ulm university universität
uulm

Done Yet? A Critical Introspective of the Cloud Management Toolbox

Mark Leznik, Simon Volpert, Frank Griesinger,
Daniel Seybold and Jörg Domaschka

DOI: 10.1109/ICE.2018.8436348

Done Yet? A Critical Introspective of the Cloud Management Toolbox

Mark Leznik, Simon Volpert, Frank Griesinger, Daniel Seybold, Jörg Domaschka[1, a)]
Ulm University, Germany

With the rapid rise of the cloud computing paradigm, the manual maintenance and provisioning of the technological layers behind it, both in their hardware and virtualized form, became cumbersome and error-prone. This has opened up the need for automated capacity planning strategies in heterogeneous cloud computing environments. However, even with mechanisms to fully accommodate customers and fulfill service-level agreements, providers often tend to over-provision their hardware and virtual resources. A proliferation of unused capacity leads to higher energy costs, and correspondingly, the price for cloud technology services. Capacity planning algorithms rely on data collected from the utilized resources. Yet, the amount of data aggregated through the monitoring of hardware and virtual instances does not allow for a manual supervision, much less data analysis or a correlation and anomaly detection. Current data science advancements enable the assistance of efficient automation, scheduling and provisioning of cloud computing resources based on supervised and unsupervised machine learning techniques. In this work, we present the current state of the art in monitoring, storage, analysis and adaptation approaches for the data produced by cloud computing environments, to enable proactive, dynamic resource provisioning.

Keywords: cloud computing, IaaS, PaaS, data science, autonomics

I. INTRODUCTION

Few terms in technology nowadays have achieved the same penetration magnitude and omnipresence across all levels of society as cloud computing. From a business point of view, the market is currently booming. The top three enterprise-cloud providers: Microsoft, Amazon and IBM, have posted cloud revenue of over $53 billion for the fiscal 2017 year[14].

The arrival of the big-data technological wave, led to an even higher exploitation of cloud technology[10]. Not only does the sheer amount of data need to be stored, the possibility of at least a primitive form of data analysis and manipulation must be supported. Cloud orchestration is now tasked with work scheduling, data storage and provisioning.

The rapid increase of infrastructure required for cloud technologies poses the additional challenge of for providers: the upkeep and capacity planning.

Hence, there is a an incentive to optimize data centres for maximum profit while maintaining service-level-agreements (SLAs)[47] towards customers. Currently, this can either result in capacity over-provisioning, which in term, leads to higher energy consumption, unused resources and higher service costs; or overbooking[46], which leads to poor user experience.

The overall research efforts in cloud environments can be divided into *(i)*: A more efficient task scheduling using e.g., more context aware scheduling and more eco-friendly data centres by means of reduced power consumption. *(ii)* Intelligent orchestration and resource provisioning[36], which can be explained by the more heterogeneous and distributed (as far as geographical location goes) nature of cloud systems and the jobs with which they are tasked.

a)Direct all correspondece to mark.leznik@uni-ulm.de

An argument can be made, that these categories cannot be easily separated, since they are closely intertwined. Reducing the amount of required hardware through efficient orchestration directly leads to an implication on the power consumption. On the other hand, a smart scheduling technique is equally responsible for a greener, more eco-friendly data centre.

In this work, we highlight current technologies enabling the optimization of cloud environments in terms of capacity planning and the difficulties in this field. We go over the processing pipeline required for collecting and evaluating data in cloud environments and explain our methodology in Section II. We then describe the components in the pipeline, starting from the data acquisition, described in Section III and provide an overview of cloud-based databases allowing the storage of the collected monitoring data in Section IV. Further, current data science approaches for the analysis of the data are depicted in Section V. Possible adaptation and remediation techniques as results of data analysis are shown in Section VI. The current state-of-the-art is presented in Section VII. We finish with a brief discussion and conclusion of the topic.

II. METHODOLOGY

While the data collection, storage, analysis and adaptation is often described as a control loop, e.g MAPE[35]. Given the introduction of data science and machine learning into the process, we feel that this must be revised. A data analysis process generally involves dumping the data from any form of database, analysing it offline and creating a model. This model is then applied to data collected online during runtime. Hence, the loop is divided into two separate parts, the data collection and analysis process, referred to as the offline process. And the data collection and adaptation process, referred to as the

online process.

Both loops can be rerun at any time with new data. A push towards direct incorporation of data collected during runtime and previously calculated models is also discussed further.

In the following sections, we provide an overview of each step of the pipeline components with the current technologies available for its implementation.

III. DATA ACQUISITION

In this section we provide an overview over the current state of monitoring and the issues and challenges in that domain. This includes in particular, how and in what form the monitored data is acquired. It directly leads to the challenges of analysing and processing this data later on. The most pressing concerns in this regard is the adaptation of a theoretically applicable approach to a real world scenario.

