AI-based approaches for infrastructure management for Cloud and Edge

Martin Szewieczek

Abstract. Due to the increasing complexity in today's software systems the approach of Artificial intelligence for IT Operations (AIOps) is getting more and more utilised. AIOps is adding artificial intelligence and machine learning to traditional IT operations. Executing IT operations (half)-automatically is making them more reliable and efficient. AIOps can be applied on different operation levels, e.g. automated anomaly detection, root cause analysis, remediation and optimization. Broadly speaking, the workflow of AIOps can be divided into three steps. Anomaly detection: Abnormal behaviour is getting detected with the help of AI models. Root cause analysis: The cause of the abnormal behaviour is being identified. Remediation engine: Solutions to the problem are getting automatically executed or proposed for manually execution. Most published solutions can be categorized as failure management or resource provisioning tools. This technical briefing is giving an introduction to AIOps in distributed software environments. Further, three different AIOps solutions are explained in detail: An AIOps platform specialised for highly distributed environments, called ZeroOps4e. A solution for orchestrating a large scale cloud computing platform. And BigDataStack, an orchestration service solution for applications in context with big data.

Keywords: AIOps · DevOps · Machine learning · Distributed systems · Cloud computing · Edge computing.

1 Introduction

Due to today's increasing complexity of software applications, the usage of DevOps is prevalent in the industry. It is enabling continuous delivery of applications and services to the end users. Since DevOps was established, the way software is written has changed. With the rise of cloud computing and microservices, software systems are getting even more complex due to the large existence of components and their distribution. Often, modern software systems are consisting of multiple IoT devices, adding even more intricacy. For example, smart cities can contain thousands of devices, all communicating together directly or indirectly. Nowadays IoT devices are not only sensors to gather information from and sending them somewhere else, they are often able to run computations locally. Consequently, not everything must be computed in the cloud anymore. The whole trend can be described as edge and fog computing. This development, it is bringing some new difficulties. IoT devices are still prone to errors and often are not running the newest software. This is providing us with a situation where a highly distributed infrastructure with a lot of divergent devices has to be managed and organised. It is important that devices are up and running constantly. Because software systems including these devices are often building the base for critical environments, affecting humans directly. Examples are medical devices and smart cities. Handling this new situation with DevOps approaches is not leading to the desired outcome. Already a vast amount of static algorithms were proposed to automate the handling of this problem, because of the different modalities and large amounts of data it did not work as intended. It showed that there is a need for intelligent learning systems, adaptable in their behaviour to new observations and situations. This is exactly what Artificial intelligence for IT Operations (AIOps) is capable of.

AIOps is using artificial intelligence, machine learning, natural language processing, and other related approaches to support traditional IT operations. Helping engineers and operators to be more efficient and reduce errors is the main goal. Manually done tasks like detecting and fixing erroneous configurations are done at least half-automatically. Doing this, reaction time can be reduced, thus making IT operations more reliable and efficient. It is especially effective in large IT infrastructures. AIOps is usable in various operation levels, e.g. in automated anomaly detection, root cause analysis, remediation, and optimization.

Notaro et al. [2] showed that AIOps is getting a lot of attention recently. They summarized the number of published papers in the field of AIOps until 2021. Figure 1 is illustrating an overview by year and categories. Publications about AIOps increased in the last years, focusing mostly on failure detection and resource provisioning, followed by root cause analysis and online failure prediction.

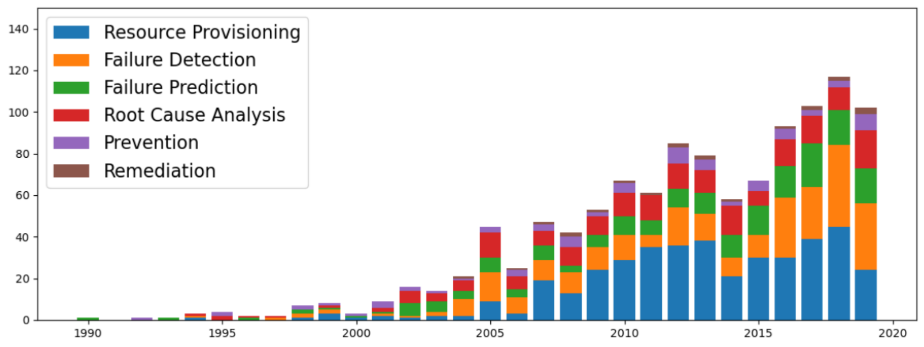

Fig. 1. Published papers in AIOps by year and categories. [2]

2 AIOps outline

Today it is not agreed-upon a common definition. Current research papers are similarly defining AIOps but having slight differences in naming conventions and describing specific use cases. In this section, three different approaches categorizing AIOps are described.

