2017 IEEE 14th International Conference on Services Computing

# An Optimization Approach for Adaptive Monitoring in IoT Environments

Samir Tata, Mohamed Mohamed and Aly Megahed
IBM Research-Almaden
San Jose, CA, USA
Email: {stata,mmohamed,aly.megahed}@us.ibm.com

***Abstract*—In Internet of Things (IoT) environments, there are multiple sensors and devices monitoring different metrics and producing massive amounts of data. Monitoring of metrics can be done at different frequencies. Systems and applications that consume monitoring data typically use constrained IT resources, e.g., constrained network facilities, storage, display, processing/computing power, and energy. Given the limited quantity of resources used by these monitoring systems and applications, it is impossible to be able to collect data of all metrics in the application's context with a very high monitoring frequency. Additionally, changes in the IoT environmental context may affect the choice of metrics that should be monitored and their monitoring frequencies. To address these issues, we propose in this paper a novel approach based on optimization model to optimally determine the metrics that should be monitored and the frequencies at which these metrics are to be monitored. Our approach is an adaptive iterative approach in which the metric frequencies are re-optimized whenever an environmental event or a monitored metric value triggers the need for such re-optimization. We also present a proof-of-concept implementation of our approach that shows the efficiency of adopting it.**

***Index Terms*—Internet of Things (IoT); Monitoring; Optimization; Mathematical Programming**

## I. Introduction

The Internet of Things (IoT) refers to the network of objects, devices, machines, vehicles, buildings, and other physical systems with embedded sensing, computing, and communication capabilities, that sense and share real-time information about the physical world [1]. According to Cisco [2], IoT integration in the industry is gaining momentum and its market is continuously increasing. Moreover, based on a study performed by Gartner [3], around 25 Billion connected things will be in use by 2020. Among these connected things, massive amounts of monitoring data for different metrics are produced by sensors and devices. Examples of sensors and devices are GPS sensors, security cameras, and temperature sensors. These sensors and devices are used to monitor metrics such as weather metrics (temperature, humidity, etc.), health metrics (blood pressure, heartbeat, etc.), and transportation metrics (traffic accidents, traffic jams, etc.).

Moreover, monitoring of metrics can be done at different monitoring frequencies that can be periodic- or event-based. With periodic-based frequency, the monitoring happens at regular time periods (e.g., measure the blood pressure once per week, collect the temperature value once per hour, and monitor the memory consumption one time per second). In the literature, this type of monitoring is referred to as monitoring by interval [4]. Whereas with event-based frequency, monitoring of metrics is done when a certain pre-defined event occurs (e.g., monitoring the number of cars stuck in an accident whenever an accident occurs, measuring the average speed of vehicles when it rains, and monitoring the heart beat when the blood pressure exceeds a certain value). In the literature, this type of monitoring is referred to as monitoring on change. This means that the monitoring is done based on some properties and whenever one or more of these properties change, the data collection is triggered.

Monitored data are typically consumed in systems and/or applications that use constrained IT resources, e.g., network facilities, storage, display, processing/computing power, energy. That is why it is practically impossible to collect metric data with a very high frequency given the limited quantity of resources used by monitoring systems and applications [5], [6]. For example, the amount of data one can send through a network is constrained by the bandwidth and storage of that network. In addition, even if one can collect any amount of data and send it through a network, processing such massive data is time and resource consuming. This makes it an issue for real-time based applications evolving in resource-constraint environments, such as IoT. Moreover, event-based changes as well as environmental context ones may affect the choice of metrics to be monitored in IoT environments and/or their frequencies.

To address these issues, we present in this paper a novel approach based on a mathematical optimization model to optimally determine the metric that should be monitored

2474-2473/17 $31.00 © 2017 IEEE
DOI 10.1109/SCC.2017.55

and their monitoring frequencies. Our approach is adaptive as we update the choice of metrics to be monitored and their optimal monitoring frequencies with event-based and/or environmental-based triggers. We also develop a proof-of-concept implementation of our approach that shows its effectiveness.

The rest of this paper is organized as follows: In Section II, we give more background on monitoring as well as present some motivating examples. In Section III we review the related literature. We then present out approach in Section IV followed by its implementation in Section V. Finally, we present our conclusions and future work in Section VI.