A. Collection

We generally distinguish between two different approaches of acquiring monitoring data. The first approach being (i) Blackbox monitoring. This strategy is of comparatively low usefulness, since it does not yield great insights on the resource being monitored. In its simplest form this kind of monitoring can be a steady stream of pings testing the availability of the resource. Blackbox monitoring is especially helpful if one does not have any control over a given resource, or is not allowed to modify it in any way, but nevertheless requires to act upon any unexpected events. (ii) Whitebox monitoring on the other hand, requires the operator to install a software which either actively pushes data to a receiving endpoint, or creates an endpoint itself, waiting to be scraped. A simple example of such an approach can be achieved using HTTP status codes returned by a prepared HTTP request. He and Wang discuss the benefits of either strategy in detail [24], proclaiming a combined approach as an optimal fit.

B. Types of data monitoring

While there is no generally agreed upon definition of monitoring data, we categorized into the following types:

Traces: Traces are a series of events an application or a software component traverses. The most useful one regarding this paper is the application request trace. These describe the life cycle of a request, either externally or internally issued, to its final destination. This request can span multiple components and represents a full path through the whole application. In contrast, a component trace only highlights the travel through a single component without the context of the whole application.

Metrics: Metrics are raw data values. They usually describe state and resource utilization of an application/instance on a given infrastructure.

Logs: Logs are usually collected on a 'per component' basis, if they are not further aggregated and are relatively similar to metrics. They consist of unstructured data, with each entry describing what the application is doing at a given point in time.

C. Acquiring and processing the data

In distributed applications, measures to aggregate all the logs, metrics and traces of the target infrastructure or application must be performed. An aggregating logical component is required combining the data and preparing it for processing. The necessary functionality is currently provided by multiple tools on the market, e.g. Logstash[1] or Fluentd[2]

The data processed by the aggregator might be in a non normalized shape, however, most of the tools in this domain assist the operator in terms of normalization and derivation.

Hereby, several assumptions made in research conditions are overthrown. Firstly, a cloud environment operator cannot be the sole provider of monitoring data. This is not feasible from a privacy point of view, since it would require the provider to monitor customers on an application level. The main research focus should be placed on data provided by the bottom two layers, namely physical servers and virtual machines running on top of them.

As stated above, the usefulness of raw metrics is limited, since they do not describe the actual performance of the component. A high CPU utilization for example might have multiple reasons with not all of them being an issue or even harmful. As Beyer et al. state[6], they circumvent that, by deriving four metrics from those raw metrics: latency, traffic, errors and saturation.

D. Concerns in monitoring

Cloud computing environments usually run on hardware which is shared amongst different parties. This is not only true for IaaS but also for PaaS where even less about the infrastructure is known. There might be someone else on the physical server in a different VM, who produces heavy disk load or who is utilizing most of the network link. Cloud providers do not act upon this issue, as long as they still fulfill the SLAs they have with their customers. In contrast, they use this practice commonly to ensure a high utilization of their resources[46]. This practice is called *Overprovisioning* or *Overbooking*.

[1] https://www.elastic.co/products/logstash
[2] https://www.fluentd.org/

Said relation is even worse when there are more layers of virtualization in between.

It can be considered a rare case where access to all layers from the hardware up to the application is given. Even more so, when logs of such a scenario are then also published, as done by Google[40,52] in their dataset. Anonymized traces of internal hardware and software to provide cluster functionality to their employers is openly shared. From the perspective of a cloud provider however, metrics like VM per compute node or geo-tags are missing to adapt it to real-world scenarios, this could however be attributed to the fact that the trace was recorded in one geographical location.

IV. CLOUD DATABASE MANAGEMENT SYSTEMS

With the demand of storing and accessing immense amount of monitoring data produced by the cloud and Big Data infrastructures, the database management system (DBMS) landscape has significantly evolved over the last decade in order cope with the challenges (Section III). In particular, the data volume, velocity and variety[1] imposes challenges to DBMS such as scalability and elasticity, diversity in data management and the usage of cloud resources[1]. NoSQL DBMS have also emerged over the last decade and became a common solution for the cloud and Big Data domains[20,53].

NoSQL DBMS are typically built as distributed systems with a shared-nothing architecture. This favours their operation on commodity hardware. Based on their distributed architecture, NoSQL DBMS support *horizontal scalability* even at runtime (*elasticity*) to handle workload peaks[2]. Yet, NoSQL DBMS offer only relaxed consistency guarantees compared to relational DBMS[49]. Further, NoSQL DBMS do not offer a standardised query interface such as SQL due to their heterogeneous storage models.

NoSQL DBMS are typically classified according to their logical data model. Initially, four data models have been defined: key-value, document-oriented, column-oriented and graph based[7]. Recent advances have brought up the time-series data model as an additional data model of NoSQL DBMS[3,25].

In the following, we introduce existing DBMS categories based on their data model and provide a concise analysis towards their usability for monitoring and data analysis systems with respect to the introduced challenges in Section III. The analysis is based on the storage model, the support for analytical queries and the exploitation of cloud resources, *i.e.* enabling scalability and elasticity. For more detailed DBMS analysis with respect to large scale monitoring systems, the avid reader is referred to[3,18,25,37].