Gulenko et al. [9] are describing their AIOps framework through the following steps:

Anomaly detection: Each component in the software system is being monitored. Components can be highly different, thus individual models are used. An algorithm inside the model is detecting abnormal behaviour and sending information gained to the next step root cause analysis.

Root cause analysis: This step is investigating the actual source of the problem. It requires knowledge about the affected components and their dependencies.

Remediation engine (decision engine): As input, the engine is using data generated during root cause analysis. The data is getting aggregated and all possible solutions are getting combined. Solutions are containing decisions to schedule and manage the execution of the appropriated actions. Before the engine is knowing how to solve problems, it must get the proper inputs from experts in this field.

Gartner Inc. [3] was introducing AIOps with: AIOps combines big data and machine learning to automate IT operations processes, including event correlation, anomaly detection, and causality determination.

Additionally, it is describing the workings of AIOps in a similar but more precise manner than Gulenko et al. [9]. In the description, AIOps is consisting of five steps:

Data Selection: Most of the data provided by today's IT environments is redundant. A vast amount of heterogeneous data is coming from many different sources, e.g. networks, applications, or the infrastructure itself. This step is removing noise and selecting relevant data. One advantage is the reduction of the number of alerts operation teams must deal with, letting them focus the important tasks. As a consequence, it is speeding up the detection and further the resolution of important problems, e.g. outages that are hurting sales.

Pattern Discovery: In this step, relationships between selected data elements are found and grouped together for further analysis. Various grouping criteria can be applied, e.g. text, time, and topology.

Inference: Is the same as root cause analysis mentioned in Gulenko et al. [1]. It is identifying the root cause of the problem. Further analysis is discovering data patterns and inferring data items that signify causes and events.

Collaboration: Notifying responsible teams and operators is done here, and cooperation between them is alleviated. This step is also preserving important data which can be used on future occasions. Everyone involved in the process is getting their relevant data to conduct further actions.

Automation: Response and remediation are done in an automated way as much as possible. Quick and easy fixes are done by the AI. Human operators can then spend their time on more important and not yet automatically solvable tasks. This is bringing us a step closer to a ticketless and self-healing environment.

Becker et al. [1] and Gartner Inc. [3] are mostly describing AIOps in connection with failure management, but there are other use cases. Notaro et al. [2] are additionally describing a taxonomy regarding resource provisioning, meaning the study of allocation of energetic, computational, storage and time resources for the optimal delivery of IT services. It is including resource consolidation, scheduling, power management, service composition, and workload estimation. An overview is shown in figure 2 on the right-hand side. On the left-hand side, the different steps regarding failure management are illustrated. A detailed description of a resource provisioning example can be seen in section 3.3. In another paper Theodorou et al. [11] are proposing an intelligent Cloud-to-Edge Data Fabric, which is automating management and orchestration operations using AIOps. Their concept management workflow is including following units. Quality-driven scheduling: It is re-allocating and re-scheduling tasks on computing nodes. Flexible Data/ML Model deployment: It is efficiently utilizing resources and is maintaining analytic quality through moving data models closer to the cloud or closer to the edge. Doing this, it is affecting the location where data model training is taking place and thus influencing the amount of training data transferred across different levels. Elasticity of Data Fibers: Detecting and predicting of the utilization of resources is done here. Further adjusting actions are taking place. Monitoring, analyzing, planning and executing are the four phases of their AIOps approach. In the monitoring phase infrastructure, platform and application level data is being gathered. Preprocessing, combining and applying AI/ML techniques is done in the analyzing phase. This can lead to further actions like reducing the output quality of a deployed workflow or adding additional edge nodes. These possible actions are getting compared to each other in the planning phase. The decision is either taken by a human operator or fully automated with the help of a priority scheme. The selected actions are being executed in the execution phase. An AIOps Prometheus Framework for system analysis and orchestration is being introduced by Di Stefano et al. [12]. The framework is consisting of the following modules. A connector, extending the official Prometheus API client library. It is communicating with the Prometheus APIs. An aggregator is transforming the data, collected from a Prometheus instance through the connector, into Panda DataFrames. A helper module is preparing the DataFrames to get further analyzed, for example with ARIMA-flavoured models (Autoregressive integrated moving average). These results can be published over the network through the publisher module. An additional module can be used for debugging and deployment.

