Identification and Classification of JMH Microbenchmark States using Time Series Analysis

Tom Wallace
tw18dw@brocku.ca
Brock University
St. Catharines, Ontario, Canada

Beatrice Ombuki-Berman
bombuki@brocku.ca
Brock University
St. Catharines, Ontario, Canada

Naser Ezzati-Jivan
nezzati@brocku.ca
Brock University
St. Catharines, Ontario, Canada

ABSTRACT

The practice of microbenchmarking is very important for observing the performance of code. As such, observing the states and anomalies experienced by the program during a benchmark is equally important. This paper attempts to evaluate the effectiveness of the matrix profile method when applied to analyse JMH benchmarks in time series format, to determine if it is a viable alternative to proven methods. We observe that, when using the matrix profile method, there is a statistically significant difference between the results of the analysis on steady state and non-steady state benchmarks. By comparing results of the matrix profile method and the proven changepoint analysis method, we are able to prove a stronger correlation between the two when the benchmark tested is non-steady state versus that of steady state.

ACM Reference Format:
Tom Wallace, Beatrice Ombuki-Berman, and Naser Ezzati-Jivan. 2023. Identification and Classification of JMH Microbenchmark States using Time Series Analysis . In *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3578245.3584694

CCS Concepts

General and reference→Performance • **Software and its engineering**→Software prototyping; Operational analysis; Software defect analysis.

Keywords

ICPE Data Challenge, Time Series Analysis, Benchmarking, Steady-State

1 INTRODUCTION

Microbenchmarking, the practice of testing the performance of one or many small pieces of code, is a significant part of evaluating software performance. When these pieces of code are run frequently and repeatedly for every execution of a program, minimizing their impact on the overall runtime is integral. The Java Microbenchmarking Harness (JMH) is one example of a microbenchmarking framework, where various performance metrics such as throughput and average running time can be calculated from repeatedly running the code in question and measuring how long each iteration takes to complete. [1]

JMH testing is designed with the idea that the initial runs of the program are not indicative of its peak performance. The common belief is that the program will, after a certain amount of runs, reach what is called a *steady state*. To this end, JMH benchmarks are run in two phases: A "warmup phase" for which the results will be discarded, followed by the actual benchmark. Though the aforementioned belief is true in many cases, research has shown that the steady state will not always be reached when using default JMH warmup settings, and manual estimations of the necessary length of the warmup phase are often incorrect [1]. As a result, a common point of research is the identification of whether a benchmark is able to reach steady state.

The raw output of such benchmarks is most often in the form of a time series: a sequential list of data values measuring the change in a variable over a series of time or a number of iterations of a process. In the case of JMH, it measures the runtime for each iteration of the code execution. It is a very common form of measurement in various fields of study, and its relative simplicity means that a wide range of methods for their analysis have been developed. Through our research questions, we aim to determine whether these methods of analysis can be useful in observing the state of a JMH benchmark. By applying the Matrix Profile technique and the pattern (motif) recognition methods to a selection of both steady and non-steady state benchmarks, we can compare the results observed in each type of benchmark. We also observe the practicality and effectiveness of these methods for performance anomaly detection. Finally, we compare the results obtained by the Matrix Profile method to those obtained through the Change Point Detection method, and attempt to observe whether there is a meaningful correlation between each method's results.

Through our experiments, we are able to prove a statistically significant difference between the matrix profile values observed in benchmarks which are able to achieve steady state, and the values observed in benchmarks which do not. Additionally, we provide examples of matrix profile analysis which demonstrate its strengths and weaknesses for the purpose of anomaly detection. Finally, we observe that benchmarks which do not reach a steady state tend to have a closer correlation between changepoints and the matrix profile. Code used for this purpose along with a selection of raw data is available on Github[1].

ICPE '23 Companion, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0072-9/23/04…$15.00
https://doi.org/10.1145/3578245.3584694

[1] https://github.com/Tom-Wallace/patternReader

2 BACKGROUND & RELATED WORK

2.1 Performance Anomalies

A performance anomaly within a program execution or benchmark is an event which causes performance metrics and/or resource utilization to leave the range of expected values. Their causes are varied, but are most often due to bugs within the program, attempts to use a large or complex workload, or problems with the hardware itself. Whenever the cause is within the programmers' power to fix, it is highly important that they attempt to do so. As they are most often characterized by a large spike or gradual increase in performance metrics, they tend to be highly visible within performance charts, and as such should be receptive to methods of time series analysis [6].