A. Relational DBMS

Relational DBMS (RDBMS) store data as tuples, forming an ordered set of attributes. In turn, a relation consists of sets of tuples while a tuple is a row, an attribute is a column and a relation forms a table. Tables are defined using a static, normalised data schema and different tables can be referenced using foreign keys. The relational data model is suitable for storing monitoring data collected in a fixed schema. Further, with SQL as established interface for generic data definition, manipulation and query language, RDBMS provide a rich support for analytical queries and aggregations. Due to the relational data model and the provided ACID consistency guarantees[17], RDBMS typically do not provide horizontal scalability or elasticity. So called NewSQL DBMS aim at providing horizontal scalability and elasticity[20], e.g. VoltDB[3]. Yet, the possible impact on the data consistency[5] and performance[41] needs to be carefully evaluated based on the application domain. RDBMS have been a common choice for persisting monitoring data for on-premise systems in the last decade[25]. Yet, with the continuously increasing data volume and the trend to (geographically) distributed applications and the increasing volume and variety of monitoring data, the usage of RDBMS for storing monitoring data decreased due to their lacking support of scalability and elasticity.

B. Key-Value DBMS

Key-value DBMS relate to hash tables of programming languages. The data records are tuples consisting of key/value pairs. While the key uniquely identifies an entry, the value is an arbitrary chunk of data. Operations are usually limited to simple create, read, update, delete (CRUD) operations of items referenced by their key. Key-value DBMS, such as Riak[4] or Redis[5], are optimised for the operation in large-scale clusters and support horizontal scalability and elasticity. With the limited query capabilities, key-value DBMS are feasible for the sole storage of large amounts of monitoring data but the limited query hinder advanced processing.

C. Document-oriented DBMS

The document-oriented DBMS build upon a are similar data model as key-value DBMS. Yet, in contrast they define a structure on the values in formats such as XML or JSON. These values are referred to as documents, but usually without fixed schema definitions. Compared to

[3] https://www.voltdb.com/
[4] http://basho.com/products/riak-kv/
[5] https://redis.io/

key-value DBMS, they allow for more complex queries, as document properties can be used for indexing and querying. Common document-oriented DBMS such as MongoDB[6] or Couchbase[7] offer query support for aggregations and analytical functions. Further, document-oriented DBMS support a distributed operation and enable horizontal scalability and elasticity even for geographically distributed cluster. Hence, these DBMS are an appropriate solution for storing large-scale monitoring data. However, the performance implications for using the aggregations and analytical functions in combination with horizontal scaling and elasticity needs to be carefully evaluated[41].

D. Column-oriented DBMS

Column-oriented DBMS store data by columns rather than by rows. This enables storing large amounts of data in bulk and for efficiently querying over very large, structured data sets. A column-oriented data model does not rely on a fixed schema. Instead, it provides nestable, map-like structures for data items which improves flexibility over fixed schemas. Yet, the query capabilities of existing DBMS such as Apache Cassandra[8] or Apache HBase[9] only partially support advanced aggregation and analytical queries. Column-oriented DBMS support horizontal scalability and elasticity for large scale and geographically distributed clusters. Based on their features, column-oriented DBMS are appropriate to store large-scale monitoring data, especially if a high write throughput is required.

E. Graph-based DBMS

Graph-based DBMS are inspired by graph theory. They use graph structures for data modeling, thus nodes and edges represent and contain data. Nodes are often used for the main data entities, while edges between nodes are used to describe relationships between entities. Querying is typically executed by traversing the graph. Due to their graph-focused data model and query capabilities, these DBMS are not applicable for storing monitoring data as the representation of the monitoring data structure (*cf.* Section III) does not fit the graph data model.

[6] https://www.mongodb.com/
[7] https://www.couchbase.com/
[8] http://cassandra.apache.org/
[9] https://hbase.apache.org/

F. Time-series DBMS

Time-series (TS) DBMSs are driven by the need for monitoring of large-scale cloud and Internet of Things (IoT) applications and infrastructures, which require horizontally scalable DBMSs with analytical query support. Therefore, time-series DBMS typically built upon existing DBMS data models, preferred key-value or column-oriented, and add a dedicated time-series data model on top. The TS data model is built upon data points which comprise a timestamp, an associated numeric value and an unstructured set of meta-data. The query capabilities of TS DBMS focus on extensive analytical and aggregation queries. As the TS DBMSs are explicitly designed for storing large-scale monitoring data, the data model fitting for the monitoring data structures and TS DBMSs provides rich query functionality for aggregation and analytical functions. TS DBMS such as InfluxDB[10], Prometheus[11] or OpenTSDB[12], are typically optimised for write performance and they support horizontal scalability and elasticity for distribution the load across large scale clusters. Yet, the support for horizontal scalability might only be available in the commercial version of the TS DBMS, *e.g.* InfluxDB. Based on their dedicated data model for monitoring data and their scalability/elasticity support, TS DBMSs seem as an appropriate solution for solution for large scale monitoring systems. In addition, TS DBMS often provide additional tools for the collection and visualization of the monitoring data. The performance of analytical queries needs to be further evaluated[41].