Dang et al. [8] noted that a service powered by AIOps will be having timely awareness of changes regarding various aspects, e.g. quality degradation, cost increase and workload bump. It may also predict its future status based on the learning information like historical behaviours and workload patterns. The self-awareness of a service will further lower the need of human intervention.

For increasing customer satisfaction, a service with built-in intelligence could automatically suggest or even execute performance tuning configurations.

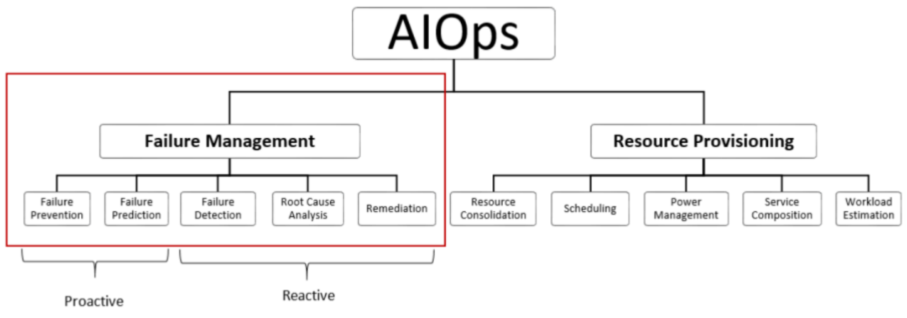

Fig. 2. Taxonomy of AIOps as observed in the identified contributions. [2]

In their paper, Notaro et al. [2] are also giving a good overview of how AIOps is used today. Most of the AIOps tools are used for failure management (62.1%), this can be even further dissected in online failure prediction (26.4%), failure detection (33.7%), root cause analysis (26.7%), and others. The other part is consisting mainly of resource consolidation, scheduling, and workload prediction. An overview can be seen in figure 3.

An often-asked question is, if introducing AI is replacing human operators, this is not the case. Maybe an operators working activities will change but it should be for the good. AI and humans are about to work together, enabling humans to focus on more meaningful tasks. Furthermore, AIOps is not replacing existing DevOps tools like monitoring, log management, or orchestration, it is integrating information from all these areas and providing help. All these tools will still be very useful and even used more efficiently. As they are used now it is hard to access the right information at the right time, this should change.

In Gartner Inc. article [3] current usage scenarios for AIOps are being listed. Some of them are:

Enterprises with large, complex environments: Companies with large IT environments, using multiple technologies are many times facing complexity and scaling difficulties. If the business model is depending heavily on IT, improving it with the help of AIOps can make a massive difference to the success of the company.

Cloud-native SMEs: AIOps is being widely used in small and medium-size enterprises (SMEs), especially those running in the cloud with the need for continuous and quick software releases. AIOps can help them preventing glitches, malfunctions, and outages.

Organizations with hybrid cloud and on-prem environments: For organisations having the whole IT in the cloud is often not preferable. Using cloud services and having part of the infrastructure on-premise is a vast adopted use case. AIOps can help to maintain control over the different environments and is providing service assurance.

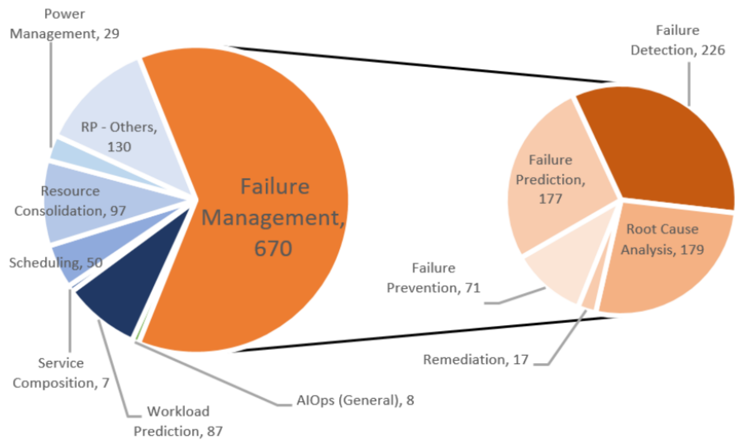

Fig. 3. Distribution of AIOps papers in macro-areas and categories. [2]

3 AIOps in practise

3.1 ZerOps4E

ZerOps4E from Becker et al. [1] is an AIOps platform specialising in highly distributed environments. By combining monitoring and AI-based methods ZerOps4E is learning the normal behaviour of edge devices and thus detecting anomalies. This is done by supporting heterogeneous CPU architectures, applying monitoring and AI models on the edge devices, and distributing analysis steps across the whole infrastructure to prevent the overloading of lightweight edge devices. To make all this work the assumption is made that the edge devices are capable of running at least docker containers, ensuring the ZerOps4E platform can run directly on the edge.

