Environment. Technology. Resources. Rezekne, Latvia
Proceedings of the 16th International Scientific and Practical Conference. 2025. Volume II, 375-382

Data Centre Monitoring Model Utilizing Artificial Intelligence, Machine Learning and Anomaly Detection Algorithms

Roberts Volkovičs
Vidzeme University of Applied Sciences
Valmiera, Latvia
roberts.volkovics@va.lv

***Abstract* - In this article the review is created of architectures of popular data centre monitoring tools and corresponding information processing techniques are summarised. Pros and cons analysis of the monitoring tools is done and novel approach is offered by utilizing Artificial Intelligence (AI), Machine Learning (ML) and Anomaly Detection (AD) algorithms to achieve research goals and prove hypothesis that data centre level monitoring model could be built using combined AI, ML and AD techniques. Oracle performance metric data are collected to perform the information analysis from such angles the most modern enterprise monitoring tools do not provide yet.**

Keywords — Data Centre, Monitoring, Tools and Techniques, Artificial Intelligence, Machine Learning, Anomaly Detection, Algorithms, Model.

I. INTRODUCTION

As the first enterprise grade data centre monitoring tool Oracle Enterprise Manager or Cloud Control will be described in the review.

In Oracle documentation [1] we can find the architecture diagram of Oracle Enterprise manager, see “Fig. 1”.

As diagram shows, Oracle Enterprise Manager consists of several components, such as “Oracle Database Management Repository”, “Oracle Management Service or OMS”, “JVMD Engine”, “Cloud Control Console” and “Oracle Management Agents with Plug-ins”.

Each of the mentioned components serves for dedicated purpose. For example, The Oracle Management Repository (Management Repository) is a storage location where all the information collected by the Management

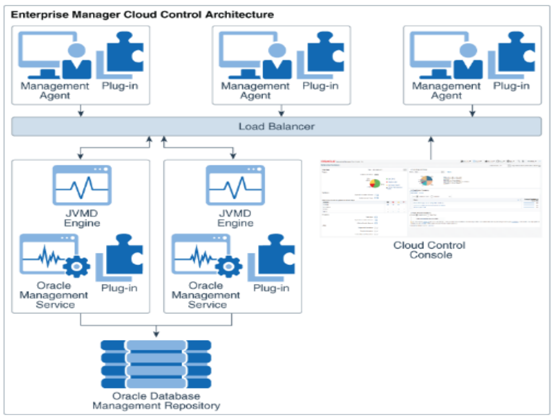


Fig. 1 Oracle Enterprise Manager architecture.

Agent gets stored. It consists of objects such as database jobs, packages, procedures, views, and tablespaces [1].

The OMS uploads the monitoring data it receives from the Management Agents to the Management Repository. The Management Repository then organizes the data so that it can be retrieved by the OMS and displayed in the Enterprise Manager Cloud Control console. Since data is stored in the Management Repository, it can be shared between any number of administrators accessing the Enterprise Manager Cloud Control [1].

Oracle Enterprise Manager is one of the best monitoring tools in the market with very rich built-in functionality to monitor Oracle software products starting from Oracle database, Oracle Linux operating system, Oracle Middleware and many more.

Oracle Enterprise manager as we can see in “Fig. 1” has as well a rich offer of different Plug-ins which support

Online ISSN 2256-070X
https://doi.org/10.17770/etr2025vol2.8565

the monitoring of products from other vendors, for example, IBM’s Db2 database.

In case one would like to have a rich set of standard monitoring functionality for Oracle centric environment then Oracle Enterprise manager or Cloud Control could be very sound choice to star with.

In “Fig. 2” we can see Zabbix monitoring tool architecture diagram [2].

Fig. 2 Zabbix architecture.

Zabbix monitoring tool from practical perspective is much more suitable for general purpose monitoring if we compare it to Oracle Enterprise Manager. Zabbix monitoring tool consists of such components as: “Server”, “Agent”, “Agent 2”, “Proxy”, “Java gateway”, “Sender”, “Get”, “Js” and “Web service”.

The server performs the polling and trapping of data, it calculates triggers, sends notifications to users. It is the central component to which Zabbix agents and proxies report data on availability and integrity of systems [2].

The agent gathers operational information locally and reports data to Zabbix server for further processing. In case of failures (such as a hard disk running full or a crashed service process), Zabbix server can actively alert the administrators of the machine that reported the failure [2]. Zabbix agent 2 is a new generation of Zabbix agent and may be used in place of Zabbix agent [2].

Zabbix proxy is a process that may collect monitoring data from one or more monitored devices and send the information to the Zabbix server, essentially working on behalf of the server. All collected data is buffered locally and then transferred to the Zabbix server the proxy belongs to [2].

From the description of Zabbix components above we can conclude that from the architecture perspective Oracle Enterprise Manager and Zabbix are similar if we emphasize facts that both monitoring tools have centralized server which could be configured in cluster mode for high availability and both monitoring tools utilize agents for data collection from remote targets. For sure Zabbix as a general-purpose monitoring tool has less built in functionality for Oracle product monitoring, but it has the Oracle template for “Agent 2” which could very well be used for Oracle database monitoring [3].

