IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

SPECIAL SECTION ON SUSTAINABLE INFRASTRUCTURES, PROTOCOLS, AND RESEARCH CHALLENGES FOR FOG COMPUTING

Received February 27, 2019, accepted March 26, 2019, date of publication April 11, 2019, date of current version April 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2910584

Data Capsule: Representation of Heterogeneous Data in Cloud-Edge Computing

ION-DORINEL FILIP[1], ANDREI VLAD POSTOACA[1], RADU-DUMITRU STOCHITOIU[1], DARIUS-FLORENTIN NEATU[1], CATALIN NEGRU[1], AND FLORIN POP[1,2], (Senior Member, IEEE)

[1]Computer Science and Engineering Department, University Politehnica of Bucharest, 060042 Bucharest, Romania
[2]National Institute for Research and Development in Informatics (ICI), 011455 Bucharest, Romania

Corresponding author: Florin Pop (florin.pop@cs.pub.ro)

This work was supported in part by the ROBIN under Grant PN-III-P1-1.2-PCCDI-2017-0734, in part by the Decentralized Storage System for Edge Computing–StorEdge under Grant GNaC 2018 ARUT - AU11-18-07, in part by the NETIO ForestMon under Grant 53/05.09.2016 and Grant SMIS2014 + 105976, and in part by the SPERO under Grant PN-III-P2-2.1-SOL-2016-03-0046, 3Sol/2017.

ABSTRACT Nowadays, robots (including non-humanoid ones, like self-driving cars) are part of the most promising technologies, and they rise various computing requirements. Some of those requirements came from the fact that the robots are involved in highly dynamic environments and have to execute complex decision algorithms in real-time, while other requirements ask for batch processing of big data compatible datasets. In this paper, we propose a cloud architecture for optimizing data processing using a cloud-edge infrastructure. Besides the computational architecture, we develop a mathematical model for each type of entity in our proposal and a formal description of a data capsule, which represents a generic and flexible representation for unstructured units of data in time series databases. The architecture includes multiple processing platforms. We evaluate the proposed model in edge-cloud computing platforms designed for robots that run machine learning tasks.

INDEX TERMS Edge computing, data communication, data representation, cloud computing, big data applications, data aggregation.

I. INTRODUCTION

In the recent years, the huge growth of one of the most promising technologies of the present, the Internet of Things (IoT), has contributed to the development of a new computing paradigm called Edge Computing in which data is processed at the edge of the network. In this class of applications, various smart devices collect data from the surroundings, can run computations locally and if needed can offload work to the Cloud.

Currently, centralized models for data acquisition and storage typically ensure data integrity. However, faced with the exponential increase in Big Data volume, the centralized model faces several problems. First, the single point of failure problem occurs affecting data availability. Second, a centralized server provides low performance in interaction with multiple users who send a high number of data access requests (i.e., reads, writes). Third, the processing model requires data close to the working nodes, as the transfer cost can significantly increase for complex Big Data applications [18].

A centralized solution clearly cannot scale with respect to the data size, producing slow search and retrieval mechanisms. In this context, we envision a reliable distributed infrastructure able to support the management of heterogeneous data from many distributed sources that gradually produce increasingly more information that needs to be collected, processed in real-time and delivered to user' applications. To be successful, the platform incorporates advanced real-time data processing techniques, supported by a reliable, secure and interoperable middleware.

Such an infrastructure has the purpose of creating software modules and services to manage robots in a digital society. In this model, the robots are equipped with sensors from which data is collected and analyzed. The robots are capable to collaborate and can request additional data computed on Cloud. Some systems of collecting and processing data, such as a mounted camera, need enhanced processing and storage capabilities which is a reason to use, when possible, the Cloud support.

The associate editor coordinating the review of this manuscript and approving it for publication was Mohammad Aazam.

Multiple and variable aspects of both past and present context are important for autonomous robots (e.g. Social Robots, Game Bots, Delivery Robot, etc.), so they generate large amounts of heterogeneous and usually unstructured data. Having all of those aspects combined with a dynamic environment which is observed through multiple sensors and algorithms, we talk about a gigantic amount of data coming from an undefined number of sources that should be stored and indexed for future efficient retrieval and update.

Considering autonomous robots, various processing platforms are suitable depending on the specific algorithm to be run and potentially offloaded to Cloud. Batch processing platforms (e.g. Hadoop) use processing techniques such as MapReduce in order to run computations on large data sets and thus offer high result accuracy by processing all the data. On the other hand, the stream processing platforms (e.g. Storm) focus on processing data streams in real time and thus attempt to minimize latency and provide partial results faster [25]. There are processing platforms that employ both techniques (e.g Spark) and they can usually be used in parallel in a lambda architecture which merges the results in a serving layer.

Large data sets are produced at the exponential rate by multiple and diverse sources, like sensor networks for environmental data monitoring, traffic management, smart phones, video surveillance cameras and so on [11], [21]. Moreover, the number of Edge devices is growing at high speed rate and is forecasted that by 2020 there will be generated 43 trillion gigabytes of data per year [7]. It will lead to an explosion of network traffic to Cloud datacenters. This traffic can be reduced by performing task execution and storage on Edge devices with such capabilities. By using nodes that are a hop away in the network we can reduce significantly or distribute optimally the network traffic [29].

In the case of geographically distributed applications, Cloud Computing [2] model may not always be the best choice. In order to improve the delivered service, the processing and data storage need to be performed closer to the data source. For instance, real-time applications have small response time in the order of milliseconds (25ms to 50ms) [1]. Processing in Cloud will introduce a significant latency between the device and the Cloud infrastructure. The round trip time between two locations can exceed far the response time for real-time applications (e.g. RTT between Canberra and Berkeley is around 175ms [23]). Other application examples are video streaming and on-demand gaming. Some alternatives are Edge nodes with computation and storage capabilities (e.g. set-top-box machines, routers, etc.) which are one hop away from data source and can be used to reduce network latency.