## II. Background and Motivating Examples

In this section, we first give some background for metric monitoring, and then present some motivating examples to illustrate the problem we address in this paper.

### A. Monitoring

Monitoring consists of informing interested parties (humans or software) of the status of a specific metric (e.g., a device property, a service attribute, or a monitored object status/attribute). Monitoring is important for both service providers and service consumers. It is a key tool for controlling and managing hardware and software infrastructures. It provides key performance indicators for devices, objects, applications, and platforms. Additionally, it involves many activities such as resources planning and management, devices diagnostic, SLA management, billing, troubleshooting, and security management.

In the literature, there are mainly two models for monitoring: monitoring by polling or by subscription. We illustrate the differences between them as follows:

*a) Monitoring by Polling:* Monitoring by polling is the simpler model of monitoring; as it allows the interested parties to request the current state of the monitored metric whenever there is a need. The interested parties can generally interact with a specific interface that provides a getter of the needed property. For example, monitoring the temperature of a room by polling involves querying a specific interface that would give us the current reading of the temperature.

*b) Monitoring by Subscription:* Monitoring by subscription is based on a publish/subscribe model which is defined as a set of nodes divided into publishers and subscribers. Subscribers express their interests by subscribing for specific notifications independently of the publishers. Publishers produce notifications that are asynchronously sent to subscribers whose subscriptions match these notifications [7]. The notifications might be classified into topics. Consequently, the subscribers might subscribe to one or more topics.

Subscription allows an observing part to be notified about changes of monitored metrics using one of two modes. The first mode is the subscription on interval, which implies that the publisher (producer) broadcasts the state of its properties periodically to the subscribers (consumers). In this case, the subscriber might specify the frequency with which he needs to receive notifications. The second one is the subscription on change, which implies that the publisher has to notify the subscribers whenever there are new monitoring information related to properties change, method calls, or service time execution. In the property changed monitoring, the monitoring producer has to send notifications to all subscribers whenever a monitored property is changed. In the method call monitoring, the monitoring producer sends notifications whenever one of the service's methods is invoked. Lastly, in the execution time monitoring, the monitoring producer notifies the subscribers about the execution time whenever a service invocation occurred.

### B. Motivating Examples

In this section, we will give two examples to motivate our work.

*1) Monitoring Routes for Smart Cars:* During a navigation task, including turn-by-turn directions, a route from a starting point to a destination is selected. The route is determined according to a predefined rule ( *e.g.* minimizing the time to the destination or minimizing the distance to it). The navigation task includes monitoring metrics such as traffic flow (*e.g.* speed and accidents), speed of the physical devices, and their locations (GPS). The frequency of measurements and any alternative routes that are monitored are limited due to resource constraints of the physical device or the network in communication with the physical device.

In response to the occurrence of a traffic accident on the current route, the physical device changes a physical setting to receive traffic data corresponding to alternative routes so that the navigation task can determine if a quicker route is available. In this case, the monitoring is adapted via adding and removing metrics based on the occurrence of an environment-related event, which is the traffic accident in this context. Such adaptation might be metric value-based as well. This is the case when, for instance, the measured traffic speed happens to be below a threshold. In this case, the physical device will respond by changing a physical setting to receive traffic data corresponding to alternative routes.

*2) Monitoring in Healthcare Applications:* Suppose the blood pressure for a certain patient is measured once per week. An example of value-based adaptation is when the

value of the measurement goes above (or below) a certain predefined threshold, triggering the need to monitor this metric more or less frequently, e.g., on a daily or a monthly basis. Another example is that a value of a measurement satisfying a monitoring condition can affect the selection of metrics to be measured. This might include adding an additional metric, e.g. a measurement of the heart rate, or removing a metric, e.g. stopping the blood pressure measurement.

## III. State of the Art

Several monitoring approaches have been defined in the literature [8]. We cite among others monitoring approaches based on sampling and filtering (e.g. [9]), monitoring approaches using probing (e.g. [10], [11]), monitoring approaches based on diagnosis (e.g. [12], [13]) and monitoring based on performance (e.g. [14], [15]).

In [9], authors present AdaM, an adaptive monitoring framework for smart IoT devices. Based on metric monitoring values, AdaM dynamically adapts the monitoring frequency and the amount of data distributed over the IoT system. The idea is to increase the monitoring frequency when a metric values approaches a user-defined threshold and to decrease the monitoring frequency when a quality of service violation is unlikely to occur. Contrary to our work, this paper always report all metrics to the consumers. It never filters any metric. In addition, the choice of the frequency is based only on the metric-values. It does not consider resource availability and constraints, while we do in our paper.