In “Fig. 3” we can see architecture diagram of Cacti monitoring tool [4].

As one can notice Cacti has the simplest architecture diagram from the all three tools mentioned so far. It basically has only two components: “Cacti server” and “server to monitor”. In reality “Cacti server” is to some extent analogical to Oracle Enterprise Manager and Zabbix from the perspective that all three of the mentioned have relational database as persistent data store and browser based graphical user interface.

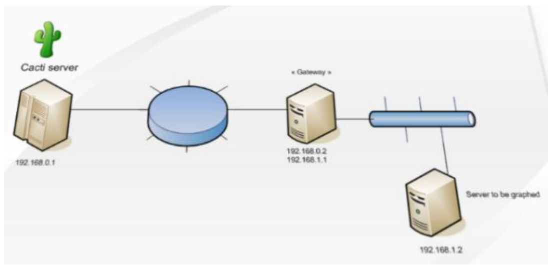


Fig. 3 Cacti architecture.

However, Cacti monitoring tool is different from perspective that it has no or we can say it has much less built-in functionality for specific vendor product monitoring and it is agent less. It as well has plugins.

Cacti monitoring tool differs from the perspective of how it stores the collected data. Cacti utilizes RRDTools Round Robin Archives (RRAs) as data sources for graph generation and that is the unique feature of Cacti. As it is stated by Cacti documentation, Cacti is a robust performance and fault management framework and a frontend to RRDTool - a Time Series Database (TSDB) [4].

Additional information on RRDTool could be found from the RRDTool documentation [5], but the main difference here is that data from monitored devices are stored in the RRA files and this file format is dedicated to store time series data, in this case database serves more as information processing tool.

We can mention other monitoring tools here such as Nagios [6], for example, but we already have figured out some pattern, that monitoring tools are similar at some extent as they have the same goal to store and process time series data about monitored objects which could be different software metrics from different vendor products or sensor data from IOT devices. Some of the monitoring tools will focus on support of specific vendor products, for example, Oracle Enterprise Manager, but some of the monitoring tools focus especially on the general-purpose monitoring tasks.

It is obvious that monitoring tools with the main aim to support specific vendor products or to support general purpose monitoring could not have built in functionality for all sorts of real-life scenarios and both types support some level of customization.

In the rest of this article the novel approach of data centre monitoring is described for which the author of the article has created the model based on AI algorithms, machine learning and anomaly detection techniques allowing to focus more on monitoring at data centre level rather than monitoring at specific target level in the data centre. The model will consider existing monitoring targets and human behaviour as the targets are getting monitored for metrics already. Each of the target metric could be displayed in more than one resulting graph and graphs could be created by different persons. This will serve as a weight factor for metrics as metrics get displayed greater number of times in regular graphs of individual monitoring targets as more weight those metrics have in graphs reporting the overall situation in the data centre.

II. Materials and methods

For demonstration purposes of the model the attention is drawn on two of the Oracle database performance metrics “consistent_gets” and “db_block_gets”. The sum of "db_block_gets" plus "consistent_gets" is considered as logical reads of database blocks from either the buffer cache or process private memory [7].

In “Fig. 4” one can see the example of histograms for

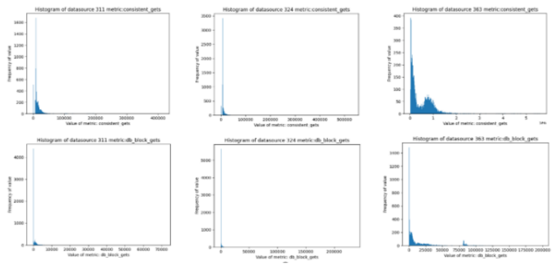


Fig. 4 Histograms of performance metrics related to "logical reads."

performance metrics “consistent_gets” and “db_block_gets” from three different data sources. One data source in “Fig. 4” represents on database in the data centre. In some cases, one database could be presented to different persons via different graphs visualizing different combinations of performance metrics. In such situations, one performance metric is part of different data sources, and it will serve as a weighting factor in the end results, as it could be considered as a factor of higher importance.

In the scope of this research hypothesis is considered that it is possible to construct a model that is based on AI algorithms, machine learning techniques, and anomaly detection techniques for data centre level monitoring of data centre systems. The purpose of data centre level monitoring of data centre systems is to provide novel ways of metric processing allowing to analyse the information from perspective which currently available tools in the market do not provide yet.

In “Fig. 4” it is visible, that chosen histograms of performance metrics “consistent_gets” and “db_block_gets” do not have a shape of normal distribution, uniform distribution, or some other popular distribution. Therefore, as good choice of anomaly detection techniques for processing these metrics naturally would be AI algorithms which support processing of data from timeseries performing anomaly detection utilizing techniques based on timeseries which are independent of data distributions. As there are plenty of monitoring targets and huge number of metrics, we will choose AI algorithms which support unsupervised machine learning.