In this paper, we address the problem of heterogeneous data set representation in Edge-Cloud platforms, to counter the problem of efficient data handling [19] with the goal of high performance, high availability and cost minimization. The main contributions of this paper are as follow:

- we create a summary of problems that should be considered when designing a Data Representation format for Cloud-Edge Computing;
- we describe a formal description for such a representational format called *Data Capsule*;
- we propose a Cloud-Edge Computing architecture to handle processing offloading over heterogeneous data represented as Data Capsules;
- we design a PoC for the proposed computing chain based on an autonomous robot running different machine learning algorithms while offloading them to Cloud or Edge;
- we asses the performance of three different serialization methods for the proposed PoC.

The paper is structured as follows. We present in Section II an overview of the related work. In Section III we formally present our problem, followed by the new proposed architecture in Section IV. Section VI presents a performance analysis when serializing data capsules using different technologies. Lastly, Section VII presents the conclusions derived from our work.

II. RELATED WORK

As great as IoT expansion and capabilities of its devices sound, the amount of generated data is increasing. It is a known fact that the entire digital data on the planet is doubling its size every 2 years, and if in 2010 the size of the total data in the world was estimated at 4.4ZB, in 2020 there will be around 44ZB. A scale factor of 2 in every year is a big challenge for data storing mechanisms. Clearly, the new focus on distributed computing field is to prepare the existing technologies to face such a scale factor.

In the past, the problems of traditional clusters with physical machines regarding administration, sustainability, managing and consumption of energy on ''not totally used'' resources were solved by the migration the Cloud Services [9], [17], [20]. In the past 12 years, the Cloud was exploited as much as it was possible, therefore this ''boom'' of data appeared.

In 2014, Cisco proposed a new way of thinking about mobile devices and how can they be used in a more complex and efficient generic architecture, known as Cloud-Edge Computing [6]. This new approach proposed a hierarchy of levels with different types of devices, starting from lower levels where end devices like phones, tables, smart watches are placed and finishing with the highest level where the Cloud machines exist [12].

From the moment Cisco introduced the possibility of designing the Cloud-Edge Computing architecture, many opportunities were analyzed by researchers [8]. Most of them address the problem of the data scale factor and the proportion of useful data.

The idea behind Cloud-Edge Computing paradigm is to move data closer to the user and at the same time store only important data. With powerful enough nodes on intermediate levels it is possible to process data at each layer and discard

all the useless data [5]. The impact of the approach of the Cloud-Edge paradigm with 4 layers in the context of smart cities and huge number of sensors was measured. As sensors generate data at a very high rate, they managed to reduce the quantity of data that is stored by 20 times using the ideas of behind the Cloud-Edge [24], [26].

All the benefits that were presented are from the point of view where the minimum response time is not a crucial service agreement [16]. The end devices might require data that can be served directly by the first layer of Edge devices and if the nodes from that layer don't have the required data, they ask the following nodes on the hierarchical path. Data sent by end devices can be batched at each intermediate level of nodes, up to the Cloud nodes, alongside with a local update on current existing data. This approach can be also improved by distributing the data to end user devices, obtaining a fine grained data distribution [15], [24]. Now these nodes can route data requests to other user devices, moving the load from edge to user devices that are capable of such operations (e.g. smartphones, tables and watches).

III. DATA CAPSULE MODEL

This section formally describes the model used throughout this paper, comprising the robots, Cloud machine and Data Capsule.

We design that format to ensure a uniform, Cloud-portable way to store relevant meta-data around multiple time-series data collections. Each collection is identified by a key composed by multiple hierarchical tokens and each element of a collection includes, beside the data blob, the relevant time-stamps and user/application-defined meta-data.

That format responds to the requirements of a storage system that aims granular data and processing offloading from user-agents to a Cloud-Edge platform. Data retrieval (search) operators are of high importance for such a system and easy to define using the chosen storage format.

A. ROBOTS

We define $\mathcal{R}$ as the *robot set* modeling N heterogeneous agents with different characteristics such as sensors, processing power with where $N = |\mathcal{R}|$. A *robot* $r \in \mathcal{R}$ has the following formal description:

$$r = (id, S, C), \tag{1}$$

where:

- id is a number that uniquely identifies a robot;
- S is the set on sensors that the robot is equipped with;
- $C = (C_1, \ldots, C_N)$ is the *resource capacity vector* of the robot $r \in \mathcal{R}$, where C_i is the amount of resource i available on the robot with $C_i > 0$.

B. DATA CAPSULE

Data flow management is one of the most important features of our proposed model. In this section we describe a structured format for data representation and storage of data, called **Data Capsule**. We define this format in the way to obtain an generic representational specification for a data unit in a time series database of unstructured data.

We define a set of data capsules as following:

$$D = \{D_i | D_i = (t_i, c_i, m_i, d_i)\}, \tag{2}$$

where:

- t_i is the time-stamp for the current data capsule;
- c_i is an ordered tuple of strings (sub-contexts) that defines the context of the data capsule;
- m_i is a set of $(key, value)$ tuples of comparable key and values that defines data capsule meta-data;
- d_i is the data content that we want to store inside the Data Capsule.