In [10], authors present a passive monitoring approach using probing. Contrary to our paper, the authors do not consider resource availability and constraints when determining the metric frequencies.

In [12], the authors adopt a monitoring approach based on diagnosis. The approach consists in identifying system faults and then adapting monitoring accordingly. Our objective is to continuously optimize monitoring adaptation given the resource constraints in the system. In addition the authors of [12] do not consider resource availability and constraints when determining the metric frequencies, while we do in our work.

In [14], authors present a method that integrates, validates, describes, and categorizes metrics of software project performance. On one hand, integrating new metrics is not optimized as we do in our paper (e.g., resource capacities, utilization, and constraints used for the monitoring are not considered). On the other hand, contrary to their work, in our paper we determine and optimize metric frequencies.

In their work, G. Katsaros et al. [16] proposed an architectural approach spanning over virtualization and physical levels to collect monitoring data. They combined many existing open source solutions (e.g. Lattice [17], Ganglia [18], Nagios [19]) to get one holistic application that cover different layers. They use collectors to extract data from different layers (virtual and physical) and externalize it to the upper layer using an external data collector. Moreover, a monitoring manager serves as the orchestrator of the whole monitoring process by controlling and providing the needed interfaces to add or to consume monitoring information. The proposed system does not allow to adapt the monitoring according to the needs. One of the reasons is that the context of this work was related to cloud environments. Therefore, no constraints were considered about the available resources in the environment.

In [20], J. Brandt et al. proposed OVIS as a tool for monitoring resources enhancing high-performance computing in distributed environments. This tool can extract the application and resources state, and based on that state it can assign new resources or shut down unused ones during the application's runtime or for next usages. The data is collected from the resources using data collectors able to collect information and save it in a distributed data base. Then, a statistical analysis is necessary to take decisions to keep or to manually reconfigure the resources' assignment for the application. In this proposal, even though the system allows to monitor and adapt resources, the monitoring resources themselves (collectors) are not subject to adaptations.

In a previous work [4], we proposed a monitoring approach for cloud resources. This approach is based on OCCI and allows to put in place all the needed facilities to monitor cloud resources. It offers also the needed facilities to reconfigure the resources when needed. Even though this approach has the needed mechanisms to enable adaptive actions, the adaptation of the monitoring tools themselves as well as the proposed system was not investigated.

## IV. Methodology

We propose a novel approach that updates the choice of metrics to be monitored and their optimal monitoring frequencies with event-based and/or environmental-based triggers while taking into account resource constraints. In the following we give an overview of our approach and then detail its steps.

### A. Approach Overview

Our approach for metric adaptation and frequencies optimizations is summarized in Algorithm 1. The monitoring systems consider a set of metrics to monitor and the frequencies those metrics should be monitored at. Metrics and their frequencies are continuously adapted using Algorithm 1.

```
1: while true do
2:   On event Triggered do
3:     if the event is environment-related then
4:       Adapt metrics (add, remove, and retain metrics)
5:       Compute metric weights
6:       Optimize metric frequencies
7:     else
8:       if the event is metric value-based then
9:         if there is a need to extend monitoring to correlated metric then
10:          Adapt metrics (add, remove, and retain metrics)
11:          Compute metric weights (function of added and retained metrics, and the event)
12:        else
13:          Compute metric weights (function of retained metrics and the event)
14:        end if
15:        Optimize metric frequencies
16:      end if
17:    end if
18:  End Event
19: end while
```

**Algorithm 1:** Metric Adaptation and Frequencies Optimization

Our algorithm works as follows: If any environment-related event that may have any impact on monitored metrics is triggered, then the set of metrics to be monitored is adapted (lines 2-4), the weights of these metrics is computed (line 5), and the frequencies those metrics should be monitored at are determined (line 6).

If any metric value-based event is triggered and if the metric value induces a need to extend the monitoring to correlated metric(s), then the set of metrics to be monitored is adapted (lines 8-10) and their weights are computed (line 11). If the value of metric value-based event does not induce any needs to change the retained metrics, then only metric weight computation is needed (line 13). In either case, we then determine the frequencies those metrics should be monitored at (line 15).

