest of this work is structured as follows: Section 2 presents time series features and distance measures we use, as well as the Perfomalist change point detection and XGBoost. Section 3 describes the statistical anaylsis of the MongoDB data. Section 4 presents our approach, as well as the results of the change point detection. Section 5 summarizes our results and depicts the future work ahead.

2 BACKGROUND

2.1 Time Series Features

We compare the time series before and after the possible change point to detect a change point. The time series before the change point starts with either the first measurement or the last confirmed change point and ends with the previous measurement of the possible change point. The other time series includes the possible change point itself and its subsequent measurement. We assume that a change point can only be found between the second and the last value of a measurement series.

To describe a possible change point, we examine how the behavior of the time series changed because of that point. In other words: We calculate the difference (Δ) of different time series features describing the time series before and after the possible change point. The time series features used are listed in Table 1.

Table 1: List of time series features used in our study.

Feature	Description
Mean	The mean value of the time series
SD	The standard deviation of the time series
Hurst	The Hurst exponent [7] describes how similar the time series is to some part of itself
Alpha	Parameter for the level component in Holt's linear trend method [8] fitted to the time series.
Beta	Parameter for the trend component in Holt's linear trend method fitted to the time series.

2.2 Time Series Distance

To determine how dissimilar the time series before the possible change point is to the time series after the change point, we calculate the time series distance between these two time series. If we apply a geometric distance, we could, for instance, the dynamic time wrapping as the time series vary in length. However, the results would then be difficult to interpret. Therefore, we use the zoomed ranking approach [6] applied to time series [2]. Mathematically, the distance between two time series T_i and T_j is defined as:

$$d(T_i, T_j) := \sum_{m=1}^{5} \frac{|f_{T_i,m} - f_{T_j,m}|}{\max\limits_{k=\{1,\ldots,s\}/\{i\}}(f_{T_k,m}) - \min\limits_{k=\{1,\ldots,s\}/\{i\}}(f_{T_k,m})}, \quad (1)$$

where $T_1, \ldots, T_s$ are all time series, s is a positive integer, and $f_{T_i,1}, \ldots, f_{T_i,q}$ are the descriptive time series features of time series T_i. The distance between two time series is in the interval $[0, \infty)$, where 0 means that the time series are equal with respect to their descriptive features. The larger the distance, the more heterogeneous the two time series are.

2.3 Perfomalist

Perfomalist [1] is a web-based tool for detecting anomalies and change points. The tool uses a method called SETDS - Statistical Exception and Trend Detection System, a variation of the Statistical Process Control method applied to time series data. The central idea of the method is EV (Exception Value). EV indicates the severity of anomalies and is a magnitude of exception. This is calculated as the difference between the control limits and the actual anomalous data points. The first occurrence of any change would appear as an anomaly and then may become a normality (new norm). As a result, severity analysis of all the anomalies collected over time provides an opportunity to find phases in the data history with different patterns. To detect change points between phases, all roots of the following equation must be found:

$$(EV)(t) = 0, \quad (2)$$

where t is the time. Using this method, the Perfomalist returns all the change points found in the input time series data.

3 STATISTICAL ANALYSIS

To better understand the change points, we examined the provided dataset using statistical analysis. To this end, for each triaged point (false or true positive), we calculated the time series features before and after that point, as described in Section 2.1. Then, we calculated the delta feature (Δ) as the difference between the feature before and after a triaged point for each of these points and features. We also calculated the time series distance for the time series before and after a triaged point.

In the first analysis, we tested the normality for all these calculated features with the Shapiro Wilk test: All features are not normally distributed. To this end, we perform a median analysis and show the results in Table 2. The table shows median values of the features with their 1st to 3rd quartiles for change points (true positive) and misclassified change points (false positive), respectively. We used the Fisher's exact test (in the case of fewer than five points) or Chi-square tests to compare the medians between the two classes: The median values are significantly different, so there is an actual difference between the medians. For instance, the ΔMean exhibits a median value of 0.83 for true change points and 0.37 for misclassified change points. Accordingly, this feature is a good indicator of whether a particular point might be a change point.

The second analysis investigates how the time series distance for change points behaves across the different projects. The results are presented in Table 3, where we apply the Wilcoxon rank-sum and signed-rank test to check whether the difference is significant. We also list the number of true and misclassified change points for each project. It is worth noting that almost all projects have a similar distance of about 2.4; however, the project sys-perf has a

Table 2: Median analysis of the considered time series features.

	True Positive	False Positive	p-value
ΔMean	0.83 [0.69;0.91]	0.37 [0.19;0.60]	2.20E-16
ΔSD	0.05 [0.02;0.10]	0.14 [0.07;0.20]	2.20E-16
ΔHurst	0.19 [0.00;0.40]	0.38 [0.04;0.46]	8.69E-132
ΔAlpha	0.34 [0.11;0.50]	0.21 [0.09;0.46]	6.86E-79
ΔBeta	1.00 [0.96;1.00]	1.00 [1.00;1.00]	5.61E-01
Distance	2.53 [2.20;2.86]	2.42 [2.15;2.68]	2.17E-56

distance of 1.79 and 1.44, and thus, the change points in this project may be harder to detect.