We defined a Data Capsule such that it allows retrieving Data Capsules based on context match, time-stamp intervals and meta-data based selections. Let us define an operator that selects the Data Capsules in a context. We consider that operator to be defined to take account of a special sub-context $(*)$ that matches any existing sub-context on a specific hierarchy level. Let that operator be:

$$\begin{aligned} SearchContext(D_{set}, c_{match}) = = \{D_i \in D_{set} \mid \forall s_j \in c_i \\ \text{and } \forall s_k \in c_{match} \text{ st. } (j = k) \\ \text{and } (s_j = s_k \text{ or } s_k = *)\}, \end{aligned} \tag{3}$$

where:

- D_{set} is the set of Data Capsules that we are searching on;
- c_{match} is a context that should be matched and may contain some wild-card $(*)$ sub-contexts;
- c_i is the context of D_i;
- $*$ is a special sub-context that matches any sub-context.

Let us define an operator that extracts certain Data Capsules by meta-data:

$$\begin{aligned} SearchMeta(D_{set}, m_{match}) = \\ = \{D_i \in D_{set} \mid \forall (key_j, value_j) \in m_i \\ \exists (key_k, value_k) \in m_{match} \text{ st. } key_j \\ = key_k \text{ and } value_j = value_k\}, \end{aligned} \tag{4}$$

where:

- D_{set} is the set of Data Capsules that we are searching on;
- m_{match} is a a set of meta values that should be matched;
- m_i is the meta-data of D_i.

Since we aim to describe some time series of data, an operator that selects all the Data Capsules for a context in a specified time interval is needed. Let that operator be:

$$\begin{aligned} SearchInterval \\ (D_{set}, c_{match}, t_{min}, t_{max}) = \\ = \{D_i \in \text{SearchContext}(D_{set}, c_{match}) | \\ t_{min} \leq t_i < t_{max}\}, \end{aligned} \tag{5}$$

where:

- D_{set} is the set of Data Capsules that we are searching on;
- c_{match} is a context that should be matched and may contain some wildcard $(*)$ sub-contexts;

- t_{min} and t_{max} are the bounds of the time interval;
- t_i is the time-stamp for the D_i data capsule;
- SearchContext is the operator that we described in Equation 3.

IV. ARCHITECTURE

The objective of the Robin-Cloud project is to create a support platform to collect data from sensors integrated in robots and to offer processing mechanisms by combining local processing with Cloud offloading and offering a wide range of algorithms that analyze the data.

A. EDGE LAYER

The *Edge* layer consists of robots that collect data from the surrounding environment and format it to Data Capsules which are sent to access points based on context. Data collected by the robots can be one of the following type: sensor data, GIS data, multimedia, machine generated data, online social interactions and documents. In this model it is possible that robots can cooperate with one another in other to solve more demanding tasks or to offload tasks to other robots.

B. FOG LAYER

The *Fog* layer consists of access points which are responsible to receive data or tasks from the robots and to forward them to the Cloud. After the data or task has been handled in the Cloud, the access points must forward the response to the robot that sent the original request. If needed, access points can run processing locally as they can be configured with a container based orchestration solution deployment and can run simple microservices. Alternatively, the access points can forward the data to other access points, as they are connected in a peer to peer network.

C. CLOUD LAYER

The *Cloud* layer consists of multiple components: a message processing queue component which stores data and tasks received from access points, an auto-scalable microservices component which manages interaction between the message processing queue component and the upper components of storage and processing. The data received from the robots is stored in the storage component which is a distributed database and is handled by the processing platform which analyze the data using different algorithms.

The message queue processing component consists of two brokers which are responsible for managing multiple queues for different applications and data types. There is a data broker that handles data received from the robots and there is a processing broker which handles tasks that are received from robots. Both the brokers should at least support asynchronous messaging with multiple protocols, message queueing, delivery acknowledgement and flexible routing to queues.

The auto-scalable microservices component consists of a container based orchestration solution deployment for managing microservices. Microservices continuously poll the queues from the two brokers, analyze the data or tasks that are received and interact accordingly with the storage or processing stack.

The storage component consists of a NoSQL database. We choose this storage model as the data or task representation is based on key-value pairs in documents and traditionally NoSQL database management systems are designed to handle large amounts of data across many machines while providing scalability and high availability. Each entry in the database abides to the following data format:

$$DataFormat = (id, robotid, time, location, context, data) \tag{6}$$

The processing platform component consists of multiple specialized distributed processing frameworks that have implemented multiple algorithms. Classical batch processing techniques can be employed using MapReduce based distributed processing frameworks and stream processing can also be when latency becomes a constraint and real time results are needed. The approaches from the batch and stream processing techniques can be combined in a lambda architecture which can provide the results in a view. The processing platforms that employ these techniques can be coupled with a distributed filesystems and can interact with the NoSQL database.

D. CLOUD SOLUTIONS FOR PoC IMPLEMENTATION

Open-Source solutions are available for each functional responsibility in the proposed architecture. In this section we identify some of them and we bind them to the high-level components in Figure 1.

The technologies used to build the proposed architecture consist of RabbitMQ which is the broker responsible for message queuing, Kubernetes for the container based orchestration component, Cassandra for the NoSQL storage and finally for the processing stack there are multiple distributed processing frameworks deployed such as Hadoop and Spark.

RabbitMQ is an open source message broker which supports asynchronous messaging with multiple protocols and supports deployment as clusters for high availability and throughput [28].

Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications. It groups containers that make up an application into logical units for easy management and discovery called pods and each pod runs multiple microservices as containers. Microservices can have various roles which can range from interacting with the message queues, service discovery or starting jobs on the processing platform [4].

Cassandra is a distributed NoSQL database management system designed to handle large amounts of data across many machines while providing scalability and high availability. It is based on a decentralized architecture with no central point of failure and it ensures fault tolerance by having data replication to multiple nodes [13].