Processing of data streams as close as possible to their source is done with the help of Bitflow [4], a lightweight stream processing framework. Kubernetes is being used as the underlying infrastructure. The framework is consisting of a data collection and a data analysis section. The required AI models are stored in a designated model repository.

Data Collection: Bitflow's [4] bitflow-collector is being used for collecting the data. It is suitable for edge devices due to its promising low resource consumption for monitoring. The data is getting sent as binary data stream to the data analysis section.

Data Analysis: This self-healing analysis pipeline is consisting of multiple algorithms in different components. The components individual results are being aggregated in a higher-level (global) decision engine, where the appropriate actions are being planned and getting executed. Looking at the data analysis section in more detail it can be split into three parts. **Decentralized analysis:** The first task after getting data from the data collection section is processing and filtering it. This is being applied near or directly on the computing device of the data source. Except the case of anomaly detection, as it may consume too many resources on low-powered edge devices. As a rule, analysis should be placed as close to the data source as possible, but if needed can be moved to other components nearby or to the cloud. **Centralized analysis:** As mentioned above, due to the low computational power of edge devices some analysis has to be done centralized on the cloud. A further example is the decision engine, which is orchestrating the remediation actions. This has to be done globally because knowledge of the whole infrastructure is required. Only the large computational resources on a centralized component are making it possible to process the high amount of global information. **Partly-centralized analysis:** Some use cases are suitable for a mixture of the two approaches described above. An example is root cause analysis, which is being applied using a network slice or server rack instead of doing it on a global level.

Communication between the analysis components is done with the help of Bitflow for high-throughput data streams and RabbitMQ for transferring events. The algorithms used in the analysis phase are following the principle of Identity Function and Threshold Model. Through this, the anomaly detection model can be automatically adjusted to the evolving data stream. Long-short-term memory (LSTM), BIRCH, and autoregressive integrated moving average (ARIMA) are being used as reconstruction functions. As dynamic threshold model, exponential moving average is getting applied. As anomalies are propagating across different components, detecting them as early as possible is done by applying a time-based root cause analysis. This technique is not suitable for complex systems. For more advanced approaches they are suggesting the survey of Solé et al. [10]. The decision engine is recommending appropriate remediation action by applying density grid pattern matching.

Operator Component: A custom extension of bitflow-k8s-operator [5], a Kubernetes operator for distributed stream processing with Bitflow, is orchestrating the distributed deployment of the analysis pipeline. Similar to Kubernetes' concept of custom resource definitions (CRDs), it is introducing data sources and data analysis steps as two custom objects. Manipulating these two resources is done the same way as regular Kubernetes objects, e.g. Pods or Services. A controller is watching for updates on the CRDs and continuously is ensuring the desired state. Data sources are representing a data source such as a monitored node, virtual machine, or pod in the system. An analysis step is a form of Kubernetes Pod template and is containing all information to start a data analysis container. The first parameter is a list of ingest selectors describing the data sources that should be consumed. These ingest selectors are matched against labels of existing data source objects and count as appropriate if the matching is successful.

Model Repository: The model repository is storing all AI models used in the analysis steps. This is enhancing the learning time of analysis algorithms. Models are adapting to seasonal changes while being updated continuously. The repository is based on a Redis in-memory database.

Performance of Anomaly Detection Algorithms in ZerOps4E

As edge devices are typically restricted in terms of processing power, memory, and energy consumption, performance-related research is focusing mainly on the overhead of resource usage. For example, metric collection and deployment of multiple anomaly detection models are two important indications.

They conducted experiments where the resource consumption of edge devices and cloud instances are being compared. An Intel Xeon CPU E3-1230 V2 3.30GHz with 16GB memory cloud instance was compared to a Raspberry Pi 3B Cortex-A53 1.4GHz with 1GB memory and a Raspberry Pi 4B Cortex-A72 1.5GHz with 2GB memory representing edge devices, all using 10/100/1000 MBit/s. The ARIMA, BIRCH, and LSTM-based anomaly detection algorithms were implemented in Java and integrated into the Bitflow framework. Docker was being used as an execution platform. For each experiment, a data set of 10,000 samples, with 28 monitoring metrics was read from a file and the respective algorithms were then applied. During execution, the average processing time per sample, and their standard deviation was being measured. The cloud instance started with 8 virtual CPUs and the Raspberry Pis with 4. The allocated virtual CPUs were getting decreased by 0.1 for each following run until reaching the minimum assignable resources at 0.1. A general observation was the exponential increase in processing time when limiting the CPU resources to zero. Comparing the different approaches, the cloud instance depicted the shortest computation times. Nevertheless, the algorithms BIRCH and ARIMA were also computationally efficient on both edge devices.