2.2 Time Series Analysis

2.2.1 Motif Discovery via Matrix Profile. The Matrix Profile of a subseries is a powerful tool for analysis. It consists of two parts; the first, the distance profile, is a secondary time series calculated using the time series to be analysed, and the second, the matrix profile index (or just profile index), is an array of indices. Given the main time series T and a window length m, the value at index i in the Matrix Profile is calculated by comparing the subsequence of T[i, i+1,..., i+m] to all other subsequences of length m using the Euclidean distance, and taking the smallest distance found via this process. The value at index i of the matrix profile array is the index of the first member of the subsequence which had the smallest distance.

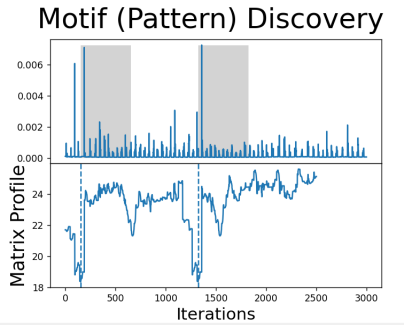

Figure 1: Example of a time series with its matching Matrix Profile

As shown by Fig. 2, the two lowest values in the matrix profile graph show the starting points of the two most similar subsequences within the time series, while the highest points in the graph signify discords; Subsequences which are unique, potentially indicating an anomaly. The natural extension of this process is to automatically identify Motifs (or Patterns), which are similar shapes within the time series.

2.2.2 Change Point Detection. Change Point Detection is an analysis method which seeks to separate a time series into distinct phases, with the moment one phase ends and another begins being referred to as a change point. The change points are found by finding points in the time series where the data values observed shift significantly. The presence of a change point generally designates a point where the mean of the values before the change point is significantly different from the mean of the values after. The PELT algorithm is one method proposed to achieve this; Though it will not always discover every admissable change point, it is reasonably effective while achieving a linear computation time with respect to the length of the time series. [2]

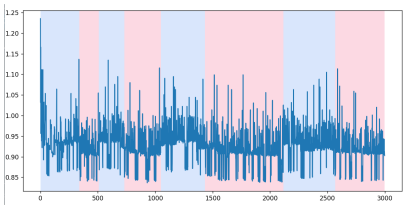

Figure 2: Example of change point detection, where each phase is marked in alternating colours

2.2.3 Related Work. A paper by Traini *et al.* [1] introduces the topic of JMH state analysis and outlines current research on the subject. In addition, they prove a number of discoveries by performing experiments on a large portion of JMH time series files. These files are provided freely for additional research, and are used for the experiments performed for this paper. Barrett *et al.* [3] discuss how frequently JMH benchmarks are able to reach steady state. They conclude that at most, 43.5% of VM/Benchmark pairs are able to consistently reach steady state. As a part of their research, they define an automated method by which change point analysis may be used to classify a benchmark as having reached a steady state.

3 METHODOLOGY

The research questions concern the previously mentioned forms of time series analysis. In particular, we aim to discover the effectiveness of the Matrix Profile & Change Point Detection methods when applied to the analysis of JMH Benchmarks.

RQ1 *How effective are Motif discovery and the Matrix Profile in aiding identification of steady state/non-steady state benchmarks?*

RQ2 *How might Motif discovery and the Matrix Profile be used to detect performance anomalies?*

RQ3 *How comparable are the results obtained by Motif discovery versus Change Point Detection?*

3.1 RQ1 - Motif Effectiveness

Comparisons will be made between the matrix profile graphs of benchmarks which are known to be non-steady state and those which are either steady state or ambiguous. We hypothesize that the matrix profile of a program which frequently changes states will capture these changes within the motif windows. Discords may similarly be useful when a state is unique.

3.2 RQ2 - Anomaly Motifs

Performance anomalies tend to be easily visible when graphed alongside the rest of the execution. For automating this process, the matrix profile may be useful to identify significant differences from normal execution. Discords will be the most useful when the anomaly occurs once within the benchmark, while motifs may be useful if it is repeated.

3.3 RQ3 - Motifs & Change Points

As a number of non-steady state benchmarks tend to alternate between similar ranges (as seen in Fig. 2), it will be observed whether the Matrix Profile approach is able to observe motifs at approximately the same locations where change points occur.