V. DATA ANALYSIS

This section provides an overview of current possibilities of analysing the data aggregated by the monitoring process described in Section III and persisted using the technologies described in Section IV. Simple descriptive statistics offer a quantitative description of the data and a simple way of visualization. Hereby a subdivision into univariate and multivariate analysis is made. This can be summarized as exploratory analysis of the data. Further, more advanced analysis can be performed, namely classification and clustering and dimensionality reduction. Prediction and forecasting of the data is conducted using either statistical models, or several types of machine learning approaches.

[10] https://www.influxdata.com/time-series-platform/influxdb/
[11] https://prometheus.io/
[12] http://opentsdb.net/

A. Exploratory and Descriptive Analysis

Exploratory data analysis applies several statistical characteristics to get an initial visualisation and description of the data. A single metric distribution, e.g. the CPU load, can be analysed in terms of: mean, median, mode, minimum, maximum, quantiles. This can then be visualized in a form of a histogram or a boxplot. Multivariate analysis incorporates several metrics and provides the relationship between several variables using e.g. the covariance or a linear correlation. Similarly a visualisation is available in the form of scatterplots or conditional density plots.

This only provides so-called descriptive statistics of the data, without any kind of forecasting or advanced analysis taking place.

B. Classification and Segmentation

Classification and segmentation, also known as clustering, is generally performed on the data using several unsupervised machine learning algorithms. A large amount of related work (cf. Section VII) employs the k-means clustering algorithm[32] for workload analysis and classification. K-means provides a segmentation of the data into a previously defined number of segments. Due to its easy of use, is it widespread in a large number of research fields, such as image analysis. It necessitates either previous knowledge of the amount of output classes, or an empirical evaluation of the correct amount. Quickshift[48] is an alternative clustering algorithm. It operates using medoids instead of means for segmentation, and also does not require a predefined number of output classes. Medoids are calculated by minimizing the absolute distance between the points and the selected so-called centroid, rather than minimizing the square distance. Hence, Quickshift is more robust to noise and outliers than k-means. Quickshift also allows for a later reclustering of additional data only by using the previously calculated medoid centres, which greatly reduces computation time[48]. Similar to this, Hao et al.[22] show a concept of a never-ending learning approach for time series data, negating the general assumption that all the training data is available from the start.

A dimensionality reduction of the data, in the form of a principal component analysis (PCA) can be performed on the data before applying an unsupervised clustering algorithm. Ding and He[11] also argue, that a k-means clustering can be performed using solely a PCA, since principal components are the continuous solutions to the discrete cluster membership indicators for k-means clustering.

Generally, due to the lack of labelled/tagged monitoring data, comparable to manually tagged images required for supervised learning in image analysis, unsupervised algorithms are predominantly used. However, correlations between stored metadata and monitoring data can be made, e.g. matching a timestamp of a failed task with the corresponding CPU load.

The approach for classification of monitoring data as shown by de Carvalho Pagliosa and de Mello[9] tackles the aforementioned problem. The authors not only show a solution for working with unlabelled data, but also address the issue of noise occurrence in the data with their solution.

Segmentation and dimensionality reduction can be used for workload classification and distinction, or any given task requiring a segmentation of the input data. In the next section, we discuss means of predicting future outcomes based on the data.

C. Prediction and Forecasting

Traditional models for forecasting include: exponential smoothing, regression and composite model forecasts. Linear regression models the relationship between variables by fitting a linear equation to the data. Hereby, the previously mentioned exploratory data analysis (cf. Section V A) is of help[43]. Time series linear regression[8] can be regarded as an established concept in the field of data science. Recurrent neural networks (RNN) introduce the concept of reasoning based on previous events, which as-is is not present in traditional neural networks. Long Short Term Memory (LSTM) networks are a subset of RNNs and applicable to the analysis of time series data[16]. LSTM networks have also been used to learn lengthy patterns in time series. This can be exploited for detection of anomalies and faults in time series data, as shown by Malhotra et al.[33].

A promising deep learning approach for generating workload or time-series data using Generative Adversarial Networks (GAN)[19] was shown by Esteban et al.[13]. In their work, the authors can generate time-series data useful for supervised training with only minor degradation in performance on real test data. This can be useful when either a validation on another form of data is necessary, or the amount of input data is too small.

VI. ADAPTATION

Subsequent to the analysis of the application's state, is a new deployment plan, which is implemented by executing adaptation actions.