To further construct the model, a set of algorithms were chosen which by theory are capable to solve the previously defined task: “Empirical Cumulative distribution-based Outlier Detection or ECOD”, “Local Outlier factor or LOF”, “Isolation Forest or IForest”, “Outlier Detection Using the Local Correlation Integral or LOCI”.

The premise of the Isolation Forest algorithm is that anomalous data points are easier to separate from the rest of the sample. To isolate a data point, the algorithm recursively generates partitions on the sample by randomly selecting an attribute and then randomly selecting a split value between the minimum and maximum values allowed for that attribute [8].

The local outlier factor is based on a concept of a local density, where locality is given by k nearest neighbours, whose distance is used to estimate the density. By comparing the local density of an object to the local densities of its neighbours, one can identify regions of similar density, and points, that have a substantially lower density than their neighbours, are outliers [9].

The existing unsupervised machine learning approaches often suffer from high computational cost, complex hyperparameter tuning and limited interpretability, especially when working with large, high-dimensional datasets. To address these issues, we present a simple yet effective algorithm called ECOD (Empirical Cumulative distribution-based Outlier Detection), which is inspired by the fact that outliers are often the "rare events" that appear in the tails of a distribution. In a nutshell, ECOD first estimates the underlying distribution of the input data in a nonparametric fashion by computing the empirical cumulative distribution per dimension of the data. ECOD then uses these empirical distributions to estimate tail probabilities per dimension for each data point. Finally, ECOD computes an outlier score of each data point by aggregating estimated tail probabilities across dimensions [10].

LOCI is highly effective for detecting outliers and groups of outliers (a.k.a. micro-clusters). In addition, it offers the following advantages and novelties: (a) It provides an automatic, data-dictated cut-off to determine whether a point is an outlier—in contrast, previous methods force users to pick cut-offs, without any hints as to what cut-off value is best for a given dataset. (b) It can provide a LOCI plot for each point; this plot summarizes a wealth of information about the data in the vicinity of the point, determining clusters, micro-clusters, their diameters, and their inter-cluster distances. None of the existing outlier-detection methods can match this feature, because they output only a single number for each point: its outlierness score. (c) Our LOCI method can be computed as quickly as the best previous methods [12].

Additionally, to theoretical aspects of previously mentioned AI algorithms we will use the Python PYOD library as the building block for the model. Python PYOD library has implementations of previously mentioned algorithms [13], [14], [15], [16], [17]. Implementations are done using other Python open-source projects such as Python scikit-learn [18].

The model for data centre level monitoring will use previously described Cacti monitoring tool as a basis and add to it the component based on AI algorithms and machine learning with capabilities to detect anomalies in the stream of monitoring data. Machine learning process will be done on historic data of performance metrics. Historic performance data are collected with the help of existing Cacti database model adjusting it as necessary to collect and store the data for longer term, by default Cacti database model keeps only small set of temporary data which could not be used by machine learning process. The architecture of the model is visible in "Fig. 5"

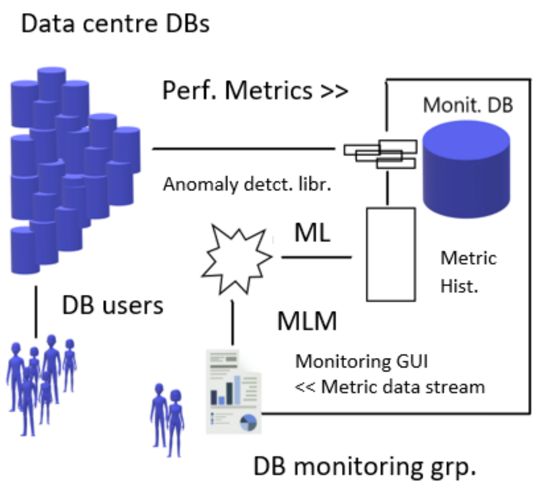


Fig. 5 Architecture of model "Data Centre monitoring".

As one can see in "Fig. 5" data centre hosts many databases (at the left side), metrics from database servers are collected and stored temporarily in the monitoring database (at the right side). Monitoring database holds historic data of incoming stream of monitoring metrics. Historic data are used as input for machine learning algorithms which analyse the data to create the modules for each metric individually which help later as basis in anomaly detection process in incoming data stream of monitoring metrics. As it is visible in "Fig. 4" such metrics as "consistent_gets" and "db_block_gets" may differ by such characteristics as the shape of the histogram of their historic data depending on the nature of the usage of the databases. That means we must have separate machine learning process and resulting module for each metric for every database.

Data structure of historical table for metrics "consistent_gets" and "db_block_gets" is visible in "Fig. 6".

Example of monitoring metric timeseries data displayed in descending order by values is visible in "Fig. 7".