4 EXPERIMENTAL SETUP

In each analysis of the Matrix Profile, the open-source python library *Stumpy*[4] is utilised. Changepoint analysis uses the open-source python library *Ruptures*[5]. All analysis is performed on randomly sampled benchmark data from the dataset provided by Traini et al [1].

4.1 RQ1

A portion of 497 steady state and 235 non-steady state benchmark forks will be identified and analysed, using changepoint analysis alongside a classification algorithm defined by Barrett *et al.*[3]. As benchmarks reach steady state in most cases, there will be more samples for such benchmarks. The minimum, maximum, and mean of each benchmark's matrix profile will be recorded and compared. Each set of measurements will be compared using a Two-sample T-Test assuming unequal variance to determine whether there is a statistically significant difference. The null hypothesis of these tests is that there will be no difference.

4.2 RQ2

This question is more difficult to definitively test, as information on whether an anomaly occurs in a benchmark is not provided, nor is it easily definable. Nevertheless, we may still test whether the matrix profile can be used to identify performance patterns commonly associated with anomalies; large spikes and large increases. After obtaining the matrix profile, the mean and standard deviation of the entire time series will be compared to those of the two identified motif windows and the discord window. If the mean of one of these windows differs from the mean of the time series by at least one time series standard deviation, or by two time series standard deviations, we can reasonably suspect an anomaly. If the standard deviation of one of the windows is at least two times larger than that of the time series, we can suspect a large spike. If any of these windows occur within the first 30 iterations, we can safely dismiss any performance difference as warmup time.

4.3 RQ3

It is already proven that change point detection can be applied to determine whether a benchmark is able to reach steady state [3]. Using this method, a number of benchmarks will be identified and sorted based on whether they reach steady state. Then, each benchmark will be visualized, displaying motifs/discords and the changepoints on the same graph, with the matrix profile below. The benchmarks will then be further sorted depending on the correlation between changepoint locations and extremities on the matrix profile. This is achieved using an automated scoring system.

For every changepoint in a benchmark fork, the system will check if it is within the window of the two motifs or the discord. If it is, a point will be counted. If there are more than 8 changepoints, a point is removed, with an extra half point removed for every changepoint after. A benchmark with a large number of changepoints may have more changepoints fall within motif windows simply by coincidence; the removal of points lessens the impact of such coincidences on the overall results. A score of less than 1 implies no evident connection, with such benchmarks being designated as having 'No correlation'. A benchmark with a score between 1 and 2 can be designated as 'Potentially correlated'. A benchmark with a score of 2 or higher is classified as having a 'Noticeable correlation'.

5 RESULTS

5.1 RQ1

t-Test: Two-Sample Assuming Unequal Variances

	Non-Steady min	*Steady min*
Mean	11.95010176	6.23967312
Variance	22.50295536	29.1739795
t Stat	14.52995687	
P(T<=t) two-tail	2.34256E-40	
t Critical two-tail	1.964563095	

t-Test: Two-Sample Assuming Unequal Variances

	Non-Steady max	*Steady max*
Mean	20.84155984	18.9595669
Variance	1.662824009	7.93059698
t Stat	12.40065945	
P(T<=t) two-tail	3.53787E-32	
t Critical two-tail	1.963223447	

t-Test: Two-Sample Assuming Unequal Variances

	Non-Steady mean	*Steady mean*
Mean	17.15208272	11.82549433
Variance	8.637659801	36.99576179
t Stat	15.97379716	
P(T<=t) two-tail	1.75515E-49	
t Critical two-tail	1.963218974	
Observations	235	497

Figure 3: Results of T-Tests for comparison between steady state and non-steady state Matrix Profiles

The results of the described tests are shown in *Fig. 3*. In all cases, the P value is less than the alpha (0.05) and the T stats are larger than the T critical values, so we reject the null hypothesis (0 difference). As such, it can be concluded that there is a meaningful difference between the average values observed in the benchmarks which reached steady state, and those which didn't. Benchmarks which reach steady state have matrix profiles with lower minimums, maximums and means than benchmarks which do not.

5.2 RQ2

The matrix profile is able to recognize isolated anomalies and spikes, an example of which can be seen in *Fig. 4*.