We now present how metrics are adapted (Section IV-B), how their weights are computed (Section IV-C) and how their frequencies are optimized (Section IV-D).

### *B. Metric Adaptation*

In our approach, metric adaptation is enacted by an environment-related event (Algorithm 1, line 3) or value-related event that induces a need to extend the monitoring to correlated metric(s) (lines 8-10). More precisely, metric adaptation is based on correlation between metrics on one hand and correlation between metrics and environment-related events on the other hand. Correlations allow to retain, add, or remove the metrics to be monitored.

A metric may be correlated with one or more of the other metrics. This connotes that the monitoring of a metric may depend on the value of another monitored metric, i.e., the monitoring value of a metric may trigger the monitoring of another metric. For example, blood pressure and heart rhythm are correlated. If the measured blood pressure turns out to be over a certain threshold (*i.e.* a metric value-based event is triggered), a metric to monitor heart rhythm should be added. The correlation in this case is positive and induce adding a new metric(s). Correlations can also be negative and may induce removing of a monitored metric(s).

A metric may be correlated with one or more environment-related event. An environment-related event is associated with an adaptation rule that includes a predefined set of metrics that are to be monitored in response to the occurrence of the event, thus metrics are added and removed based on the predefined set of metrics. For example, if it is raining, a metric to monitor the traffic speed is added. Another example is the triggering of an environment-related event specifying that the monitored route of a smart car is closed for some car accident. In this case, the smart car needs to begin getting monitoring data for the other new routes to be used and at the same time stop getting monitoring data for the closed/jammed route. The environment-related event would be described such that it would be positively correlated to metrics of the new routes and negatively correlated to the closed/jammed route.

### C. Metric Weights Computation

Weights are assigned to the metrics that are currently monitored (retained and not removed) and any newly added metric(s) to be monitored. Computation of metric weights depends on whether a metric value-related event or an environment-related event has been the trigger.

In the case of metric value-related events that induces a need to extend the monitoring to correlated metric(s), the weight of each of the metrics to be added is computed as an aggregation function of the weights of the metric that triggered the event, the weights of other metrics already being monitored that the newly added metric is correlated with, and the correlation coefficient between them. An example of such function is the multiplications of correlation coefficients. If the value-related event doesn't induce a need to extend the monitoring to any new metrics, the newer weights of the existing metric(s) that triggered the event are computed as a non-decreasing function of the absolute difference from the monitored metric value and the threshold value (or threshold difference) that triggered the value-related event.

In the case of environment-related events, weights of the metrics to be added are computed as an aggregation function of the correlation coefficient between that metric and the environmental-related event, the weights of other metrics already being monitored that the newly added metric is correlated with, and the correlation coefficient between them.

### D. Frequencies Optimization

Let $I$ represent the set of metrics to be monitored. There are different possible frequency intervals at which we can monitor each metric (e.g., monitor each 10 ms or each 2 minutes). Let $K$ be the set of these intervals, ordered from the lowest to the highest interval value. We assume that we are given the minimum and maximum possible monitoring interval for each metric. The minimum possible value in $K$ is the minimum among these minimums, $k_{min}$ and vice-versa for $k_{max}$ or $|K|$, where $|K|$ is the cardinality of the set $k$. The difference between the values of each two successive elements of the set $K$ is equal to the minimum given interval value.

We also define set $J$ as the set of potential time stamps at which we can do any of the monitoring. $|J|$, the cardinality of set $J$ is equal to the $k_{max}$ (or $|K|$). Elements of $J$ are also ordered, and the step size between each element and the next is equal to a given/chosen minimum precision value. Computing the optimal frequency interval for each metric $i \in I$ is the main target of this problem with the objective of maximizing the monitoring frequency of the metrics with higher weights. We define $w_i$ as the given weight for metric $i \in I$. Let the binary variable $Interval_{ki}$ be 1 if metric $i \in I$ is to be monitored at frequency $k \in K$, and zero otherwise. Let the binary variable $B_{ij}$ be 1 if the monitoring of metric $i \in I$ is to start at time stamp $j \in J$, and zero otherwise.