Apache Hadoop is an open-source framework that allows the distributed processing of large datasets over a cluster

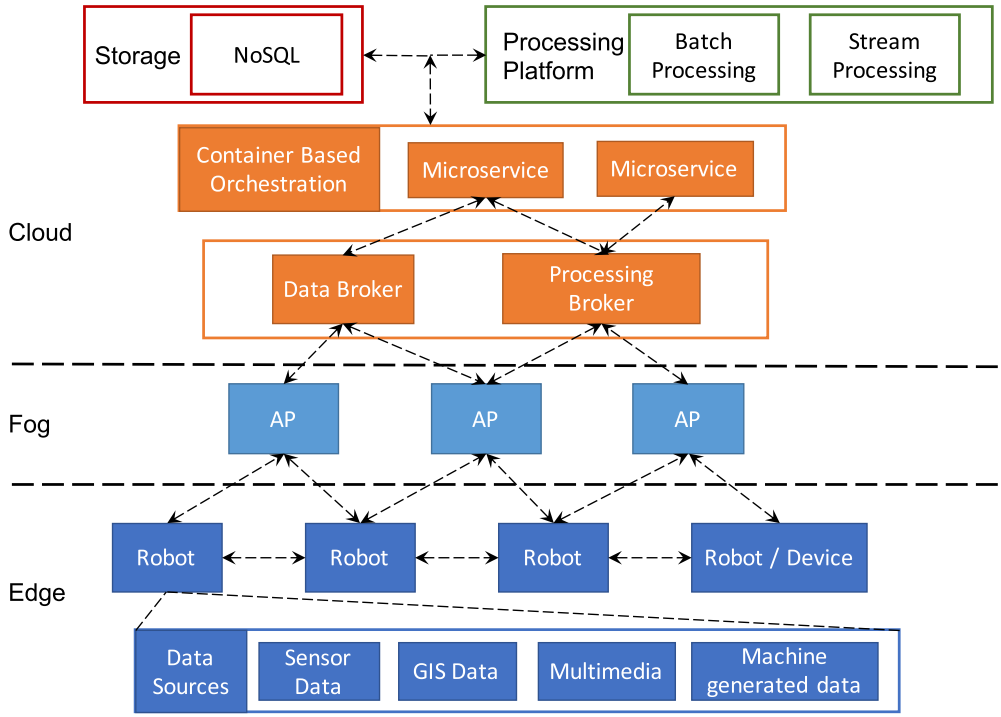

FIGURE 1. Proposed architecture.

of machines. Hadoop consists of the Hadoop Distributed File System (HDFS), YARN which is the solution for resource management and job scheduling and MapReduce which is a parallel programming model for data processing [27].

Apache Spark is a fast platform for large-scale data processing. It has an advanced DAG execution engine that supports acyclic data flow and in-memory computing based on the RDD abstraction. Spark can be used for machine learning, data analytics, SQL and streaming [30].

As an alternative in-memory computing platform to Spark, Apache Ignite can be used. Ignite is an in-memory distributed database, caching and processing platform for transactional, analytical and streaming workloads which is capable of delivering in-memory speeds at petabyte scale [10].

When there are multiple processing frameworks deployed in a single cluster, resource sharing can be handled by Apache Mesos. Mesos is a distributed system kernel that abstracts compute resources such as CPU, memory, storage from the machines in order to ensure resource sharing between multiple computing platforms. It ensures good scalability, high availability and fault-tolerance [14].

V. BUILDING AN AUTONOMOUS ROBOT ON A CLOUD-EDGE SYSTEM

In this section we describe the design and implementation of an autonomous robots which implements a few machine learning algorithms while being assisted by a Cloud-Edge processing platform.

A. DATA COLLECTION AND FILTERING

As we describe in Section IV, there are three possible data scopes to be used which divide data in three representative categories: (i) **local data**; (ii) **edge data**; and (iii) **Cloud data**.

Based on the robot's processing power and its available about resources, as well as the usefulness of the data to read, we keep the data real-time local processes (for example, data used for moving-coordinating algorithms) on the device and we offload other data to a gateway which distribute the data to Edge and Cloud nodes. That decision implies having a Data Processing System (DPS) with the predominant function of pre-processing the data gathered in real-time-based subsystems.

Data pre-processing is divided into multiple steps, as it follows:

- **Data cleaning** - removing noise and misreads or data that misfits from our input;
- **Data integration** - concatenating samples that fit together;
- **Data transformation** - keeping our data in a required threshold, such that we have a scalable system at the end;
- **Data reduction** - compressing the data in a readable, simplified way to be more efficiently stored.

B. HARDWARE, EXPERIMENT PROPOSE AND USAGE SCENARIO

As an experimental proof of concept we built a LEGO car-like robot using Raspberry Pi 3 Model B as main control unit. On top of that Raspberry Pi we set up a BrickPi board for the

purpose of making the interactions with LEGO components easier.

Raspberry Pi 3 disposes an ARMv7 Quad Core Processor which runs at a maximum speed of 1.2GHz, which well-suited for the purpose of data collection, little processing and sending Data Capsules to Fog. Those are the characteristics that makes Raspberry Pi 3 a good baseline for a mobile autonomous agent.

The network communication is ensured by the on-board WiFi network adapter which allows the board to communicate with different Access Points around.

Other important hardware characteristics for our robot consist in:

- 64GB microSD card as storage;
- 1GB of on-board RAM;
- 1 Raspberry Pi CSI camera;
- 2 microphones for sound recording and noise cancellation.