Table 3: Comparison of time series distances across different MongoDB projects. We list the true positives and false positives as TP and FP, respectively.

Project	Median Distance		p-value	Number	
	TP	FP		TP	FP
PERFORMANCE	2.83	2.41	9.34E-294	1362	2513
PERFORMANCE-4.2	-	1.91	-	0	13
PERFORMANCE-4.4	2.30	2.65	2.42E-04	12	914
PERFORMANCE-5.0	2.79	2.48	7.04E-45	674	598
SYS-PERF	2.47	2.36	2.51E-15	3204	2944
SYS-PERF-4.0	1.77	1.04	4.71E-05	20	36
SYS-PERF-4.2	2.44	2.42	7.17E-01	69	216
SYS-PERF-4.4	2.67	2.35	9.90E-23	291	905
SYS-PERF-5.0	1.79	1.44	4.98E-04	1358	202

Lastly, we are interested in how well the points can be separated by splitting the data according to the median of the total population of a particular time series feature see Table 4. To investigate whether the distribution of the points is significant, we apply the Chi-square test. Since we used the median value of the total population, the number of points below and above the median is 'identical'. For example, if we use only the ΔMean to classify triaged points into true change points and misclassified change points, we can detect 5708 out of 6910 true change points.

4 DETECTION OF CHANGE POINTS

4.1 Voting based on Time Series Features (Classification)

Based on the statistical analysis performed in Section 3, we build a voting system for classifying change points (see Figure 1). More specifically, for a potential change point, we calculated the time series features before and after the possible change point and calculated the difference between the two sets of features, as described in Section 2. Then, for each of these delta features, we perform a binary decision (1 = change point; 0 = no change point), as shown in Figure 2. To not rely on a single delta-features, we applied a voting system. Each delta feature has a weight (derived from the statistical analysis), and the weighted decisions are summed up. Then the sum is normalized by the sum of the weights (~3.46), and if the normalized sum is greater than 0.5, the potential change point is a real change point.

Table 4: Comparison of time series features according to the change points.

	True Positive	False Positive	p-value
ΔMean >= 0.62	5708/7666	1958/7666	2.20E-16
ΔMean < 0.62	1282/7665	6383/7665	
ΔSD >= 0.09	2016/7666	5650/7666	2.20E-16
ΔSD < 0.09	4974/7665	2691/7665	
ΔHurst >= 0.31	2729/7666	4937/7666	3.88E-136
ΔHurst < 0.31	4261/7665	3404/7665	
ΔAlpha >= 0.26	3963/7666	3703/7666	7.09E-52
ΔAlpha < 0.26	3027/7665	4638/7665	
ΔBeta >= 1.00	3572/7666	4094/7666	1.34E-02
ΔBeta < 1.00	3418/7665	4247/7665	
Distance >= 2.48	4012/7666	3654/7666	6.32E-63
Distance < 2.48	2978/7665	4687/7665	

Table 5: Evaluation of the voting approach.

	Accuracy	Sensitivity	Specificity
Without distance	0.6962	0.7721	0.6056
With distance	0.7797	0.8065	0.7476

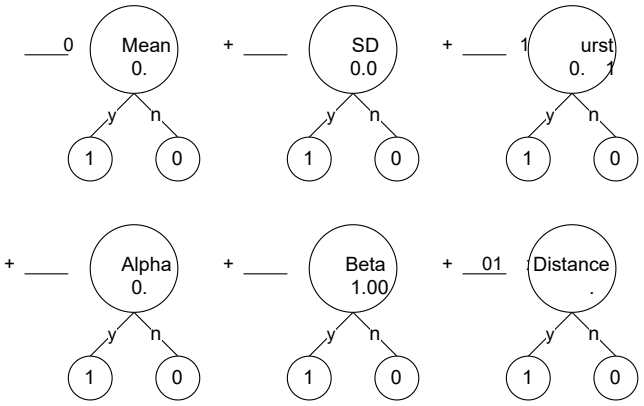

Figure 2: Voting system used for change point classification using the time series features.

To investigate this voting system, we classify each triaged point (false positive or false positive). We also investigate whether using the time series distance increases or decreases the accuracy. The results are given in Table 5. Without the distance, the voting system has an accuracy of ~70%, while using the distance, the accuracy is ~78%.

4.2 Perfomalist (Detection)

In addition to the classification based on the voting system, we also require the detection of change points (see Figure 1). To this end, we

applied Perfomalist to find change points within a time series. We investigated only the time series that contained labeled points (false negative, true positive, not triaged, or under submission) to have ground truth. The results are shown in Table 6. Overall, Perfomalist found 4691 of 6990 true change points (true positive) and 2110 misclassified change points (false positive) in these time series. In addition, Perfomalist labeled 1490 not triaged points as change points and found 9930 change points that were not annotated.