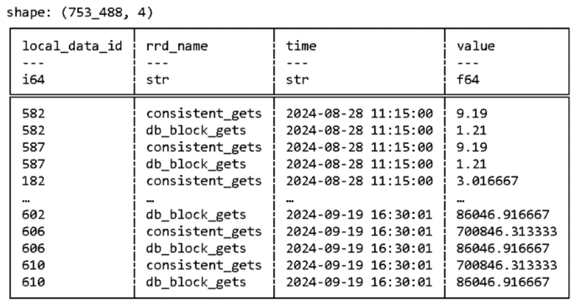

```
shape: (753_488, 4)
┌────────────────┬─────────────────┬─────────────────────┬───────────────┐
│ local_data_id  ┆ rrd_name        ┆ time                ┆ value         │
│ ---            ┆ ---             ┆ ---                 ┆ ---           │
│ i64            ┆ str             ┆ str                 ┆ f64           │
╞════════════════╪═════════════════╪═════════════════════╪═══════════════╡
│ 582            ┆ consistent_gets ┆ 2024-08-28 11:15:00 ┆ 9.19          │
│ 582            ┆ db_block_gets   ┆ 2024-08-28 11:15:00 ┆ 1.21          │
│ 587            ┆ consistent_gets ┆ 2024-08-28 11:15:00 ┆ 9.19          │
│ 587            ┆ db_block_gets   ┆ 2024-08-28 11:15:00 ┆ 1.21          │
│ 182            ┆ consistent_gets ┆ 2024-08-28 11:15:00 ┆ 3.016667      │
│ …              ┆ …               ┆ …                   ┆ …             │
│ 602            ┆ db_block_gets   ┆ 2024-09-19 16:30:01 ┆ 86046.916667  │
│ 606            ┆ consistent_gets ┆ 2024-09-19 16:30:01 ┆ 700846.313333 │
│ 606            ┆ db_block_gets   ┆ 2024-09-19 16:30:01 ┆ 86046.916667  │
│ 610            ┆ consistent_gets ┆ 2024-09-19 16:30:01 ┆ 700846.313333 │
│ 610            ┆ db_block_gets   ┆ 2024-09-19 16:30:01 ┆ 86046.916667  │
└────────────────┴─────────────────┴─────────────────────┴───────────────┘
```

Fig. 6 Structure of metric history table.

Using all the components described previously the model was developed in Python programming language in following steps:

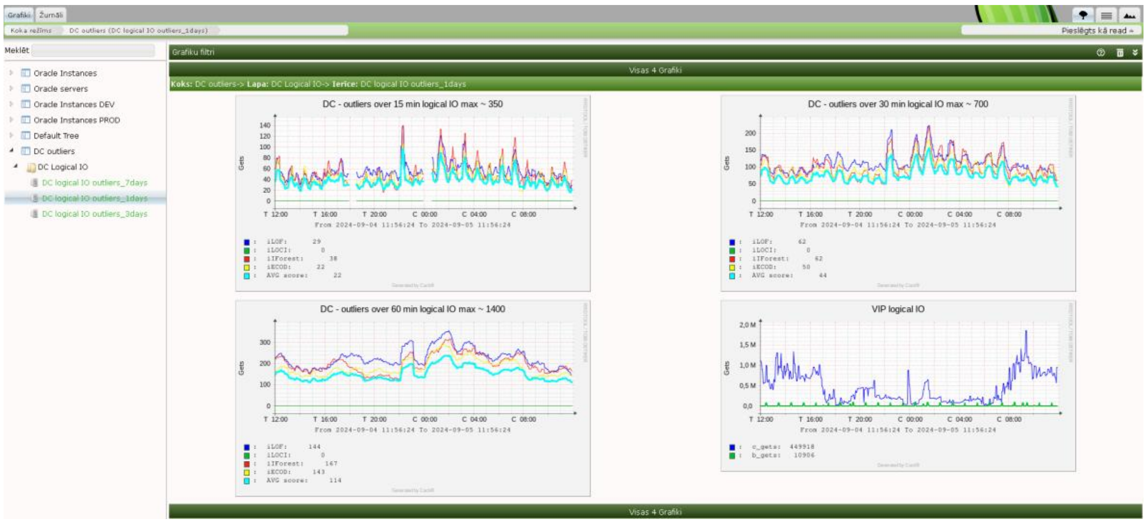
1. historical metric collection from incoming data stream (data point per 5 minutes for each metric);
2. unsupervised machine learning process for each metric (in total for 20 Oracle database) with:
    a. 1 day of data points (288)
    b. 3 days of data points (864);
    c. 7 days of data points (2016);
3. unsupervised machine learning process for each metric produced a module as output which could be used for real-time anomaly detection in incoming monitoring data stream:
    a. "1days" learning process modules;
    b. "3days" learning process modules;
    c. "7days" learning process modules;
4. all the modules: "1days", "3days" and "7days" are applied for each metric of data to produce comparable results;
5. the binary result of anomaly detection is summarized to gain overall results from all 20 databases for each metric in data centre.

III. Results and Discussion

Several novel results and approaches of monitoring and possibilities to gain even positive economic impact on data centre monitoring process were discovered during the one-month long period of using the model in practice in a data centre.