There is a number of resources that are used for the metric monitoring. Let $R$ represent the set of these resources. Each resource $r \in R$ has a given capacity $rcap_{rj}$ at time stamp $j \in J$. Each metric $i \in I$ utilizes an amount $ut_{ir}$ of resource $r \in R$ when it is being monitored at any time stamp. Lastly, let the binary variable $X_{ij}$ be 1 if metric $i \in I$ is to be monitored at time stamp $j \in J$, and zero otherwise.

Note that the way we defined the set $J$ will ensure that the solution of the optimization model will determine the optimal monitoring intervals and time stamps for one complete cycle. This cycle should be repeated when these results are implemented.

Given the aforementioned notation, our problem can be formulated in equations 1-10 where objective function 1 maximizes the monitoring frequency of the metrics with a direct linear proportion to their weights. Constraint 2 ensures that each metric begin to be monitored only once during the planning time cycle. Constraint 3 guarantees that each metric is assigned only one interval size. Constraints 4 and 5 set the relationship between the $X_{ij}$ variables and $B_{ij}$ ones, where the former sets the binary variable corresponding to the first monitored time stamp to 1, and the latter sets all the monitorings before the first one for each metric to zero. Thus, so far, we guarantee that monitoring of each metric will begin at the specific time that the model chooses and will never begin earlier. Now, we need to make sure that the next monitorings are at the equal chosen interval for each metric. This is what constraint 6 is for. This constraint ensures that, if interval $k \in K$ is the chosen one for a certain metric, exactly one period out of each $k$ successive ones is chosen as a monitoring period for that metric. Constraint 7 is the resource capacity one. It makes sure that at each time stamp $j \in J$, the total resource utilization of each metric that is to be monitored at that time stamp, is less than or equal to the available resources at that time stamp for each resource in the set $r \in R$. Constraints [8-10] are the binary restrictions of our binary variables.

## V. Implementation

In this section, we will give a description of the implementation that we realized during a proof of concept preparation for our work. As shown in Figure 1, we implemented our adaptive monitoring system with conformance to what we presented previously. We used 4 Raspberry Pi 3 boards model V as our Edge devices connected to the system

$$\max \sum_{i \in I} \sum_{j \in J} w_i * X_{ij} \tag{1}$$

$$\text{s.t.} \sum_{j \in J} B_{ij} = 1, \quad \forall i \in I \tag{2}$$

$$\sum_{k \in K} Interval_{ki} = 1, \quad \forall i \in I \tag{3}$$

$$X_{ij} \geq B_{ij}, \quad \forall i \in I, \quad \forall j \in J \tag{4}$$

$$X_{il} \leq (1 - B_{ij}), \quad \forall l \in \{1, \ldots, j-1\}, \quad \forall i \in I \tag{5}$$

$$\sum_{z \in \{j, \ldots, j+k-1\}} X_{iz} = Interval_{ki}, \quad \forall i \in I, \quad \forall j \in J : (j+k-1) \in J, \quad \forall k \in K \tag{6}$$

$$\sum_{i \in I} ut_{ir} * X_{ij} \leq rcap_{jr}, \quad \forall j \in J, \quad \forall r \in R \tag{7}$$

$$X_{ij} \in \{0, 1\}, \quad \forall i \in I, \quad \forall j \in J \tag{8}$$

$$B_{ij} \in \{0, 1\}, \quad \forall i \in I, \quad \forall j \in J \tag{9}$$

$$Interval_{ki} \in \{0, 1\}, \quad \forall k \in K, \quad \forall i \in I \tag{10}$$

sensors. Each Raspberry Pi was flashed with Hypriot OS[1] to be able to host Docker[2] containers [21].

The different components that make our proof of concept are the system and environmental sensors responsible for collecting monitoring data. These sensors are driven by data collectors. Moreover, we have a monitoring manager responsible of configuring the data collectors of the system sensors through two local agents: the *Frequency Manager* and the *Metric Manager*. We also implemented an *Analyzer* implementing our optimization algorithms and responsible for analyzing all of the collected data and the correlation between the metrics as described in Section IV. All monitoring data is stored in a Cloudant database offered by Bluemix®.