To sum it up, applying anomaly detection based on the algorithms BIRCH or ARIMA are both feasible on edge devices while running in the cloud should be considered for LSTM due to higher qualitative results. In the future, the combination of quick responses at the edge could be aggregated with higher qualitative results from cloud services.

3.2 AIOps solution for SystemX

Li et al. [6] are using an AIOps solution for orchestrating an ultra-large scale cloud computing platform, named SystemX. SystemX is running inside Alibaba's data centers and is being deployed over tens of thousands of nodes. Using AIOps is especially beneficial due to the high complexity of the system. Each node has multiple configurations (e.g. CPU, memory) and is installing a variety of software applications (e.g. databases, ML platforms). The AIOps solution is predicting potential node failures, leaving DevOps engineers enough time to react. DevOps engineers can perform preventative actions, like live migration, and thus are minimizing the impact of node failures on the production system. The engineers are first performing stress testing on the affected nodes to validate if the node is indeed failing. Then, migration to a healthy node is done. During migration, the complete running status of the virtual machine on the failing node is saved and reloaded on the healthy node. Through that, the system can provide identical services without impacting the user experience. At the time this paper was written, no self-healing strategies were applied, but there is an aspiration to do it in the future.

An AIOps solution pipeline leveraging readily-available alerts, is providing actionable suggestions for the DevOps engineers. This type of communication is used because such alerts are already commonly known by engineers. The AIOps solution is generating these alerts by processing terabytes of monitoring data using derived rules of domain experts.

Overview of approach

As shown in figure 4, the AIOps pipeline is consisting of three parts. Processing monitoring data and producing useful features for the ML models is done in the feature engineering of the training data part. Next, ML models are getting trained based on the created features. The mode evaluation part is examining the effectiveness of the trained ML models in a production-like usage context.

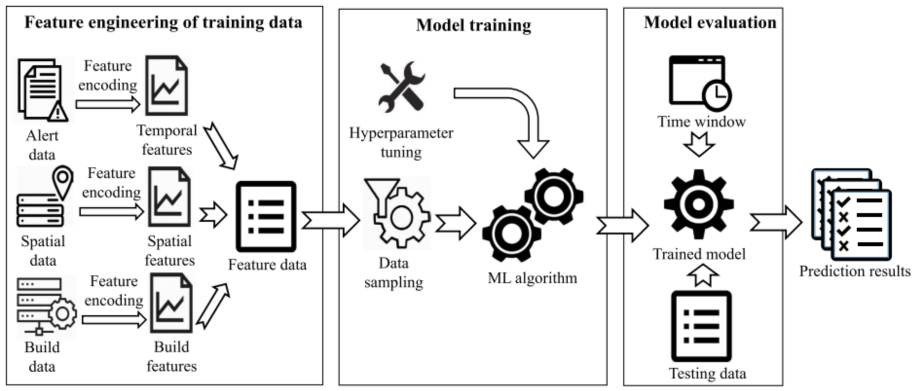

Fig. 4. Overview of AIOps pipeline. [6]

Feature engineering of training data: Three different types of data are getting transformed into features, usable by the model training. Alert data is representing early warnings, spatial data is providing additional information about nodes (e.g. the location), and build data is showing information about the software and hardware configuration of nodes. All these types of data have been proven helpful in analyzing and predicting node failures.

Because the performance of a ML model is highly depending on it, transforming raw data into features (feature engineering) is one of the key steps in successful ML projects. Due to different ML algorithms being used, the encoding has to be done in different ways. For the ML learning algorithm random forest, raw time-series data is being converted into temporal features using back-tracking. This approach can be described through an example of encoding alert data. Alert data is represented by two dimensions of temporal features frequency and change ratio. These are highlighting the differences between normal and failed nodes. **Alert frequency:** There are significantly more alerts happening in failed nodes and these alerts are happening at different time intervals prior to node failures. Thus, the frequency is used to capture the occurrences of different types of alerts over time. For each data point of a node, the frequency is getting calculated in each of the previous hours (a.k.a. back-tracking). **Alert change ratio:** Relative changes of alert occurrences are measured by observing the change ratio of alerts. The ratio between the number of generated alerts now and adjacent hours is being measured.