Example of outlierness level over data centre Oracle database performance metrics related to logical reads is shown in “Fig. 8” for “1days” unsupervised ML period.

Example of outlierness level over data centre Oracle database performance metrics related to logical reads is shown in “Fig. 9” for “3days” unsupervised ML period.


Grafiki
Žurnāli
Koka režīms
DC outliers (DC logical IO outliers_1days)
Pieslēgts kā read
Meklēt
Oracle Instances
Oracle servers
Oracle Instances DEV
Oracle Instances PROD
Default Tree
DC outliers
DC Logical IO
DC logical IO outliers_7days
DC logical IO outliers_1days
DC logical IO outliers_3days
Grafiku filtri
Visas 4 Grafiki
Koks: DC outliers-> Lapa: DC Logical IO-> Ierīce: DC logical IO outliers_1days
DC - outliers over 15 min logical IO max ~ 350
DC - outliers over 30 min logical IO max ~ 700
DC - outliers over 60 min logical IO max ~ 1400
VIP logical IO
Gets
T 12:00
T 16:00
T 20:00
C 00:00
C 04:00
C 08:00
From 2024-09-04 11:56:24 To 2024-09-05 11:56:24
Visas 4 Grafiki

DC - outliers over 15 min logical IO max ~ 350

🟦	iLOF:	29
🟩	iLOCI:	0
🟥	iIForest:	38
🟨	iECOD:	22
🟦	AVG score:	22

DC - outliers over 30 min logical IO max ~ 700

🟦	iLOF:	62
🟩	iLOCI:	0
🟥	iIForest:	62
🟨	iECOD:	50
🟦	AVG score:	44

DC - outliers over 60 min logical IO max ~ 1400

🟦	iLOF:	144
🟩	iLOCI:	0
🟥	iIForest:	167
🟨	iECOD:	143
🟦	AVG score:	114

VIP logical IO

🟦	c_gets:	449918
🟩	b_gets:	10906

Fig. 8 Outlierness level over Data Centre "1days".

Graphs in “Fig. 8” for “1days” learning period and “Fig. 9” for “3days” learning period display four sub graphs for period of the last day:

1. DC outlierness level for last “15 min” period for logical reads in scoring points out of maximum possible 350 scoring points;
2. DC outlierness level for last “30 min” period for logical reads in scoring points out of maximum possible 700 scoring points;
3. DC outlierness level for last “60 min” period for logical reads in scoring points out of maximum possible 1400 scoring points;
4. Example database “VIP” exact values for performance monitoring metrics “consistent_gets” and “db_block_gets” for period of the last day.

Monitoring for outlierness level and storing the results of the outlierness level for long term allows to represent the information visually and perform analysis and make further discussions finding options for research direction.

Obviously, it is possible to establish certain standards for outlierness level and react involving human experts in case outlierness level exceeds previously agreed thresholds.

In case of situation when outlierness level exceed thresholds, experts could figure out the causing reason for that and create, for example, knowledge base for further analysis of information and possible automation options of data centre management.

One option to establish thresholds for the level of outlierness is described in “RRDtool” documentation [5]. It is based on existing historic data and statistics methods built in “RRDtool” function “PREDICT”.

Prediction done using the function “PREDICT” about outlierness level that could be expected for next period of three days based on the outlierness level in previous seven days is shown in figures “Fig. 10” for outlierness scoring