Table 6: Distribution of change points discovered by Perfomalist.

True Positive	False Postive	Not Triaged	Not Annotated
4691	2110	1490	9930

Based on the true and false positive labeled change points in the data set, we calculated the accuracy of the detection of Perfomalist. The results are listed in Table 7. Perfomalist exhibits an accuracy of ~71%.

Table 7: Accuracy, Sensitivity and Specifity of the change points identifed by Perfomalist.

Accuracy	Sensitivity	Specificity
0.7124	0.6898	0.7305

4.3 Perfomalist information with Voting in XGBoost

To combine the classification of the voting system and the detection by Perfomalist, we use XGBoost [3] as an intermediary between these two approaches (see Figure 1). The idea is that XGBoost gets the time series features, the decision of the voting system, and the information from Perfomalist to detect and classify change points. More specifically, XGBoost should learn whether the voting system or Perfomalist classified a change point or made a mistake based on the time series features.

To train XGBoost, we split the time series containing labeled points into 90% training and 10% test. To avoid arbitrary splitting, we split the data set ten times. For each split, the XGBoost model receives as input the time series features (ΔMean, ΔSD, ΔHurst, ΔAlpha, ΔBeta, and Distance), the decision of the voting system and the detection results from Perfomalist and as label whether a point is a change point or not.

The results of testing the XGBoost model are shown in Table 8. The values given are the averages over the ten splits. Combining the voting system with the information provided by Perfomalist increases the accuracy to ~83% and has a sensitivity and specificity above ~80%.

4.4 Threats to Validity

We perform all experiments only with the time series containing labeled points. Therefore, we cannot generalize the results to the entire data set. However, we are confident that our approach would provide similar accuracy for the rest of the data set. The methods and time series features used were selected by trial-and-error. There may be better features or methods available. Finally, the entire approach is trained based on the labels of the data set, i.e., the approach relies on the correctness of these data.

Table 8: XGBoost results in an increase of accuracy, sensitivity, and specificity in detecting and classifying change points.

	Accuracy	Sensitivity	Specificity
Voting	0.7797	0.8065	0.7476
Pefomalist	0.7124	0.6898	0.7305
XGBoost	0.8290	0.7906	0.8609
Max. improvement	**0.1166**	**0.1008**	**0.1304**

5 CONCLUSION

MongoDB's automatic performance tests triggered after each commit allow them to find performance regressions and maintain the quality of their software. Although they apply a detection algorithm to find potential change points, the labeling is done by experts. To avoid or reduce humans in the loop, we propose an automatic approach to change point detection and classification in this paper. The key idea is to combine time series features describing the time series behavior before and after a potential change point, a voting system based on statistically determined rules, and an anomaly detection algorithm to address this task. Our results show that we are able to automatically detect and classify change points in the MongoDB dataset with 83% accuracy. Based on these results, we are confident that our approach can reduce the human effort in this process.

REFERENCES

[1] [n.d.]. Perfomalist. https://www.perfomalist.com/.

[2] André Bauer, Marwin Züfle, Simon Eismann, Johannes Grohmann, Nikolas Herbst, and Samuel Kounev. 2021. Libra: A Benchmark for Time Series Forecasting Methods. In *Proceedings of the 12th ACM/SPEC International Conference on Performance Engineering (ICPE)*. ACM, New York, NY, USA.

[3] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *ACM Special Interest Group on Knowledge Discovery in Data 2016*. ACM, 785–794.

[4] David Daly. 2021. Creating a Virtuous Cycle in Performance Testing at MongoDB. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 33–41.

[5] David Daly, William Brown, Henrik Ingo, Jim O'Leary, and David Bradford. 2020. The use of change point detection to identify software performance regressions in a continuous integration system. In *Proceedings of the ACM/SPEC International Conference on Performance Engineering*. 67–75.

[6] Patrícia Maforte Dos Santos, Teresa Bernarda Ludermir, and Ricardo Bastos Cavalcante Prudencio. 2004. Selection of Time Series Forecasting Models Based on Performance Information. In *Fourth International Conference on Hybrid Intelligent Systems (HIS'04)*. IEEE, 366–371.

[7] John Haslett and Adrian E Raftery. 1989. Space-time modelling with long-memory dependence: Assessing Ireland's wind power resource. *Journal of the Royal Statistical Society: Series C (Applied Statistics)* 38, 1 (1989), 1–21.

[8] Charles C Holt. 1957. *Forecasting Seasonals and Trends by Exponentially Weighted Moving Averages.* Technical Report. Carnegie Institute of Technology.

[9] David S Matteson and Nicholas A James. 2014. A nonparametric approach for multiple change point analysis of multivariate data. *J. Amer. Statist. Assoc.* 109, 505 (2014), 334–345.