The system sensors are driven by *Data Collectors* that dictate which metric to be collected and at which frequency. The metrics to be monitored are specified by the *Metric Manager*. This component can turn on and off the data collection by activating and deactivating a specific data collector. The frequencies of the data collection is controlled by the *Frequency Manager*, which can tune the frequency of data collection of a given metric through its *Data Collector*. These three components (the Frequency Manager, Metric Manager, and Data Collector) are implemented as Java applications exposing REST interfaces to facilitate the communication. We used Jersey JAX-RS (JSR 311) [22] for the RESTful implementation. They are deployed as Docker containers running on our Raspberry Pi boards. Since Raspberry Pi boards have an ARM processor, it is not currently possible to run regular docker containers on them. This is because the regular docker containers are built for a totally different architecture of processors. Thus, instead, we used the "hypriot/rpi-java[3]" base image to build our containers.

The *Frequency Manager* receives the ID of the metric and the new monitoring frequency. This component is also caching the previous frequencies of monitoring. When it receives a new frequency, it compares it to the old value. If that value is different, it reconfigures the associated *Data Collector*. If it is not, then it just updates its record. The *Metric Manager* offers interfaces for CRUD functions of metrics. It allows adding a new metric, listing the monitored metrics, updating an existing metric, and deleting a metric. The *Data Collector* components, they are instantiated on-demand by the *Metric Manager*. They are responsible for collecting the data from the sensors, transforming that data to the standard format that our system understands, and publishing them to Bluemix®IoT platform. Each collector has a default frequency of data collection. This frequency might be changed via an interface exposed by the collector. It is noteworthy that each collector is configured to auto-register to the Bluemix®IoT platform to enable it to send its data using MQTT [23].

Moreover, we implemented the *Analyzer* and the *Monitoring Manager* as two Java web applications deployed on the Bluemix®Platform as a Service. The *Analyzer* con-

[1]https://blog.hypriot.com/

[2]https://www.docker.com/

[3]https://hub.docker.com/r/hypriot/rpi-java/

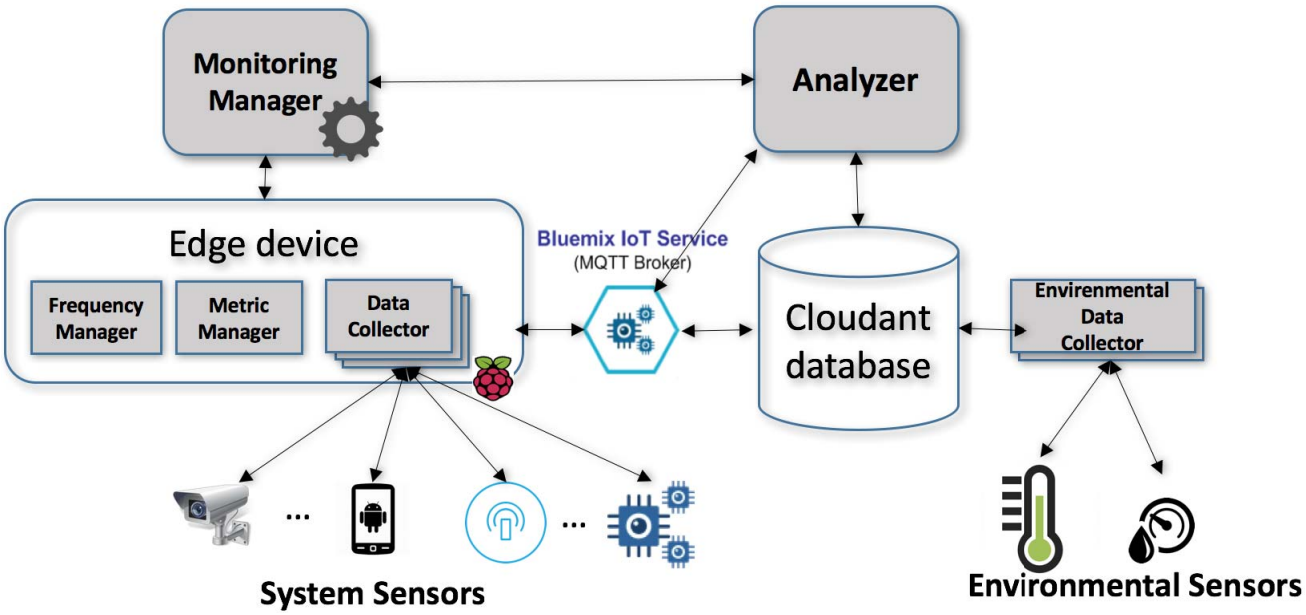


**Figure 1:** Architecture Overview

sumes the data from Cloudant, applies the optimization model described in Section IV, and determines the optimal metrics to be monitored and the optimal frequencies. The optimization model is solved via CPLEX [24], which has a collable library in Java. The output of the *Analyzer* is sent to the *Monitoring Manager* as a REST POST query. The *Monitoring Manager* has the list of devices and their capacities in term of monitoring. Upon receiving the set of metrics to be monitored and their frequencies, it sends the needed REST queries to the right targets to add or remove metrics through the *Metric Managers* and to tune the frequencies through the *Frequency Managers*.