Many rule languages for elastic cloud applications were developed in the last years, mostly with the focus on autonomous scaling[31]. Further, there have been languages that outsource the actual adaptation to external tools and define actions as pure notifications, when a particular situation occurs, e.g[45]. This need for more than just adaptation in terms of scaling can also be seen in the Scalability Rule Language (SRL)[27], which was just recently extended to support custom adaptation workflows, instead of only supporting monolithic scaling actions[28].

As elasticity is considered a main attribute of Cloud computing[23], auto-scaling of applications naturally plays a big role in cloud environments. By horizontally scaling in or out a component, a copy of a running component is created or deleted. A component recovery is a repair operation that might need take place after an outage of a provider's data centre, which is also a situation that can be sensed by the monitoring system. The update of a component is an operation that becomes more important due to the advances in DevOps, and continuous integration and deployment in the last years with shrinking roll-out cycles[29]. Due to dynamics in user behaviours, the reconfiguration of a component may become necessary. Among this, we distinguish between (i) wiring of components, (ii) restart of components, and the (iii) change of parameters. Deployment plans can comprise multiple components with a dynamic number of component instances[4]. Therefore, it is necessary that a definition of what happens when communicating components are deployed is performed. The restart of a component can be necessary, e.g. due to temporary, unforeseeable infrastructure problems. Thus, deployment plans should be parametrizable, even on component basis.

Another possible operation is migration, e.g. by moving a component from one cloud provider to another. Cloud bursting is an operation for which resources of another cloud are used, once the initial cloud is running out of resources or is starting to violate QoS metrics. Load balancing is an adaptation that can be executed as reaction of variation in the use behaviour. Also, reactions to failures can be instructed via adaptations, such as data recovery.

These operations can happen on different cloud layers. This means a scaling action can happen by the change of the hosting virtual machine (infrastructure layer) or the change of software components (service layer). Also, the adaptation can happen in a cross-layer manner. A new dimension to this can be added by considering the user view of the action by a data-driven design of the deployment of a cloud application[12]. Further to the distinction of the several fields of operation (infrastructure, platform, component, workload, etc.), an action can be distinguished by sub-classes of them. E.g. traditionally infrastructure services are the compute services (CPU, RAM, etc.), but also the networking infrastructure is a sub-class of the infrastructure field. This is mainly fostered by the uprise of software-defined networking.

VII. RELATED WORK

The research efforts related to this work and field can be divided into approaches dedicated to reducing energy consumption in data centres and work directly dealing with the autonomics of various cloud environments. In the following, we will briefly summarize several publications particularly interesting to the context of this work.

Wang et al.[51] present and algorithm to efficiently select the ideal node utilization in a heterogeneous cluster for optimal energy efficiency when processing large data. Given that computing clusters are mainly comprised of heterogeneous nodes, the correct choice of nodes for a given task can reduce energy consumption by up to 60%. This however, does not take into account real-life circumstances such as node failures.

Guyon et al.[21] show that the energy consumption can be monitored in a more granular way with the context information provided by Platform as a Service (PaaS) applications. The authors investigate the link between user configurable PaaS parameters and the resulting energy consumption. While a fluctuation between different programming languages is noticeable, a wide consumption gap can be seen between different database technologies.

Kang et al.[26] propose a workload-aware brokering system for energy efficient containers. For this purpose, the incoming requests are classified upon their resource utilization using a the k-medoid clustering algorithm[38]. Further, an energy cost model with heterogeneous of cloud environments in mind is formulated to improve power efficiency. While the authors do compare their solution to regular dockerized setups and show the necessity of their algorithm, a comparison with more conventional virtual servers was not performed.

Liu et al.[30] also take into account the heterogeneous nature of cloud environments, as well as factors like network traffic for workload classification. A workload characterization is dynamically performed using a 0-1 integer programming model. Experimental results performed on the Google cluster dataset[39] show a significant improvement in the prediction of the resources compared to established forecasting models like the Autoregressive Integrated Moving Average (ARIMA) and linear regression.

A solution for scaling cloud applications while also taking SLAs into account is shown by Tran et al.[44]. Hereby, an autoscaling solution based on time series analysis of monitoring data using a fuzzy approach, genetic algorithm and neural networks is applied. The main contributions lie in a combination of several monitoring parameters (CPU, memory etc.) during the data analysis step to forecast system usage. Additionally, scaling decisions are made based on rules derived from SLAs and SLA violation estimation.

Wang et al.[50] are motivated very similarly to this work, touching on the subject of capacity over-provisioning by providers, in this case of virtual server instances. The authors propose to forecast server load based on previous monitoring data for an appropriate, dynamically determined number of running servers. This works stands out for using monitoring data obtained by the authors for a period of more than 3 years. This allows for a meaningful user pattern workload recognition and serves as a basis for a capacity planning process to match supply and demand for resource utilization. In total, a server amount reduction of 30% is achieved. However, given that this approach is used to classify users, it might be not be directly applicable to an enterprise solution from

a privacy point of view. This also ties in with our rationale presented in Section III, where we consider the resource supervision from a privacy perspective. While in a closed solution used by a single company, such an aspect might not be of importance, the multi-tenant nature of cloud applications however, shifts this topic into the foreground.