Around that commodity hardware, we build an autonomous, car-like robot that incorporates machine learning algorithms and external APIs to act as an intelligent surveillance and memory capturing device.

The main features of this robot consist in:

- being able to autonomously navigate in a room;
- recognizing a set of authorized persons through face recognition;
- identifying simple vocal commands in real-time through voice to text algorithms;
- recording surveillance footage while offloading data to cloud.

Since we only have limited processing capabilities on the autonomous agent, all the Machine Learning (ML) algorithms get offloaded to the Edge and we drive that pilot not just to prof the feasibility of our proposed data model for a given use-case, but also to asses the performance of 3 different serialization methods.

In deep analysis of each algorithm is on great interest for those aiming to implement and optimize them, but we keep theirs description to the minimum that is useful for our study.

We also describe additional steps that are necessary in data collection, enrichment and filtering since they emphasis the usefulness of being able to process unstructured data in a advanced way.

For the software level implementation of RPC (Remote Procedure Call) we demonstrate the implementation of them using gRPC.

C. VOICE RECOGNITION

High-fidelity voice recognition are available online through public Cloud, but a voice-controlled robot should not stop working if Internet connection is unavailable, so that is why we also integrated some offline alternatives.

D. NOISE REDUCTION

The LEGO motors that we used are really loud and that interferes with the voice recognition process if we do not separate the noise from the potential voice source.

We solve this problem by a very natural idea: separating the audio sample into relevant and not so relevant components. That is the main principle of a PCA (Principal Component Analysis) Machine which is a mathematical (or statistical) procedure used to transform a set of human or machine-discovered observations into a set of **principal components**.

Let's consider X a matrix representing the audio data. It has n_v lines representing the variables and n_o columns representing the observed data for each variable. We note P as the PCA matrix which is made of n_v lines and n_{ev} columns that represent the selected eigenvectors. Let's consider C the coefficient matrix representing weight of each eigenvector regarding each observation, so we can consider C_{jk} the weight of the j^{th} eigenvector regarding the k^{th} observation. X notes the columns of the matrix X stacked upon each other. Its covariance is kept in V and W has the same size as X and represents the weights of the matrix if the noise measurement would be independent [3]. Our main target is to minimize the sum in Equation 7.

$$\sum_{variable\ i, observation\ j} [X_{ij} - PC]_{i,j}{}^2 \tag{7}$$

E. TEXT TO SPEECH

In order to achieve both offline and online text-to-speech functionality we use **pyttsx3**, which is a Python TTS (Text to Speach) library. We use Python3 and we welcome the fact that this Open-Source library has a very small delay.

F. FACIAL RECOGNITION AND OBJECT CAPTIONING

The purpose is to recognize faces and objects so it can act according to its prediction in an assistive way. We use the official **TensorFlow** models to train our DNNs.

As our proof of concept, GIGEL, captures video data, we use a machine learning trained neural network to analyze every 32 frames per second.

G. IMPLEMENTATION OF DATA CAPSULE IN EDGE DEVICES

We defined the Data Capsule in Equation 6. Now we need a programming language and platform neutral mechanism for serializing heterogeneous data. With this in mind, we can consider multiple solutions and one of the most reliable is **ProtoBuffer** which is a technology created and maintained by Google which implements portable data serialization based on a very efficient byte-level representation. It can be used in C/C++, Java, Python, and many other programming languages. It is easy to integrate with gRPC, which is also maintained by Google.

```
message DataCapsule {
required Timestamp timestamp $=$ 1;
required Location location $=$ 2;
required Metadata metadata $=$ 3;
required RawData data $=$ 4;
message Timestamp {
```

```
int64~seconds $=$ 1;
int32~nanos $=$ 2;
}
message Location {
double latitude $=$ 1;
double longitude $=$ 2;
}
message Metadata {
repetead string key $=$ 1;
repetead string value $=$ 2;
}
message RawData {
optional bytes data $=$ 1;
}
}
```

The description of a data-capsule is shown in the below listing. Notice that we define a special message type for all the fields we needed inside a Data Capsule. Each of the numbers in the figure means a tag inside the message for faster lookup. We define a timestamp as a correlation between the number of seconds and the number of nanoseconds, the location as a combination of latitude and longitude, the metadata as a map. ''repeated'' meaning zero or more fields of the respective type. We define RawData as a sequence of bytes of random length. We also need to define a gRPC service that can be called from any robot as bellow.

```
message SendRequest {
required DataCapsule dataCapsule $=$ 1;
}
message SendResponse {
}
message RetrieveRequest {
optional Timestamp timestamp $=$ 1;
optional Location location $=$ 2;
}
message RetrieveResponse {
repetead DataCapsule dataCapsule $=$ 1;
}
service DataCapsuleService {
rpc Send (SendRequest)
returns (SendResponse);
rpc Retrieve (RetrieveRequest)
returns (RetrieveResponse);
}
```

We can populate with Data Capsules by calling ''Send'' which contains a SendRequest (made up of a DataCapsule) and it does not respond, but we define the response for further compatibility purposes. Also, we can retrieve one or more Data Capsules by calling ''Retrieve'' and a timestamp or/and a location. This list can easily be extended.