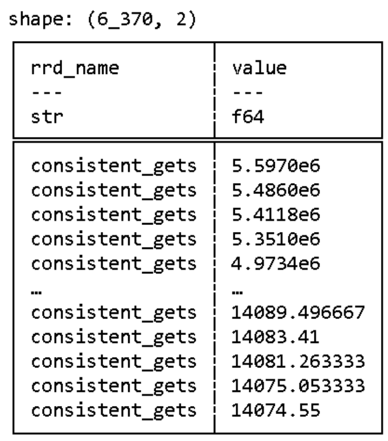
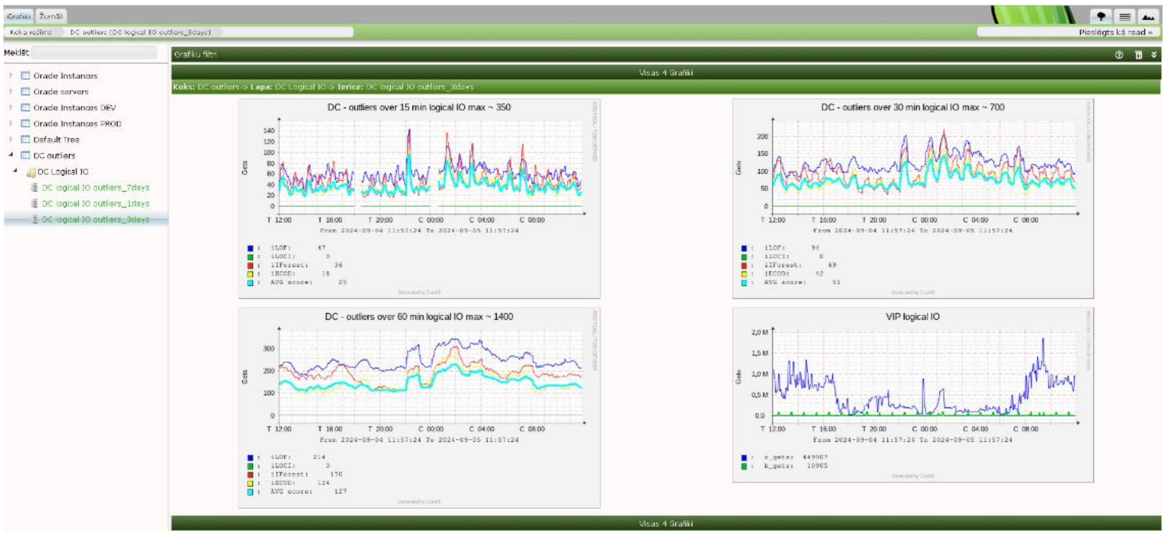

```
shape: (6_370, 2)
┌─────────────────┬──────────────┐
│ rrd_name        ┆ value        │
│ ---             ┆ ---          │
│ str             ┆ f64          │
╞═════════════════╪══════════════╡
│ consistent_gets ┆ 5.5970e6     │
│ consistent_gets ┆ 5.4860e6     │
│ consistent_gets ┆ 5.4118e6     │
│ consistent_gets ┆ 5.3510e6     │
│ consistent_gets ┆ 4.9734e6     │
│ …               ┆ …            │
│ consistent_gets ┆ 14089.496667 │
│ consistent_gets ┆ 14083.41     │
│ consistent_gets ┆ 14081.263333 │
│ consistent_gets ┆ 14075.053333 │
│ consistent_gets ┆ 14074.55     │
└─────────────────┴──────────────┘
```

Fig. 7 Example of monitoring metrics timeseries

done by algorithm “IForest”, “Fig 11” for outlierness scoring done by algorithm “LOF” and “Fig 12” for outlierness scoring done by algorithm “ECOD”. By the red colour (please, see the arrows in Fig. 10 when reading


Grafiki
Žurnāli
Grafiku filtri
Visas 4 Grafiki
Oracle Instances
Oracle servers
Oracle Instances DEV
Oracle Instances PROD
Default Tree
DC outliers
DC Logical IO
DC logical IO outliers_7days
DC logical IO outliers_1days
DC logical IO outliers_3days
Koks: DC outliers -> Lapa: DC Logical IO -> Ierīce: DC logical IO outliers_3days
DC - outliers over 15 min logical IO max ~ 350
DC - outliers over 30 min logical IO max ~ 700
DC - outliers over 60 min logical IO max ~ 1400
VIP logical IO
Gets
From 2024-09-04 11:57:24 To 2024-09-05 11:57:24

```
■ :  1LOF:       47
■ :  1LOCI:       0
■ :  1IForest:     36
■ :  1ECOD:      18
■ :  AVG score:     25
```

```
■ :  1LOF:       94
■ :  1LOCI:       0
■ :  1IForest:     69
■ :  1ECOD:      42
■ :  AVG score:     51
```

```
■ :  1LOF:      214
■ :  1LOCI:       0
■ :  1IForest:    170
■ :  1ECOD:     124
■ :  AVG score:    127
```

```
■ :  r_gets:  449987
■ :  b_gets:   10905
```

Fig. 9 Outlierness level over Data Centre "3days".

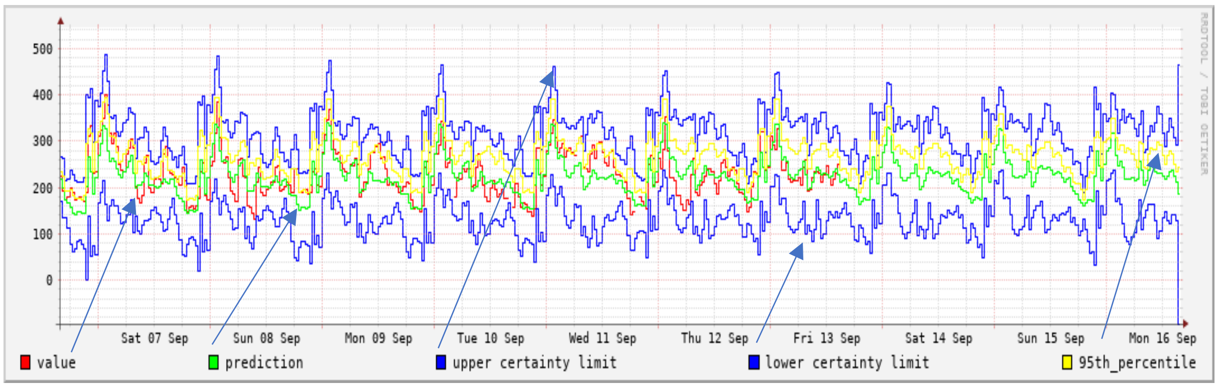


Fig. 10 Prediction "3days" IForest level of outlierness.