Mann's survey details the allocation of virtual machines in cloud data centres[34], and provides an extensive overview of the state of cloud technologies, the current problems in the field and optimization algorithms.

Due to its uniqueness, the Google cluster dataset and the original data analysis performed on it[39], is very popular among the research community, spawning a large amount of publications directly based on it [13]. The very high amount of nodes in the cluster (over 12000) as well as the recorded time length (29 days) provide a large amount of data for almost any cluster related data analysis topic.

Garraghan et al.[15] present a large-scale analysis of the Google cluster dataset, presenting several insights, including the resource utilization, workload environment characteristics, as well as wasted resource utilization. The authors show a high level of workload diversity and dynamicity in the trace log, while the correlation between resource utilization and workload variability is dependent on the architecture type. This leads to a suggestion, that the server utilization is not influenced by the dynamicity of the workload itself. The resource waste in the cluster, measured between 4.5-14.2 % for the CPU an 1.2-7.6 % for the memory is mainly attributed to the task termination primarily influenced by the cloud workload environment, rather than the workload scheduler. This naturally suggests the direction of future resource in that area.

Sîrbu and Babaoglu[42] similarly base their work on the Google cluster dataset, coining the term data-driven autonomics. Using Google's BigQuery [14], they show the applicability and feasibility of a data science based approach of using predictive models acquired from a data center log. They then build a classifier to predict the possibility of a node failure in the next 24 hours window. A precision rate of up to 72% can then be achieved. A node redirection can then be performed and hence, task failure avoided. Combined with the previously described insight of task termination based on the same dataset, this would offer a tremendous improvement in resource utilization.

[13] A full list of publications as well as the dataset itself can be found at https://github.com/google/cluster-data

[14] https://cloud.google.com/bigquery

VIII. DISCUSSION

The majority of the challenges regarding the acquisition and processing of monitoring data are solved and widely adopted. While this gives the operators of infrastructure and applications a certain degree of confidence that the systems are working as expected, there is a huge gap to make this data really useful. The biggest struggle is the extraction of useful data which really quantifies and qualifies the expected operation of the infrastructure or application and the detection of overprovisioning.

Yet, the power of monitoring causes challenges for other domains, as the tremendous amount of monitoring data that can be produced shall lead to a meaningful analysis. Further, with cloud-edge infrastructures gaining traction, the additional challenge of the geo-locality of the analysis arises.

A push towards dockerized containers and PaaS solutions can also currently be seen, due to its rapid and lightweight deployment and multi-tenant characteristics. Such solutions offer a more granular monitoring approach, which naturally is beneficial in the context of data analysis based automation.

In the overall current research landscape of using machine learning in cloud based environments, the topic of workload description and classification currently stands out. This can be explained by the ease of use of unsupervised algorithms, and additionally the lack of openly accessible data. The current research in the field of machine learning coupled with technological leaps enabling the use of neural network paradigms already established years ago opens up a lot of possibilities. The prospects of processing large amount of data in a short period of time with already given frameworks and implementation is very beneficial. We see a lot of potential in the use of this technology for this use case, specifically the application of recurrent neural networks to time series.

Current adaptation plans are mostly static and predefined during design-time of the deployment plan. Dynamic is realised by concatenating different rules and allow for variables in terms of cardinality of the components of a cloud application. A machine learning system should be able to be trained how to react properly to different situations. So an adaptation plan does not necessarily need to be created along with the deployment plan, but can be created autonomously during run-time. This has to be proven in future research.

IX. CONCLUSIONS AND FUTURE WORK

In this work, we revised the currently utilized data analysis loop of cloud computing environments, and introduced the components and technologies required for performing a data science based analysis and optimization. This work not only highlights the current state-of-the-art but also lists challenges and difficulties in implementing and running an optimization pipeline for cloud

computing setups. Future work shall focus and implementing concrete autonomous solutions for this purpose.

ACKNOWLEDGMENT

The research leading to these results has received funding from the EC's Framework Programme HORIZON 2020 under grant agreement numbers 732667 (RECAP) and 731664 (MELODIC).

[1] Abadi, D., Agrawal, R., Ailamaki, A., Balazinska, M., Bernstein, P. A., Carey, M. J., Chaudhuri, S., Chaudhuri, S., Dean, J., Doan, A., et al. The beckman report on database research. *Communications of the ACM 59*, 2 (2016), 92–99.

[2] Agrawal, D., El Abbadi, A., Das, S., and Elmore, A. J. Database scalability, elasticity, and autonomy in the cloud. In *International Conference on Database Systems for Advanced Applications* (2011), Springer, pp. 2–15.

[3] Bader, A., Kopp, O., and Falkenthal, M. Survey and comparison of open source time series databases. In *BTW (Workshops)* (2017), pp. 249–268.