H. DATA COLLECTION WORKFLOW

The main workflow of a Data Capsule starts when a microservice decides to send data to Cloud. At that moment, a Data

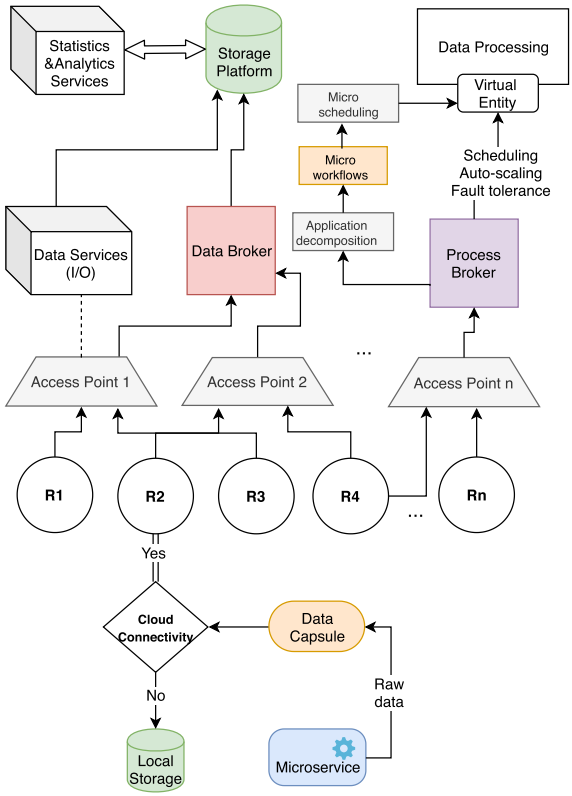

FIGURE 2. Data collection workflow.

Capsule Builder takes the raw data and transforms it into a Data Capsule under the ProtoBuffer message model. If there is no Cloud connectivity, the Data Capsule is saved locally as a FIFO set and is it checked again after some delay if the connection is up. If it is up, gRPC sends the Data Capsule to the closest Access Point which processes or schedule it to the cloud.

In Cloud most of the processing is especially made for Machine Learning algorithms. The Data Capsule Storage is periodically divided into three sets:

- Training set: used for training the weights of the Neural Networks;
- Validation set: used to tweak the parameters of the training so it has a minimum difference between the accuracy on predicted training examples and predicted validation examples;
- Test set: used for the final accuracy, an unbiased result because all of the tweaks were made against another data sets in different batch sizes or learning rates.

VI. EXPERIMENTAL RESULTS

A. EVALUATION METRICS

In order to evaluate the performance [22], of the serializers for the Data Capsule model, we compare the latency between multiple technologies for both serialization and deserialization. We have chosen three technologies: Protocol Buffers (an extensible mechanism for serializing structured data from Google), Wire (which is mainly used in the.NET

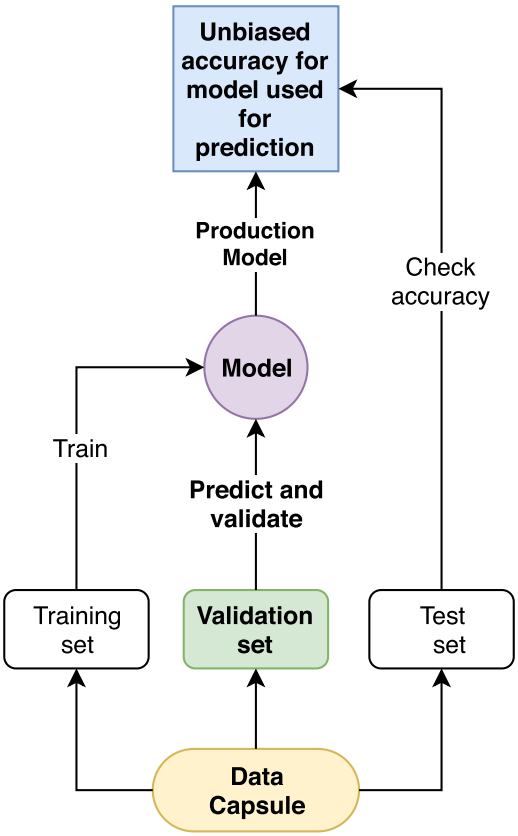

FIGURE 3. Machine learning platform on Cloud.

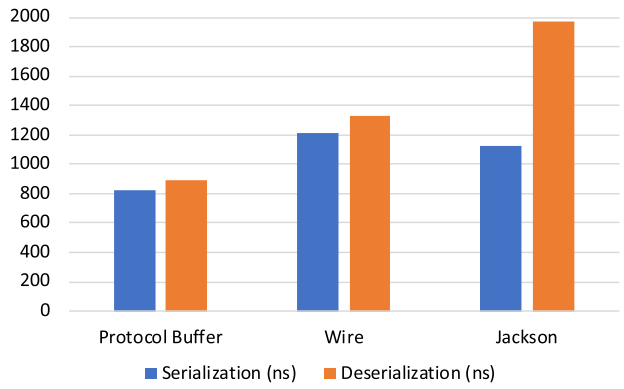

FIGURE 4. Comparison between serializing technologies.

framework) and Jackson (a framework used in most of the Java-based architectures including other frameworks as Java Spring).

B. PERFORMANCE ANALYSIS

In order to analyze the performance of the Data Capsule model for serialization and deserialization we have run multiple tests using the technologies: Protocol Buffer, Wire and Jackson. In 10000 runs, we have measured the overhead times in nanoseconds for each serialization and deserialization operation.

TABLE 1. Table with serializers comparison.

Serializer	Serialization (ns)	Deserialization (ns)
Protocol Buffer	821.23	893.81
Wire	1211.04	1327.75
Jackson	1121.13	1969.44

As it can be seen in Table 1 the latency obtained by using Protocol Buffer is the lowest when compared with Wire and Jackson for both serialization and deserialization. An overview of the difference can also be seen in the Figure 4.

VII. CONCLUSION

This paper proposes a Cloud-Edge computational architecture for autonomous robots, considering the fitness of different workloads and computational models.