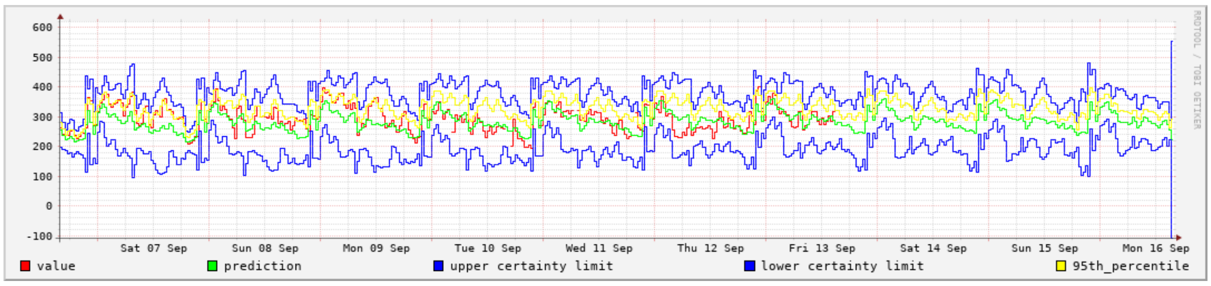


Fig. 11 Prediction "3days" LOF level of outlierness.

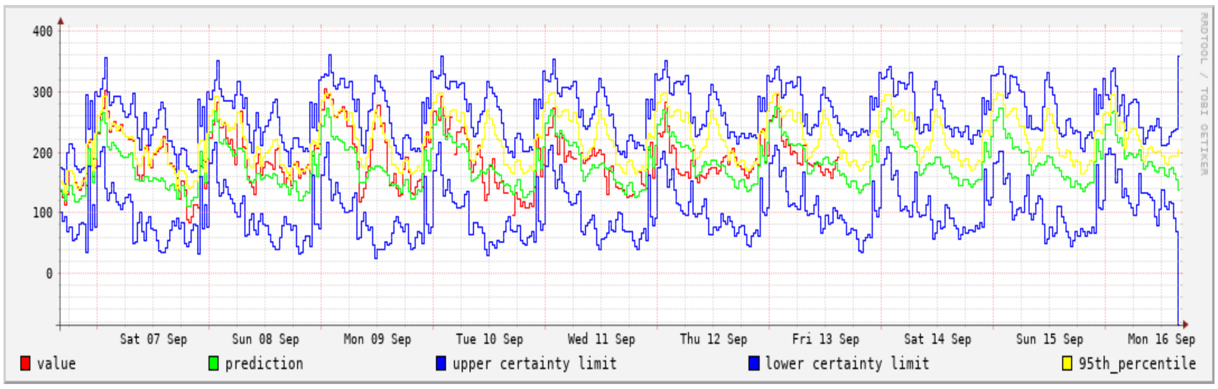


Fig. 12 Prediction "3days" ECOD level of outlierness.

printed grayscale variant) the current value of outlierness scoring is shown, by the green value predicted value is shown, blue colour shows lower and upper certainty levels and yellow shows 95th percentile. All colours except red are present in graphs 3 days after the current day of graph generation "Friday 13th of September" and could be considered as prediction. At the same time such prediction could be considered as a threshold level exceeding which human experts are involved to verify situation.

IV. Conclusions

The research hypothesis was proved as it is possible to construct a model based on AI algorithms, machine learning techniques and anomaly detection techniques for data centre level monitoring of data centre systems.

The results of model are visible in "Fig. 8" and "Fig. 9" which show us level of outlierness which is the result of processing Oracle performance metric timeseries data by AI algorithms using unsupervised ML techniques. Following algorithms were used during the research: "IForest", "LOF", "ECOD" and "LOCI" to achieve the results. One of the algorithms "LOCI" did not provide any anomaly scoring results, so we assume it either not good choice for our type of timeseries data or there is a bug in Python PYOD library. The other three: "IForest", "LOF", "ECOD" produced slightly different results for each particular data point, but in case we do not focus on particular data points, but on longer timeseries as it is visible in "Fig. 10", "Fig. 11" and "Fig. 12", the results are very similar and they provide similar predictions based on historic data. Additionally, it is worth to mention the performance of algorithms during unsupervised ML process. As previously mentioned, we can assume "LOCI" did not provide useful results in our case, from the rest "IForest" and "LOF" were significantly faster than "ECOD", due to that it was possible to complete unsupervised ML process for periods of "1days" and "3days" during reasonable amount of time. In the process of anomaly detection in real time data all three algorithms performed fast enough for the scope of the task.

Conclusion could be made, that it is possible to establish a knowledge base of cases when thresholds are exceeded and continue the work on actions advised by the experts on how to handle certain situations.