[4] Baur, D., Seybold, D., Griesinger, F., Tsitsipas, A., Hauser, C. B., and Domaschka, J. Cloud orchestration features: Are tools fit for purpose? In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)* (Dec 2015), pp. 95–101.

[5] Bermbach, D., and Kuhlenkamp, J. Consistency in distributed storage systems. In *Networked Systems*. Springer, 2013, pp. 175–189.

[6] Beyer, B., Jones, C., Petoff, J., and Murphy, N. R. *Site Reliability Engineering: How Google Runs Production Systems*. " O'Reilly Media, Inc.", 2016.

[7] Cattell, R. Scalable sql and nosql data stores. *Acm Sigmod Record 39*, 4 (2011), 12–27.

[8] Chen, Y., Dong, G., Han, J., Wah, B. W., and Wang, J. Multi-dimensional regression analysis of time-series data streams. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases* (2002), Elsevier, pp. 323–334.

[9] de Carvalho Pagliosa, L., and de Mello, R. F. Semi-supervised time series classification on positive and unlabeled problems using cross-recurrence quantification analysis. *Pattern Recognition* (2018).

[10] Demchenko, Y., Turkmen, F., de Laat, C., Blanchet, C., and Loomis, C. Cloud based big data infrastructure: Architectural components and automated provisioning. In *High Performance Computing & Simulation (HPCS), 2016 International Conference on* (2016), IEEE, pp. 628–636.

[11] Ding, C., and He, X. K-means clustering via principal component analysis. In *Proceedings of the twenty-first international conference on Machine learning* (2004), ACM, p. 29.

[12] Domaschka, J., Griesinger, F., Baur, D., and Rossini, A. Beyond mere application structure thoughts on the future of cloud orchestration tools. *Procedia Computer Science 68* (2015), 151 – 162. 1st International Conference on Cloud Forward: From Distributed to Complete Computing.

[13] Esteban, C., Hyland, S. L., and Rätsch, G. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633* (2017).

[14] Evans, B. Why microsoft is ruling the cloud, ibm is matching amazon, and google is $15 billion behind, Feb. 2018.

[15] Garraghan, P., Townend, P., and Xu, J. An analysis of the server characteristics and resource utilization in google cloud. In *Cloud Engineering (IC2E), 2013 IEEE International Conference on* (2013), IEEE, pp. 124–131.

[16] Gers, F. A., Eck, D., and Schmidhuber, J. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*. Springer, 2002, pp. 193–200.

[17] Gilbert, S., and Lynch, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News 33*, 2 (2002), 51–59.

[18] Goldschmidt, T., Jansen, A., Koziolek, H., Doppelhamer, J., and Breivold, H. P. Scalability and robustness of time-series databases for cloud-native monitoring of industrial processes. In *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on* (2014), IEEE, pp. 602–609.

[19] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in neural information processing systems* (2014), pp. 2672–2680.

[20] Grolinger, K., Higashino, W. A., Tiwari, A., and Capretz, M. A. Data management in cloud environments: Nosql and newsql data stores. *Journal of Cloud Computing: advances, systems and applications 2*, 1 (2013), 22.

[21] Guyon, D., Orgerie, A.-C., and Morin, C. An experimental analysis of paas users parameters on applications energy consumption. In *IC2E 2018-IEEE International Conference on Cloud Engineering* (2018), pp. 1–7.

[22] Hao, Y., Chen, Y., Zakaria, J., Hu, B., Rakthanmanon, T., and Keogh, E. Towards never-ending learning from time series streams. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), ACM, pp. 874–882.

[23] Herbst, N., Kounev, S., and Reussner, R. Elasticity in cloud computing: What it is, and what it is not, 06 2013.

[24] Huang, H., and Wang, L. P&p: A combined push-pull model for resource monitoring in cloud computing environment. In *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on* (2010), IEEE, pp. 260–267.

[25] Jensen, S. K., Pedersen, T. B., and Thomsen, C. Time series management systems: A survey. *IEEE Transactions on Knowledge and Data Engineering 29*, 11 (2017), 2581–2600.

[26] Kang, D.-K., Choi, G.-B., Kim, S.-H., Hwang, I.-S., and Youn, C.-H. Workload-aware resource management for energy efficient heterogeneous docker containers. In *Region 10 Conference (TENCON), 2016 IEEE* (2016), IEEE, pp. 2428–2431.

[27] Kritikos, K., Domaschka, J., and Rossini, A. Srl: A scalability rule language for multi-cloud environments. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science* (Dec 2014), pp. 1–9.

[28] Kritikos, K., Zeginis, C., Griesinger, F., Seybold, D., and Domaschka, J. A cross-layer bpaas adaptation framework. In *2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)* (Aug 2017), pp. 241–248.

[29] Leppänen, M., Mäkinen, S., Pagels, M., Eloranta, V. P., Itkonen, J., Mäntylä, M. V., and Männistö, T. The highways and country roads to continuous deployment. *IEEE Software 32*, 2 (Mar 2015), 64–72.

