B. König-Ries et al. (Hrsg.): Datenbanksysteme für Business, Technologie und Web (BTW 2023),
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 1

Workload Prediction for IoT Data Management Systems

David Burrell[1] Xenofon Chatziliadis[2] Eleni Tzirita Zacharatou[3] Steffen Zeuch[4] Volker Markl[5 6]

Abstract: The Internet of Things (IoT) is an emerging technology that allows numerous devices, potentially spread over a large geographical area, to collect and collectively process data from high-speed data streams. To that end, specialized IoT data management systems (IoTDMSs) have emerged. One challenge in those systems is the collection of different metrics from devices in a central location for analysis. This analysis allows IoTDMSs to maintain an overview of the workload on different devices and to optimize their processing. However, as an IoT network comprises of many heterogeneous devices with low computation resources and limited bandwidth, collecting and sending workload metrics can cause increased latency in data processing tasks across the network.

In this ongoing work, we present an approach to avoid unnecessary transmission of workload metrics by predicting CPU, memory, and network usage using machine learning (ML). Specifically, we demonstrate the performance of two ML models, linear regression and Long Short-Term Memory (LSTM) neural network, and show the features that we explored to train these models. This work is part of an ongoing research to develop a monitoring tool for our new IoTDMS named NebulaStream.

Keywords: Internet of Things; stream processing; machine learning; workload prediction

1 Introduction

The Internet of Things (IoT) describes a distributed system in which a large number of devices with sensing or processing capabilities communicate with each other [PPS16]. The IoT enables new possibilities for applications that have led to advances in many fields, including healthcare [Ab19], disaster management [OTM21], and smart cities [Mo21].

[1] TU Berlin, Database Systems and Information Management, Einsteinufer 17, 10587 Berlin, Germany d. burrell736@googlemail.com
[2] TU Berlin, Database Systems and Information Management, Einsteinufer 17, 10587 Berlin, Germany x. chatziliadis@tu-berlin.de
[3] IT University of Copenhagen, Data-intensive Systems and Applications, Rued Langgaards Vej 7, DK-2300 Copenhagen S, Denmark elza@itu.dk
[4] German Research Center for Artificial Intelligence (DFKI), Alt-Moabit 91c, 10559 Berlin, Germany steffen. zeuch@dfki.de
[5] TU Berlin, Database Systems and Information Management, Einsteinufer 17, 10587 Berlin, Germany volker. markl@tu-berlin.de
[6] German Research Center for Artificial Intelligence (DFKI), Alt-Moabit 91c, 10559 Berlin, Germany Volker. Markl@dfki.de

To process data in real time in such a distributed environment, stream processing engines (SPEs) are often used. SPEs enable the continuous execution of user-defined queries on data streams to extract useful information. SPEs are designed to execute these queries across distributed devices optimally, in terms of the placement of the data processing operations and the management of the information flow. However, SPEs are designed for the cloud, which has significantly different characteristics compared to the IoT.

For system-internal decision-making processes, such as load balancing and query optimization, existing SPEs often require performance metrics from their topology. In an IoT topology, the collection and sending of these metrics can be detrimental to the overall data processing efficiency. Many devices have few computational resources, and the collection of workload metrics leads to reduced processing performance [Ch21, CFSF20]. Additionally, the collection process affects the network, since it requires a large amount of bandwidth to send the collected metrics and therefore introduces additional latency [SJL18]. Finally, the additional processing and network efforts required for the metric collection process reduce the operational lifetime of battery-powered devices [Ma05].

To mitigate these issues, it would be beneficial for an SPE to be able to infer workload metric values and avoid the need to collect these metrics continuously. This would reduce the load on the workers themselves as well as the volume of data being sent through the network, thus decreasing the response time for user queries. In this work, we show our current progress in testing the suitability of machine learning techniques to predict the workloads of the NebulaStream (NES) IoTDMS [Ze20a, Ze20b]. Using a small-scale lab topology with five devices, we have trained a linear regression and an LSTM model to predict the CPU, memory, and network utilization of different query workloads. We show that for all workloads, our linear regression models significantly outperform the LSTMs with a PRED 25 score (i.e., percentage of predicted values that are within 25% of the actual value) between 56% and 86%.

The remainder of this paper is structured as follows: In Section 2, we analyze the different workloads and features for our ML models. In Section 3, we evaluate the suitability of linear regression and LSTM to predict CPU, memory, and network utilization. We conclude our work in Section 4 and discuss future work directions.