However, a major downside is that there is significant potential for blind spots with this method. If a majority of the benchmark is in an anomalous state, using the mean to identify anomalies would result in none being found, as the mean under the anomaly dominates the overall mean and standard deviation. Another potential problem stemss from the matrix profile increasing and decreasing based on similarity; it may prioritize spikes over sustained increases in performance when both are present. This effect, as shown in *Fig. 6*, could be lowered by increasing the sizes of the comparison windows, or increasing the number of discords to analyse. In summary, this detection method shows some promise, but likely requires some tweaking to mitigate its shortcomings and achieve useful results. Future tests should compare the results of using different comparison windows for the same benchmark, different classification functions, and/or by increasing the number of motifs/discords to be analysed.

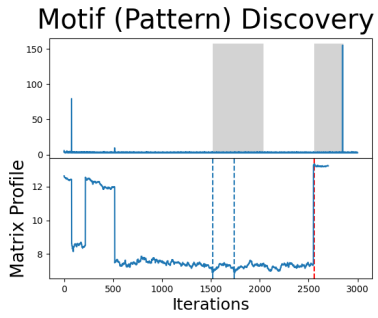

Figure 4: An example of a detected spike at the discord, marked by the red line.

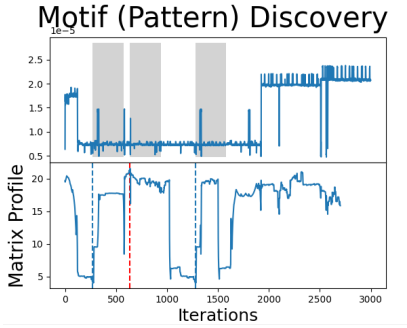

Figure 5: An example of a situation where minor spikes are detected, but other anomalous behaviour is not.

Correlation	Steady State	Steady State (%)	No Steady State	No SS (%)
None	123	40.32%	44	23.78%
Potential	123	40.32%	93	50.27%
Noticeable	59	19.34%	48	25.94%
Total	305		185	

Table 1: Comparisons of JMH Benchmark types based on correlation of matrix profile and changepoints

5.3 RQ3

As depicted in table 1, the matrix profile and changepoints have a stronger correlation in the benchmarks which do not reach a steady state. It should be noted that the increase in the percentage of noticeable correlations is only relatively minor, and most of the increase is in the potential correlation category. As a result, it would likely require a more comprehensive study to determine the exact relationship between the two measurements.

6 THREATS TO VALIDITY

Coding

It is possible that, in translating the steady state detection function described by Barrett *et al.* [3] to Python, there were some mistakes made and/or inconsistencies (different changepoint detection methods and/or storage of changepoints) that affected the results of classifications.

Evaluation

We note that some of our evaluation methods are not directly indicative of a method's quality, and in the case of RQ2, merely qualitative.

7 CONCLUSIONS

The experiments show that, with some changes and improvements, the matrix profile and changepoint analysis methods show good potential in assessing the state of JMH Microbenchmarks. Although they have a good degree of versatility, it will take further research to determine whether they will be able to achieve results as desirable as the current industry standard.

ACKNOWLEDGMENT

The support of the Natural Sciences and Engineering Research Council of Canada (NSERC) is gratefully acknowledged.

REFERENCES

[1] L. Traini, V. Cortellessa, D. Di Pompeo, and M. Tucci, "Towards effective assessment of steady state performance in Java software: Are we there yet?," Empirical Software Engineering, vol. 28, no. 1, 2022. https://doi.org/10.1007/s10664-022-10247-x

[2] 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. https://doi.org/10.1080/01621459.2012.737745

[3] E. Barrett, C. F. Bolz-Tereick, R. Killick, S. Mount, and L. Tratt, "Virtual machine warmup blows hot and cold," Proceedings of the ACM on Programming Languages, vol. 1, no. OOPSLA, pp. 1–27, 2017. https://doi.org/10.1145/3133876

[4] S. M. Law, 'STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining', The Journal of Open Source Software, vol. 4, no. 39, p. 1504, 2019.

[5] C. Truong, L. Oudre, N. Vayatis, "Selective review of offline change point detection methods," Signal Processing, 167:107299, 2020.

[6] C.-C. M. Yeh, Y. Zhu, L. Ulanova, N. Begum, Y. Ding, H. A. Dau, D. F. Silva, A. Mueen, and E. Keogh, "Matrix profile I: All pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets," 2016 IEEE 16th International Conference on Data Mining (ICDM), 2016.