Outlierness level monitoring at data centre level could help as well to identify certain situations which are not identifiable by monitoring individual objects which could help to react faster on certain type of theoretically possible cases touching field of IT security where quite often experts deal with situations that have never happened before and that are not visible in the monitoring done by traditional approaches.

As it is visible in Oracle 21c documentation [7], there exist more than 500 performance metrics. In Oracle 21c documentation all the metrics are divided in following classes: "User", "Redo", "Enqueue", "Cache", "OS", "Real Application Clusters", "SQL", "Debug", "Instance level" [7]. For example, a class "SQL" represents a statistic that relates to SQL statements and caching. That provides wide field for further research based on the model developed in the scope of this research and author hopes to continue his work on it.

V. Acknowledgments

This research was funded by the European Commission, Research Executive Agency grant number 101079206, "Twinning in Environmental Data and Dynamical Systems Modelling for Latvia" (TED4LAT).

VI. References

[1] Oracle Enterprise Manager, "Enterprise Manager Cloud Control Architecture." *Copyright © 1995, 2024, Oracle and/or its affiliates*. [Online]. Available: https://docs.oracle.com/en/enterprise-manager/cloud-control/enterprise-manager-cloud-control/13.5/emcon/enterprise-manager-cloud-control-architecture.html [Accessed: Sep. 19, 2024].

[2] Zabbix documentation, "Zabbix features." *Policy© 2001-2024 by Zabbix LLC. All rights reserved.* [Online]. Available: https://www.zabbix.com/features [Accessed: Sep. 19, 2024].

[3] Zabbix documentation, "Zabbix integrations Oracle." *Policy© 2001-2024 by Zabbix LLC. All rights reserved.* [Online]. Available: https://www.zabbix.com/integrations/oracle [Accessed: Sep. 19, 2024].

[4] Cacti documentation, "What is Cacti?" *Copyright © 2004-2021 The Cacti Group, Inc. - Cacti is the registered trademark The Cacti Group, Inc.* [Online]. Available: https://www.cacti.net/info/cacti [Accessed: Sep. 19, 2024].

[5] RRDtool documentation, "RRDtool documentation" *OETIKER+PARTNER AG* [Online]. Available: https://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html [Accessed: Sep. 19, 2024].

[6] Nagios documentation, "Nagios documentation" *Copyright © 2009-2023 Nagios Enterprises, LLC. All rights reserved.* [Online]. Available: https://www.nagios.org/documentation/ [Accessed: Sep. 19, 2024].

[7] Oracle Database Reference 21, "Statistics Descriptions." *Copyright © 1995, 2024, Oracle and/or its affiliates*. [Online]. Available: https://docs.oracle.com/en/database/oracle/oracle-database/21/refrn/statistics-descriptions-2.html [Accessed: Sep. 20, 2024].

[8] Wikipedia, "Isolation forest," [Online] Available: https://en.wikipedia.org/wiki/Isolation_forest [Accessed: Sep. 20, 2024].

[9] Wikipedia, "Local outlier factor," [Online] Available: https://en.wikipedia.org/wiki/Local_outlier_factor [Accessed: Sep. 20, 2024].

[10] Zheng Li, Yue Zhao, Xiyang Hu, Nicola Botta, Cezar Ionescu, George H. Chen, "ECOD: Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions, " 2022, https://arxiv.org/abs/2201.00382

[11] Wikipedia, "Empirical distribution function," [Online] Available: https://en.wikipedia.org/wiki/Empirical_distribution_function [Accessed: Sep. 20, 2024].

[12] S. Papadimitriou, H. Kitagawa, P. B. Gibbons and C. Faloutsos, "LOCI: fast outlier detection using the local correlation integral," Proceedings 19th International Conference on Data Engineering (Cat. No.03CH37405), Bangalore, India, 2003, pp. 315-326, doi: 10.1109/ICDE.2003.1260802.

[13] Pyod 2.0.2 documentation, "Python library for detecting anomalous/outlying objects in multivariate data," [Online] Available: https://pyod.readthedocs.io/en/latest/ [Accessed: Sep. 20, 2024].

[14] pyod.models.lof module, "Local Outlier Factor (LOF). Implemented on scikit-learn library," [Online] Available: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.lof [Accessed: Sep. 20, 2024].

[15] pyod.models.loci module, "Local Correlation Integral (LOCI)," [Online] Available: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.loci [Accessed: Sep. 20, 2024].

[16] pyod.models.iforest module, "IsolationForest Outlier Detector. Implemented on scikit-learn library," [Online] Available: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.iforest [Accessed: Sep. 20, 2024].

[17] pyod.models.ecod module, "Unsupervised Outlier Detection Using Empirical Cumulative Distribution Functions (ECOD)," [Online] Available: https://pyod.readthedocs.io/en/latest/pyod.models.html#module-pyod.models.ecod [Accessed: Sep. 20, 2024].

[18] scikit-learn, "scikit-learn machine learning library," [Online] Available: https://scikit-learn.org/ [Accessed: Sep. 20, 2024].