2 Methodology

Data collection and data engineering steps are of essential importance to produce adequate training data sets for accurate ML models. To this end, we create an emulated NES topology where we execute specific queries on predefined data to generate workload information. During the execution of these queries, we collect metrics on CPU, memory, and network utilization. We collect these metrics along with additional features about the topology that can be used to train our ML models. In the remainder of this section, we describe in detail

how we generate different workloads in NES and the features we investigate to train our ML models.

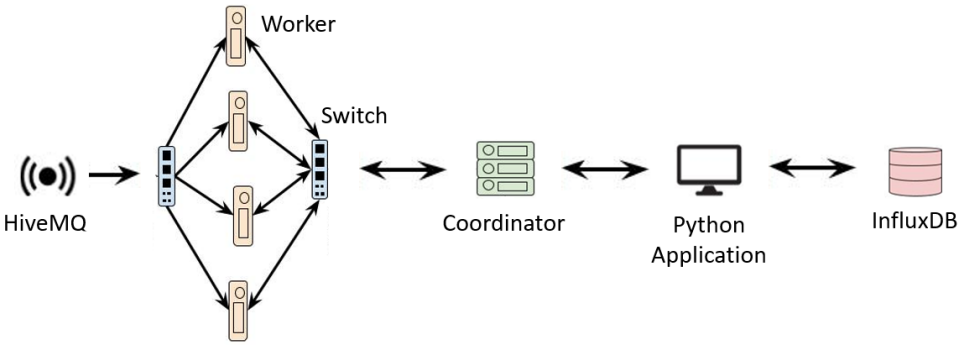

Fig. 1: Overview of the emulated topology in NebulaStream.

2.1 Workload Generation

For generating training load data we create a controlled NebulaStream lab topology with Docker containers [Me14]. Communication between containers is established using the Containernet network emulation [PKvR16]. In total, the topology consists of one coordinator, four workers, and an MQTT broker (cf. Fig. 1). The MQTT broker (HiveMQ) has four topics, one for each worker to subscribe to and where data is pushed in regular intervals. The data that is pushed to the workers is generated using the humidity and temperature sensors data set of the UCI Machine Learning Repository [Hu16]. The workers are used to process and send the data to the coordinator. In order to represent the heterogeneity of the IoT environment in our lab topology, we assign each worker different resources in terms of available memory, memory swap limit, and CPU shares. Regarding the distribution of resources, worker number zero had the fewest available resources with a memory limit of 10Mb, a memory swap limit of 30Mb, and a CPU share of 20%. For each worker, we incrementally increased then the resources, such that worker number three has 40Mb memory limit, 120Mb memory swap limit, and 80% CPU shares.

The coordinator is the central component of NebulaStream, which is responsible for processing user requests, scheduling queries, and managing the life cycle of running queries. We used the coordinator as a sink for our streaming queries and an endpoint for our queries. To generate different load types, we deployed four queries to NES, 1) a select-all query, 2) a projection, 3) a filter with a selectivity of 0.77, and 4) a user-defined function with the map operation. These queries were deployed from our Python application via the REST interface of NES. Workload metrics were collected through the NES monitoring API and the docker metrics API while running the queries. For analysis and feature exploration, we stored all workload metrics in the InfluxDB key-value store.

2.2 Feature Selection

For training accurate ML models, we require feature variables that have some relationship with the target variables, i.e., the workload metrics. In general, the stronger the relationship between the features and the target, the higher the model's prediction accuracy. We have analyzed two different types of features.

The first type are static features, where the value does not change over time. The static features collected from the emulated topology are: the worker resource limits, the tuples per second received at the workers, the executed queries, and their operators. The second type of features are time-series features that we divide into two classes. The first class belongs to the independent variables that are recorded separately to the workload metric streams. These variables are the number of nanoseconds since the query began, the data received at the coordinator from each worker, and the coordinator's CPU usage. The second set of time-series features are the metric streams of the workloads.

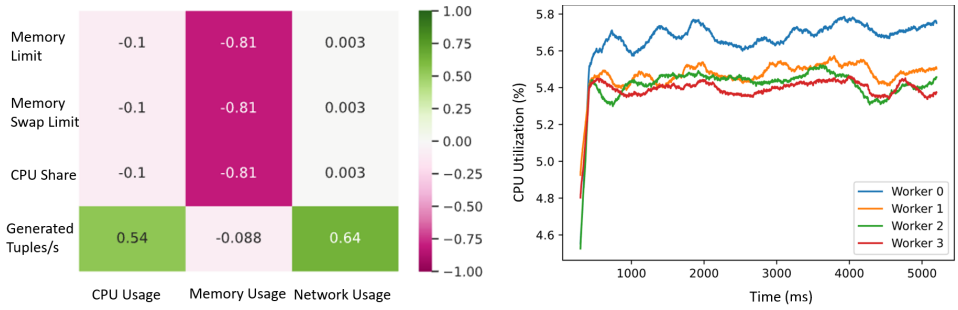

Fig. 2: (left) Heatmap of the SRCC between the static variables and the workload metrics. (right) The moving average of the CPU usage during the execution of the ßelect all"query.