As a part of our work, we defined a mathematical model that describes main entity types in out architecture and extends the common knowledge with a formal description of a *Data Capsule* witch is a time-series optimized databases for heterogeneous unstructured data.

As a part of our architecture description, we also included a study over different computing models and technologies and described the fitness between of them and different workload types/processing patterns.

We plan to continue the work in this paper, considering the following objectives:

- Extend the Fog level of the current architecture by integrating it with the proposal in NetIoT platform;
- Extend the architecture description with technology-aware specification of the middle-ware components;
- Conduct some use-case based experiments to asses the performance and fitness of the proposed architecture.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their time and expertise, constructive comments and valuable insight.

REFERENCES

[1] S. Agarwal, M. Philipose, and P. Bahl, "Vision: The case for cellular small cells for cloudlets," in *Proc. 5th Int. Workshop Mobile Cloud Comput. Services*, 2014, pp. 1–5.

[2] M. Armbrust *et al.*, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[3] S. Bailey, "Principal component analysis with noisy and/or missing data," *Publications Astronomical Soc. Pacific*, vol. 124, p. 1015, Sep. 2012.

[4] D. Bernstein, "Containers and cloud: From LXC to docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81–84, Sep. 2014.

[5] J. Cao, Q. Zhang, and W. Shi, "Challenges and opportunities in edge computing," in *Edge Computing: A Primer*. Cham, Switzerland: Springer, 2018, pp. 59–70.

[6] R.-I. Ciobanu, C. Negru, F. Pop, C. Dobre, C. X. Mavromoustakis, and G. Mastorakis, "Drop computing: Ad-hoc dynamic collaborative computing," *Future Gener. Comput. Syst.*, vol. 92, pp. 889–899, Mar. 2017.

[7] S. Curtis, "Quarter of the world will be using smartphones in 2016," *Telegraph*, vol. 11, Dec. 2014. [Online]. Available: https://www.telegraph.co.uk/archive/2014-12-11.html

[8] J. A. F. F. Dias, J. J. P. C. Rodrigues, C. X. Mavromoustakis, and F. Xia, "A cooperative watchdog system to detect misbehavior nodes in vehicular delay-tolerant networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 12, pp. 7929–7937, Dec. 2015.
[9] C. D. Dimitriou, C. X. Mavromoustakis, G. Mastorakis, and E. Pallis, "On the performance response of delay-bounded energy-aware bandwidth allocation scheme in wireless networks," in *Proc. IEEE Int. Conf. Commun. Workshops (ICC)*, Jun. 2013, pp. 631–636.
[10] T. W. Dinsmore, "In-memory analytics: Satisfying the need for speed," in *Disruptive Analytics*. Berkeley, CA, USA: Apress, 2016, pp. 97–116.
[11] E. A. El-Shafeiy and A. I. El-Desouky, "A big data framework for mining sensor data using Hadoop," *Stud. Inform. Control*, vol. 26, no. 3, pp. 365–376, 2017.
[12] P. G. Lopez *et al.*, "Edge-centric computing: Vision and challenges," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 5, pp. 37–42, 2015.
[13] J. Han, E. Haihong, G. Le, and J. Du, "Survey on NoSQL database," in *Proc. 6th Int. Conf. Pervasive Comput. Appl. (ICPCA)*, Oct. 2011, pp. 363–366.
[14] B. Hindman *et al.*, "Mesos: A platform for fine-grained resource sharing in the data center," in *Proc. NSDI*, vol. 11, 2011, p. 22.
[15] G. V. Iordache, M. S. Boboila, F. Pop, C. Stratan, and V. Cristea, "A decentralized strategy for genetic scheduling in heterogeneous environments," *Multiagent Grid Syst.*, vol. 3, no. 4, pp. 355–367, 2007.
[16] Y. Jararweh, A. Doulat, O. AlQudah, E. Ahmed, M. Al-Ayyoub, and E. Benkhelifa, "The future of mobile cloud computing: Integrating cloudlets and mobile edge computing," in *Proc. 23rd Int. Conf. Telecommun. (ICT)*, May 2016, pp. 1–5.
[17] M. A. Mohammed and N. Ţapus, "A novel approach of reducing energy consumption by utilizing enthalpy in mobile cloud computing," *Stud. Inform. Control*, vol. 26, no. 4, pp. 425–434, 2017.
[18] C. Negru and V. Cristea, "Cost models–pillars for efficient cloud computing: Position paper," *Int. J. Intell. Syst. Technol. Appl.*, vol. 12, no. 1, pp. 28–38, 2013.
[19] C. Negru, M. Mocanu, V. Cristea, S. Sotiriadis, and N. Bessis, "Analysis of power consumption in heterogeneous virtual machine environments," *Soft Comput.*, vol. 21, no. 16, pp. 4531–4542, 2017.
[20] C. Negru, F. Pop, V. Cristea, N. Bessisy, and J. Li, "Energy efficient cloud storage service: Key issues and challenges," in *Proc. 4th Int. Conf. Emerg. Intell. Data Web Technol.*, Sep. 2013, pp. 763–766.
[21] C. Negru, F. Pop, M. Mocanu, and V. Cristea, "A unified approach to data modeling and management in big data era," in *Data Science and Big Data Computing*. Springer, 2016, pp. 95–116.
[22] F. Pop, C. Dobre, and V. Cristea, "Performance analysis of grid DAG scheduling algorithms using MONARC simulation tool," in *Proc. Int. Symp. Parallel Distrib. Comput.*, Jul. 2008, pp. 131–138.
[23] M. Satyanarayanan, V. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.
[24] A. Sfrent and F. Pop, "Asymptotic scheduling for many task computing in big data platforms," *Inf. Sci.*, vol. 319, pp. 71–91, Oct. 2015.
[25] S. Sotiriadis, N. Bessis, E. G. M. Petrakis, C. Amza, C. Negru, and M. Mocanu, "Virtual machine cluster mobility in inter-cloud platforms," *Future Gener. Comput. Syst.*, vol. 74, pp. 179–189, Sep. 2017.
[26] B. Tang, Z. Chen, G. Hefferman, T. Wei, H. He, and Q. Yang, "A hierarchical distributed fog computing architecture for big data analysis in smart cities," in *Proc. ASE BigData SocialInformatics*, 2015, p. 28.
[27] V. K. Vavilapalli *et al.*, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, 2013, p. 5.
[28] A. Videla and J. J. Williams, *RabbitMQ in Action: Distributed Messaging for Everyone*, 1st ed. Shelter Island, NY, USA: Manning Publications, 2012.
[29] K. Yiannos, C. X. Mavromoustakis, J. M. Batalla, G. Mastorakis, and E. Pallis, "Resource usage prediction for optimal and balanced provision of multimedia services," in *Proc. 19th IEEE Int. Workshop Comput.-Aided Modeling Anal. Design Commun. Links Netw. (CAMAD)*, Dec. 2014, pp. 1–3.
[30] M. Zaharia *et al.*, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Networked Syst. Design Implement.*, 2012, p. 2.