Spatial data is mostly in ordinal format, this has to be normalized for usage with ML models. A min-max approach is being used, normalizing data into the same range (0, 1) with the minimal ordinal value being encoded to 0 and the maximum to 1. In between, values are getting normalized based on their distance to their minimum and maximum. This is being done by using a linear transformation $\frac{x-min}{max-min}$, with x being the ordinal variable. For example, the memory size of nodes can be 20GB, 50GB, or 100GB. These are getting normalized to 0, 0.375, and 1.

Model training: LSTM, random forest, and MING are the three different ML algorithms being used. LSTM, a deep neural network-based ML algorithm, is commonly used for predicting time-series data. As node failures are being predicted using temporal features, LSTM is a fitting use case for it. A bi-directional version of LSTM is getting used, because in such a context it is promising the best performance. For a fair comparison to the other algorithms the non temporal features (spatial and build) are getting bounded with the temporal features. Random forest is an ensemble-based classification algorithm. Compared to the black-box algorithm LSTM, understanding its decision process and studying the relative importance of features is possible. This was a big factor in choosing the random forest algorithm. It is usually achieving the best performance among other traditional classifiers. MING, the third algorithm used, is a combination from LSTM and random forest. The LSTM model is learning the temporal features and the random forest model is learning the spatial and build features. Using a Learning to Rank model, these intermediate feature results are getting combined. The output is one undefined prediction outcome. At the time the paper was written MING was the state-of-the art approach for predicting node failures.

In this stage two major challenges are data skewness and hyperparameter tuning. **Data skewness:** The amount of failed and normal nodes is highly imbalanced in the network. Even for failed nodes, the healthy period is exceeding the failing. This skewness in the data set is posing challenges in building an effective node failure model. Dealing with this problem, data re-balancing (e.g., oversampling or undersampling) approaches can be used. **Hyperparameter Tuning:** Deep neural network-based models like LSTM and MING are containing many configuration parameters, collectively called hyperparameters. It is known that the performance can vary significantly based on the choice of configuration parameters. In this AIOps solution, random search is used to tune the configuration.

Model evaluation: One of their main challenges described is evaluating ML models in a production-like setting. For that, a specific evaluation technique is being developed. The prediction performance is getting evaluated for each node. This is done with the criteria of predictions, helping DevOps engineers spotting the failure of a node. To avoid premature and late predictions, a time window for predictions is getting defined. The definition is done by DevOps engineers themselves and is 2 to 72 hours before a node will fail.

Future improvements
Currently, once per hour monitoring data from individual nodes is getting aggregated and streamed to a central repository. With possible increasing data volume, the monitoring infrastructure could be unable to handle it in the future. A more fine-grained data aggregation and streaming process could be used. Alerts are containing information ranking their relevancy. Focusing on data from important alerts can reduce the bandwidth and should not be a problem regarding the performance of finding node failures. The monitoring infrastructure is highly system and company-dependent, making it hard to transfer AIOps solutions to different systems. For example, a solution is working well in the Microsoft Azure cloud, but not on a self-hosting infrastructure.

3.3 BigDataStack

Orchestrating services in the context of Big Data is a complex optimization problem considered NP-hard. The composition of services in current cloud-edge Big Data/AI applications is usually following a pipeline pattern, in which stream and batch data is being processed by multiple components. This type of architecture is adding new challenges and requirements. For example, trade-offs between computation and performance are necessary, and errors introduced in early components are cascading through the whole pipeline, affecting end-to-end performance. Nowadays, orchestrating components cannot be seen on an individual component level, but the whole system must be considered. McCreadie et al. [7] are presenting BigDataStack, a data-driven infrastructure management system using AIOps, trying to solve the problem of orchestrating services in the context of Big Data.

The systems Quality of Service (QoS) is being modeled as a constrained optimization problem. Requirements are being modeled as constraints, e.g. processing documents with an end-to-end latency less or equal to 1 second. Service performance, such as precision or battery consumption is modeled as objective. The difference between constraints and objectives is that constraints are defining a minimum or maximum value, but the object does not. Maximizing the objective of a service while satisfying the defined constraints is the main goal. Developers are defining the service requirements along with the metrics to monitor them, as well as the parameters that can be adapted and the values they can assume. During runtime, the system is trying to find the best configuration of parameter values that is maximizing or minimizing the objective, while respecting the constraints.