To identify important candidates, we use Pearson and Spearman Rank Correlation Coefficient (SRCC) to determine if a relationship exists between potential features and our target variables CPU, memory, and network usage. Due to space limitations, we highlight here the most influential features, which can be seen based on SRCC between the resource limits (i.e., memory, memory swap, and CPU share) and memory usage, with a particularly strong negative correlation (cf. Fig. 2 (left)). The reason is that workers with the fewest resources available to process the arriving data are required to store the data for a longer period before the data can be sent to the coordinator.

By looking at the moving average of the values over time, it is possible to see the effect of having different percentages of CPU for each worker (cf. Fig. 2 (right)). For the moving average, we use a window size of 300ms and a sampling period of 20ms. It can be seen that worker number zero has the most CPU usage due to having the smallest share of CPU and requiring more time to complete tasks, while the other workers (which were closer together in value) followed in order of increasing percentage of CPU share.

In summary, the patterns we observe via SRCC and the moving average plot confirm that available resources on the workers have a huge effect on the workload and are thus important features that should be taken into account when training an ML model.

3 Evaluation

For each load metric, i.e., CPU, memory, and network, we train a linear regression and LSTM model, which results in six models in total. We evaluate the models using the PRED 25 score, which measures the percentage of predicted values that are within 25 percent of the actual value (cf. Table 1). The models are trained on 80 multivariant time series with 246.883 different samples, and for validation on 246.883 samples. Overall, the linear regression model (LR) is more accurate at predicting the three workload metrics than the LSTM model, which is similar to the results reported by [AB16]. In the remainder of this section, we will give a more detailed discussion of our results and findings.

Query Type	**CPU Model**		**Memory Model**		**Network Model**	
	LR	*LSTM*	*LR*	*LSTM*	*LR*	*LSTM*
Select all	98.53	87.98	0.15	57.11	89.15	16.68
Projection	29.00	20.65	69.13	42.59	83.23	15.39
Filter	94.94	39.83	76.59	43.23	89.26	28.17
Map	49.20	39.45	89.85	36.94	91.97	22.80
Total Mean	73.07	50.32	58.87	46.13	86.34	28.55

Tab. 1: PRED 25 score for each of the machine learning models.

3.1 CPU Model

The linear regression CPU model achieves, on average, for all query types a PRED 25 score of 73.07%, which significantly outperforms the LSTM model that reaches only 50.32%. Looking at the scores of individual query types, we can see that the linear regression performs extremely well for the select all and filter query type with a score over 90%. On the contrary, for the projection and map query, our models achieve a score below 50%. For the latter our models had captured in general the trends of the curves. However, they predicted incorrectly the starting values which consequently lead to wrong forecasts of the subsequent values.

3.2 Memory Model

For the memory model we reach with the linear regression in total a mean score of around 59% and the LSTM around 46%. Looking at the different query types individually we can see, on the one hand, that the performance of the select all query is the biggest detractor for the linear regression model with a score below 1%. During the execution of the projection, filter and map query, we could observe often spikes in the curves regarding memory consumption. However, for the select all query a spiking of memory consumption does not happen, as the internal memory management of NES is keeping for that particular case always the same amount of data tuples in memory. We believe that the performance of the linear regression can be improved here by adding more information about the query type to the ML model. For the projection, filter and map query, on the other hand, the linear regression achieved a performance of ca. 70% up to 90%. The LSTM model performed for the prediction of memory utilization again worse for all query types with a score between 37% and 57%.

3.3 Network Model

The linear regression model is performing consistently well for all different query types with a PRED 25 score between 83% and 92% and an average score of 86.34%. The LSTM on the contrary reaches only scores between ca. 15% and 28%. During training, we had observed that the LSTM model was having problems with over fitting, as the prediction performance on the training data set was always larger than 90%. In future work, we will further investigate the reasons why the LSTM was over fitting that extremely for predicting network utilization.

4 Conclusion and Future Work

This paper explores the applicability of ML approaches to predict workload characteristics in streaming IoT environments. Our early results show that for a particular small-scale lab topology, workloads can be estimated with a PRED 25 score of up to 86% by using a linear regression model. However, to completely avoid direct value collection from the devices in the topology, further work is needed to increase the prediction accuracy. In future work, we plan to incorporate topological traits and create workload data on the basis of more complex queries. In our current work, we tested our models only against a single data set that was generated on a static lab topology. Additionally, we plan to integrate our approach of workload prediction to NebulaStream, in order to evaluate the general benefits.