[30] Liu, C., Liu, C., Shang, Y., Chen, S., Cheng, B., and Chen, J. An adaptive prediction approach based on workload pattern discrimination in the cloud. *Journal of Network and Computer Applications 80* (2017), 35–44.

[31] Lorido-Botran, T., Miguel-Alonso, J., and Lozano, J. A. A review of auto-scaling techniques for elastic applications in cloud environments. *Journal of Grid Computing 12*, 4 (Dec 2014), 559–592.

[32] MacQueen, J., et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability* (1967), vol. 1, Oakland, CA, USA, pp. 281–297.

[33] Malhotra, P., Vig, L., Shroff, G., and Agarwal, P. Long short term memory networks for anomaly detection in time series. In *Proceedings* (2015), Presses universitaires de Louvain, p. 89.

[34] Mann, Z. Á. Allocation of virtual machines in cloud data centersa survey of problem models and optimization algorithms. *Acm Computing Surveys (CSUR) 48*, 1 (2015), 11.

[35] Maurer, M., Breskovic, I., Emeakaroha, V. C., and Brandic, I. Revealing the mape loop for the autonomic manage-

ment of cloud infrastructures. In *Computers and Communications (ISCC), 2011 IEEE Symposium on* (2011), IEEE, pp. 147–152.

[36] Östberg, P.-O., Byrne, J., Casari, P., Eardley, P., Anta, A. F., Forsman, J., Kennedy, J., Le Duc, T., Marino, M. N., Loomba, R., et al. Reliable capacity provisioning for distributed cloud/edge/fog computing applications. In *Networks and Communications (EuCNC), 2017 European Conference on* (2017), IEEE, pp. 1–6.

[37] Palpanas, T. Data series management: The next challenge. In *Data Engineering Workshops (ICDEW), 2016 IEEE 32nd International Conference on* (2016), IEEE, pp. 196–199.

[38] Park, H.-S., and Jun, C.-H. A simple and fast algorithm for k-medoids clustering. *Expert systems with applications 36*, 2 (2009), 3336–3341.

[39] Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., and Kozuch, M. A. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In *ACM Symposium on Cloud Computing (SoCC)* (San Jose, CA, USA, Oct. 2012).

[40] Reiss, C., Wilkes, J., and Hellerstein, J. L. Google cluster-usage traces: format + schema. Technical report, Google Inc., Mountain View, CA, USA, Nov. 2011. Revised 2014-11-17 for version 2.1. Posted at `https://github.com/google/cluster-data`.

[41] Seybold, D. Towards a framework for orchestrated distributed database evaluation in the cloud. In *Proceedings of the 18th Doctoral Symposium of the 18th International Middleware Conference* (2017), ACM, pp. 13–14.

[42] Sîrbu, A., and Babaoglu, O. Towards operator-less data centers through data-driven, predictive, proactive autonomics. *Cluster Computing 19*, 2 (2016), 865–878.

[43] Tabachnick, B. G., and Fidell, L. S. *Using multivariate statistics*. Allyn & Bacon/Pearson Education, 2007.

[44] Tran, D., Tran, N., Nguyen, G., and Nguyen, B. M. A proactive cloud scaling model based on fuzzy time series and sla awareness. *Procedia Computer Science 108* (2017), 365–374.

[45] Trihinas, D., Pallis, G., and Dikaiakos, M. D. Jcatascopia: Monitoring elastically adaptive applications in the cloud. In *2014 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing* (May 2014), pp. 226–235.

[46] Tsitsipas, A., Hauser, C. B., Domaschka, J., and Wesner, S. Towards usage-based dynamic overbooking in iaas clouds. In *International Conference on the Economics of Grids, Clouds, Systems, and Services* (2016), Springer, pp. 263–274.

[47] Vaquero, L. M., Rodero-Merino, L., Caceres, J., and Lindner, M. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review 39*, 1 (2008), 50–55.

[48] Vedaldi, A., and Soatto, S. Quick shift and kernel methods for mode seeking. In *European Conference on Computer Vision* (2008), Springer, pp. 705–718.

[49] Vogels, W. Eventually consistent. *Communications of the ACM 52*, 1 (2009), 40–44.

[50] Wang, L., Cao, J., and Qu, Y. A prediction based capacity planning strategy for virtual servers. In *Data Science and Data Intensive Systems (DSDIS), 2015 IEEE International Conference on* (2015), IEEE, pp. 46–52.

[51] Wang, L., Ge, W., Li, Z., Lei, Z., and Chen, S. A large data processing algorithm for energy efficiency in a heterogeneous cluster. In *ITM Web of Conferences* (2018), vol. 17, EDP Sciences, p. 03023.

[52] Wilkes, J. More Google cluster data. Google research blog, Nov. 2011. Posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`.

[53] Wu, L., Yuan, L., and You, J. Survey of large-scale data management systems for big data applications. *Journal of computer science and technology 30*, 1 (2015), 163.