Driving service orchestration is done by leveraging Reinforcement Learning (RL) in conjunction with expert knowledge. A Dynamic Orchestration (DO) component is working alongside a Triple Monitoring Engine (TME), which is monitoring and triggering the redeployment of BigDataStack applications during runtime. When a new application or service is being deployed, the DO is receiving and managing monitoring requests and informing the TME and the Quality of Service (QoS) components what metric should be used for monitoring. If any violation is occurring, the QoS is informing the DO, which is deciding on the redeployment change necessary.

Figure 5 is showing their custom RL (Reinforcement Learning) framework called Tutor4RL. The tutor is possessing external knowledge and is helping the RL agent improving its decisions. This is especially effective in the early phases when the agent is inexperienced. In each step, the tutor is taking the state of the environment as input and is outputting the recommended action to take. Using this knowledge the RL agent is able to make better decisions itself. The tutor is implemented as a series of programmable functions defined by domain experts. These functions can be of two types. **Constraint functions:** These functions are disabling certain options an agent could take. For example, the maximum cost of a project must not be exceeded. **Guide functions:** These functions are expressing domain heuristics which are guiding an agent in its decisions.

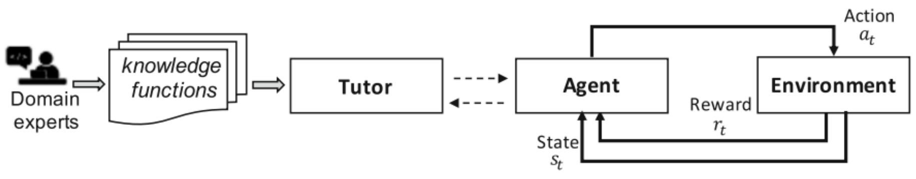

Fig. 5. Overall Working of Tutor4RL. [7]

There are two main benefits to using Tutor4RL. Firstly, the tutor is enabling faster bootstrapping to a reasonable performance level by providing domain knowledge, during training. Secondly, partial knowledge by the tutor is sufficient to improve the agents' decisions.

Example
The following example is demonstrating a connected consumer use case enabled with the help of BigDataStack. BigDataStack is helping to implement a grocery markets recommendation system, by hosting and managing all aspects of the underlying system. An overview of the underlying infrastructure can be seen in figure 6. Each box in the figure is representing a Kubernetes pod. From an external perspective, two endpoints are present. A feedback endpoint, which is receiving click and purchase events generated by users. And a recommendation endpoint, which is responding with recommended products for users. When users are visiting the website, a request is getting sent to a recommendation endpoint, which is then retrieving cached grocery recommendations from a transaction database. If users are clicking or purchasing products, events are sent to the feedback endpoint. There, they are getting reformatted and sent into the main recommendation update component via Kafka queues. The recommendation update component is a continuous application that is running in parallel over multiple Apache Spark workers.

Three main performance factors are chosen: The response time for recommendations, the delay when feedback is being recorded and the delay when user recommendations will finish updating, and the total cost of the system. In a realistic setting ,volumes of recommendation requests and feedback can vary over time, leading to periodic bursts of activity. Handling this uncertainty, BigDataStack is able to take the following actions: Increasing and decreasing the amount of Feedback Collector, Kafka, and Recommender components. Increasing and decreasing table replication within the transactional database, and increasing the number of Spark workers to which the recommendation update service having access.

A classical day of using BigDataStack can look like the following description: In the early hours of the day, BigDataStack is running the application in a minimal configuration (typically one instance per pod). This is minimizing the cost while there is little traffic. Around 7 am, the workload is beginning to increase. The triple monitoring engine is reporting that response times and feedback updates are still within acceptable bounds, thus nothing is changed. Later, the Quality of Service component is reporting failure on recommendation updates of 1.1 on average. Based on resource usage, the dynamic orchestrator is determining the bottleneck inside the recommendation updater component. It is triggering an enlargement of the spark cluster, adding additional workers. Once they are ready, the recommendation updater is getting restarted leveraging both workers. The cost per hour has now increased, but a working system is more important than money. While traffic is reaching its peak around mid-day, a second failure is being reported. The average recommendation response time is too high. Because of that, the dynamic orchestrator is increasing the replication factor of the recommender component. After collecting new measurements, it is seen that the failure is still present. Now the dynamic orchestrator is instructing the database to increase its table replication. After the process has being completed, new measurements are done and it is shown that the failure is resolved. After 6 pm the workload is decreasing. In response, the RL (Reinforcement Learning) agent within the dynamic orchestrator is experimenting with decreasing the amount of replication on the recommender component. Because no failure is observed, the decreasing state can be retained.