It should be noted that in production environments it is recommended to use a broker like Bluemix®IoT broker to register all the sensors. But for the sake of simplicity, in our use case, we did not register the environmental sensors to the broker. Instead, we set them to target the Cloudant database using its URL without passing through the broker.

## VI. Conclusion and Future Work

Given the limited resources for metric monitoring in IoT environments, it is essential to provide a mechanism that determines which metrics to monitor and at which frequencies, with the objective of increasing the monitoring frequencies of metrics that have a higher importance as much as possible. In this paper, we presented a novel approach that provides such mechanism in which we iteratively determine the metrics to be monitored, assign them weights of importance and then optimize, via a mathematical model, the frequency at which each metric is to be monitored. We presented a proof-of-concept implementation of our approach and showed its effectiveness. Our approach is also iterative. That is, metrics to be monitored and their frequencies get changed whenever an environmental and/or metric-value event triggers a need for that change.

There are several directions for future research to this work. First, one of our implicit assumptions in this paper is that our optimization model can be solved (as well as implemented) in significantly less time than the next monitoring update. We note that the model might become computationally intractable for large real-sized problem instances and thus its solution time to optimality or even sub-optimality might be fast enough for our approach to work, satisfying the aforementioned assumption. That is because of the huge number of integer variables that would arise in these instances [25], [26]. Thus, one direction for future work is to either provide an alternative formulation that is easier to solve for large-sized instances, or come up with approximation algorithms that might give sub-optimal solutions but in reasonbale times. One example for an alternative formulation is column generation, where the decision variables would be different monitoring schedules that are to be generated in an iterative approach [27].

Second, we note that our current formulation of the mathematical model enforces beginning the monitoring of any metric in one of the first $k$ periods if the chosen monitoring

frequency of that metric is $k$. An interesting extension is to relax this restriction and let the model determine a starting time stamp for monitoring that is higher than $k$ but maybe lower than a certain predetermined threshold. Then, numerical comparisons between the performance of our model and this extension can be carried out.

Third, our approach works for a single tenant environment. A realistic extension is the multi-tenant case of our problem. Several complexities arise in multi-tenant environments, where resources are shared and some metrics to be monitored might or might not be common across different tenants. Ideas from the ride-sharing and the resource sharing operations research literature can be borrowed in solving this extension.

## REFERENCES


[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (IoT): A vision, architectural elements, and future directions," *Future Gener. Comput. Syst.*, vol. 29, no. 7, September 2013.

[2] A. Noronha, R. Moriarty, K. OĆonnell, and N. Villa", "Attaining iot value: How to move from connecting things to capturing insights, gain an edge by taking analytics to the edge," Cisco white paper, Tech. Rep., 2014.

[3] J. Rivera and R. van der Meulen", "Gartner says 4.9 billion connected 'things' will be in use in 2015,," 2014. [Online]. Available: http://www.gartner.com/newsroom/id/2905717

[4] M. Mohamed, "Generic monitoring and reconfiguration for service-based applications in the cloud," PhD, Institut National des Télécommunications, November 2014. [Online]. Available: https://tel.archives-ouvertes.fr/tel-01123740

[5] M. Barcelo, A. Correa, J. Llorca, A. M. Tulino, J. L. Vicario, and A. Morell, "IoT-Cloud service optimization in next generation smart environments," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 4077–4090, 2016. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2016.2621398

[6] S. D. T. Kelly, N. K. Suryadevara, and S. C. Mukhopadhyay, "Towards the implementation of IoT for environmental condition monitoring in homes," *IEEE Sensors Journal*, vol. 13, no. 10, October 2013.