ION-DORINEL FILIP received the Engineering degree in computer science from the University Politehnica of Bucharest, in 2017, where he is currently pursuing the M.Sc. degree in parallel and distributed computer systems. His diploma project was on a subject related to cloud-edge scheduling. He is an enthusiast for Computer Architectures, Computer Networks and Parallel Programming, and Parallel Programming and he is convinced by the statement assigned to an Emeritus Professor Donald Knuth—A huge gap exists between what we know is possible with today's machines and what we have so far been able to finish—and he aims to develop his contribution in both teaching and research parts of academia while finding those things that once done will improve the positive impact of computer science in our lives and in the way we apply the usage of computers in different fields. He is currently a member with the Distributed System Laboratory, Computer Science and Engineering Department, Faculty of Automatic and Computers, University Politehnica of Bucharest.

ANDREI VLAD POSTOACA is currently pursuing the master's degree in parallel and distributed systems with the University Politehnica of Bucharest, where he is a member with the Distributed System Laboratory, Computer Science and Engineering Department, Faculty of Automatic and Computers. He is also an Enthusiast of parallel and distributed systems, operating systems, and artificial intelligence. He has industry experience having done multiple internships as a Software Engineer with Microsoft and Adobe. His motto is striving to always learn and grow. His general research interests include distributed processing, cloud computing, scheduling in distributed systems, container orchestration, and virtualization.

RADU-DUMITRU STOCHITOIU is currently pursuing the master's degree in artificial intelligence with the University Politehnica of Bucharest. He is a Computer Science Engineer. He is interested in programming languages and data structures. He has industry experience having done multiple internships as a Software Engineer with Google and Bloomberg. He also participated in the Google Summer of Code, in 2016. His general research interests include machine learning, numerical methods, distributed processing, and performance evaluation. He received several awards from Bloomberg, NXP, and ACM ICPC.

DARIUS-FLORENTIN NEATU is currently pursuing the master's degree in parallel and distributed systems with the University Politehnica of Bucharest. He is a Computer Science Engineer. He is a member with the Distributed System Laboratory, Computer Science and Engineering Department, Faculty of Automatic and Computers, University Politehnica of Bucharest. He has industry experience having done multiple internships as a Software Engineer with Google and Bloomberg. His general research interests include distributed processing, cloud computing, edge computing, and performance evaluation.

CATALIN NEGRU received the Ph.D. degree in computer science from the University Politehnica of Bucharest, in 2016. The main subject of the Ph.D. dissertation was Resource Management for Cost Optimization in Cloud Storage Systems. He is a System Engineer with the Computer Science Department, Faculty of Automatic Control and Computers, University Politehnica of Bucharest, and a member with the Distributed System Laboratory. He has an experience of nine years in research projects. He has authored or coauthored more than 30 publications (books, chapters, and papers in international journals and well-established and ranked conferences). His research interests include data storage, resource management, cost optimization, edge computing, tools and applications development, and energy efficiency.

FLORIN POP (M'05–SM'18) received the Engineering degree in computer science, the M.Sc. degree in computer science, and the Ph.D. degree *(magna cum laude)* in computer science from the University Politehnica of Bucharest, in 2003, 2004, and 2008, respectively. He is currently a Professor with the Computer Science Department and is also an Active Member with the Distributed System Laboratory. He is also a Scientific Researcher with the National Institute for Research and Development in Informatics (ICI), Bucharest. He has authored or coauthored more than 150 publications (books, chapters, and papers in international journals and well-established and ranked conferences). His research interests include scheduling and resource management, multi-criteria optimization methods, grid middleware tools, applications development, prediction methods, self-organizing systems, and contextualized services in distributed systems. He is a Senior Member of ACM and euroCRIS. He received the IBM Faculty Award, in 2012, or the project "CloudWay–Improving Resource Utilization for a Smart Cloud Infrastructure," the Prize for Excellence from IBM and Oracle, in 2008 and 2009, respectively, the Best Young Researcher in Software Services Award, the FP7 SPRERS Project, Strengthening the Participation of Romania in European Research and Development in Software Services, in 2011, and two best paper awards. He involved in several international (EGEE III, SEE-GRID-SCI, ERRIC, and Data4Water) and national research projects in the distributed systems field as a Coordinator and as a member as well.