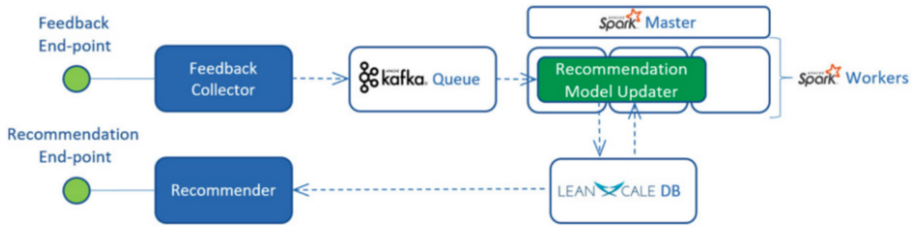

Fig. 6. Overview of the connected consumer grocery recommendation system. [7]

4 Conclusion

In this technical briefing, an introduction to AIOps was given. Information from various literature was being discussed, ranging from survey papers to individual AIOps solutions. This was revealing that AIOps is a fast changing research area, focusing nowadays on failure management and resource provisioning. Section 3 is describing three different solutions deployed in production. It is showing that IT operations are already getting improved through the help of AI. Section 3 also is showing that there is not one individual solution to all the problems AIOps should be solving. The term AIOps is being broadly used and is basically including all approaches where AI is getting used for IT operations. Using it is also adding new problems to the task of handling an IT infrastructure, but these can be solved. Using a classical DevOps approach for today's highly distributed environments, is not yielding to the desired outcome. AIOps can help here, making software systems safer and more efficient.

References

1. Becker, S., Schmidt, F., Gulenko, A., Acker, A. and Kao, O., 2020, December. Towards aiops in edge computing environments. In 2020 IEEE International Conference on Big Data (Big Data) (pp. 3470-3475). IEEE.
2. Notaro, P., Cardoso, J. and Gerndt, M., 2020, December. A systematic mapping study in AIOps. In International Conference on Service-Oriented Computing (pp. 110-123). Springer, Cham.
3. Prasad, Pankaj, and Charley Rich. "Market guide for aiops platforms." Retrieved March 12 (2018): 2020.

4. A. Gulenko, A. Acker, F. Schmidt, S. Becker, and O. Kao, "Bitflow: An in situ stream processing framework," in 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C). IEEE, 2020, pp. 182–187.
5. Kubernetes operator for distributed stream processing with Bitflow, https://github.com/bitflow-stream/bitflow-k8s-operator (retrieved as of Nov. 25, 2022)
6. Li, Y., Jiang, Z.M.J., Li, H., Hassan, A.E., He, C., Huang, R., Zeng, Z., Wang, M., Chen, P.: Predicting Node Failures in an Ultra-Large-Scale Cloud Computing Platform: An AIOps Solution. ACM Transactions on Software Engineering and Methodology 29(2) (Apr 2020). https://doi.org/10.1145/3385187
7. McCreadie, R. et al. (2022). Leveraging Data-Driven Infrastructure Management to Facilitate AIOps for Big Data Applications and Operations. In: Curry, E., Auer, S., Berre, A.J., Metzger, A., Perez, M.S., Zillner, S. (eds) Technologies and Applications for Big Data Value . Springer, Cham. https://doi.org/10.1007/978-3-030-78307-5_7
8. Dang, Y., Lin, Q. and Huang, P., 2019, May. AIOps: real-world challenges and research innovations. In 2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 4-5). IEEE.
9. A. Gulenko, M. Wallschläger, F. Schmidt, O. Kao, and F. Liu, "A System Architecture for Real-time Anomaly Detection in Large-scale NFV Systems," in ProcediaComputer Science, 2016.
10. M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on Models and Techniques for Root-Cause Analysis," pp. 1–18, 2017. [Online]. Available: http://arxiv.org/abs/1701.08546
11. Theodorou, V., Gerostathopoulos, I., Alshabani, I., Abelló, A., Breitgand, D. (2021, May). MEDAL: An AI-Driven Data Fabric Concept for Elastic Cloud-to-Edge Intelligence. In International Conference on Advanced Information Networking and Applications (pp. 561-571). Springer, Cham.
12. Di Stefano, A., Di Stefano, A., Morana, G., Zito, D. (2021, October). Prometheus and AIOps for the orchestration of Cloud-native applications in Ananke. In 2021 IEEE 30th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE) (pp. 27-32). IEEE.