[7] R. Baldoni, R. Beraldi, S. Piergiovanni, and A. Virgillito, "Measuring notification loss in publish/subscribe communication systems." in *Proceedings of the 10th IEEE Pacific Rim International Symposium, Dependable Computing*, March 2004.

[8] G. Aceto, A. Botta, W. de Donato, and A. Pescapé, "Cloud monitoring: A survey," *Computer Networks*, vol. 57, no. 9, pp. 2093 – 2115, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128613001084

[9] D. Trihinas, G. Pallis, and M. D. Dikaiakos, "AdaM: An adaptive monitoring framework for sampling and filtering on IoT devices," in *IEEE International Conference on Big Data*. Los Alamitos, CA, USA: IEEE Computer Society, 2015.

[10] D. Jeswani, M. Natu, and R.-K. Ghosh, "Adaptive monitoring: A framework to adapt passive monitoring using probing," in *Proceedings of the 8th International Conference on Network and Service Management*, ser. CNSM '12. Laxenburg, Austria, Austria: International Federation for Information Processing, 2013, pp. 350–356.

[11] D. Jeswani, M. Natu, and R. K. Ghosh, "Adaptive monitoring: Application of probing to adapt passive monitoring," *Journal of Network and Systems Management*, vol. 23, no. 4, pp. 950–977, 2015. [Online]. Available: http://dx.doi.org/10.1007/s10922-014-9330-8

[12] M. A. Munawar, T. Reidemeister, M. Jiang, A. George, and P. A. S. Ward, *Adaptive Monitoring with Dynamic Differential Tracing-Based Diagnosis*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[13] M. Natu and A. S. Sethi, *Probabilistic Fault Diagnosis Using Adaptive Probing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-75694-1_4

[14] M. Doraisamy, S. bin Ibrahim, and M. N. Mahrin, "Metric based software project performance monitoring model," in *Proceedings of the IEEE International Conference on Open Systems (ICOS)*. IEEE, August 2015.

[15] Y. Cheng, W. Chen, Z. Wang, and X. Yu, "Performance-monitoring-based traffic-aware virtual machine deployment on numa systems," *IEEE Systems Journal*, vol. PP, no. 99, 2015.

[16] G. Katsaros, G. Gallizo, R. Kübert, T. Wang, J. O. Fitó, and D. Henriksson, "A Multi-level Architecture for Collecting and Managing Monitoring Information in Cloud Environments." in *Proceedings of the third International Conference on Cloud Computing and Services Science, CLOSER*, F. Leymann, I. I. Ivanov, M. van Sinderen, and B. Shishkov, Eds. SciTePress, 2011.

[17] S. Clayman, A. Galis, and L. Mamatas, "Monitoring virtual networks with Lattice." in *Network Operations and Management Symposium Workshops*, April 2010.

[18] M. L. Massie, B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: design, implementation, and experience." *Parallel Computing*, vol. 30, no. 7, 2004.

[19] N. Entreprises, "Nagios Documentation," http://www.nagios.org/documentation, 2014.

[20] J. Brandt, A. Gentile, J. Mayo, P. Pebay, D. Roe, D. Thompson, and M. Wong, "Resource monitoring and management with OVIS to enable HPC in cloud computing environments." in *IEEE International Symposium on Parallel Distributed Processing*, May 2009.

[21] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, March 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2600239.2600241

[22] M. Hadley and P. Sandoz, *JAX-RS: The Java API for RESTful Web Services*, Java Specification Request (JSR) 311, Java Community Process Program Std., September 2009. [Online]. Available: http://www.jcp.org/en/jsr/detail?id=311

[23] MQTT Community, "MQTT: A M2M/Internet of Things Connectivity Protocol," http://mqtt.org/, 21 February 2017, [Online; accessed 15-January-2017].

[24] I. I. CPLEX, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.

[25] G. L. Nemhauser and L. A. Wolsey, "Integer programming and combinatorial optimization," *Wiley, Chichester. GL Nemhauser, MWP Savelsbergh, GS Sigismondi (1992). Constraint Classification for Mixed Integer Programming Formulations. COAL Bulletin*, vol. 20, pp. 8–12, 1988.

[26] L. A. Wolsey, *Integer programming*. Wiley New York, 1998, vol. 42.

[27] C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. Savelsbergh, and P. H. Vance, "Branch-and-price: Column generation for solving huge integer programs," *Operations research*, vol. 46, no. 3, 1998.