Finally, we also plan to investigate different query optimization and operator placement strategies. By examining the operators that are deployed on each worker we hope to be able to identify a pattern in which cases the static and time-series characteristics can replace the continuous collection of performance measures and why in other cases this does not work.

Acknowledgments

This work has received funding by the German Ministry for Education and Research as BIFOLD - Berlin Institute for the Foundations of Learning and Data (ref. 01IS18025A, 01IS18037A) and from the European Union's H2020 research and innovation programme under grant agreement No. 957286.

References

[AB16] Ajila, Samuel; Bankole, Akindele: Using Machine Learning Algorithms for Cloud Client Prediction Models in a Web VM Resource Provisioning Environment. Transactions on Machine Learning and Artificial Intelligence, 2016.

[Ab19] Abdellatif, Alaa; Mohamed, Amr; Chiasserini, Carla-Fabiana; Tlili, Mounira; Erbad, Aiman: Edge Computing For Smart Health: Context-aware Approaches, Opportunities, and Challenges. IEEE Network, 2019.

[CFSF20] Cid-Fuentes, Javier Álvarez; Szabo, Claudia; Falkner, Katrina E.: Adaptive Performance Anomaly Detection in Distributed Systems Using Online SVMs. IEEE Transactions on Dependable and Secure Computing, 2020.

[Ch21] Chatziliadis, Xenofon; Tzirita Zacharatou, Eleni; Zeuch, Steffen; Markl, Volker: Monitoring of stream processing engines beyond the cloud: an overview. Open Journal of Internet Of Things (OJIOT), 2021.

[Hu16] Huerta, Ramon; Mosqueiro, Thiago; Fonollosa, Jordi; Rulkov, Nikolai F; Rodriguez-Lujan, Irene: Online decorrelation of humidity and temperature in chemical sensors for continuous monitoring. Chemometrics and Intelligent Laboratory Systems, 2016.

[Ma05] Madden, Samuel R.; Franklin, Michael J.; Hellerstein, Joseph M.; Hong, Wei: TinyDB: An acquisitional query processing system for sensor networks. ACM Transactions on database systems (TODS), 2005.

[Me14] Merkel, Dirk: Docker: lightweight linux containers for consistent development and deployment. Linux journal, 2014.

[Mo21] Moreno-Bernal, Pedro; Cervantes-Salazar, Carlos Alan; Nesmachnow, Sergio; Hurtado-Ramírez, Juan Manuel; Hernández-Aguilar, José Alberto: Open-Source Big Data Platform for Real-Time Geolocation in Smart Cities. Ibero-American Congress of Smart Cities, 2021.

[OTM21] Ouro Paz, Elena Beatriz; Tzirita Zacharatou, Eleni; Markl, Volker: Towards Resilient Data Management for the Internet of Moving Things. In: Datenbanksysteme für Business, Technologie und Web (BTW). volume P-311 of LNI, pp. 279–301, 2021.

[PKvR16] Peuster, M.; Karl, H.; van Rossem, S.: MeDICINE: Rapid prototyping of production-ready network services in multi-PoP environments. 2016 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), 2016.

[PPS16] Patel, Keyur K; Patel, Sunil M; Scholar, P: Internet of things-IOT: definition, characteristics, architecture, enabling technologies, application & future challenges. International journal of engineering science and computing, 2016.

[SJL18] Son, Yunsik; Jeong, Junho; Lee, YangSun: An Adaptive Offloading Method for an IoT-Cloud Converged Virtual Machine System Using a Hybrid Deep Neural Network. Sustainability, 2018.

[Ze20a] Zeuch, Steffen; Chaudhary, Ankit; Monte, Bonaventura Del; Gavriilidis, Haralampos; Giouroukis, Dimitrios; Grulich, Philipp M.; Bress, Sebastian; Traub, Jonas; Markl, Volker: The NebulaStream Platform: Data and Application Management for the Internet of Things. CIDR, 2020.

[Ze20b] Zeuch, Steffen; Tzirita Zacharatou, Eleni; Zhang, Shuhao; Chatziliadis, Xenofon; Chaudhary, Ankit; Del Monte, Bonaventura; Giouroukis, Dimitrios; Grulich, Philipp M; Ziehn, Ariane; Mark, Volker: NebulaStream: Complex analytics beyond the cloud. The International Workshop on Very Large Internet of Things (VLIoT), 2020.