# GaruaGeo: Global Scale Data Aggregation in Hybrid Edge and Cloud Computing Environments

Otávio Carvalho, Eduardo Roloff and Philippe O. A. Navaux
*Informatics Institute, Federal University of Rio Grande do Sul, Brazil*

Keywords: Edge Computing, Fog Computing, Cloud Computing, Internet of Things, Distributed Computing.

Abstract: The combination of Edge Computing devices and Cloud Computing resources brings the best of both worlds: Data aggregation closer to the source and scalable resources to grow the network on demand. However, the ability to leverage each time more powerful edge nodes to decentralize data processing and aggregation is still a significant challenge for both industry and academia. In this work, we extend the Garua platform to analyze the impact of a model for data aggregation in a global scale smart grid application dataset. The platform is extended to support global data aggregators that are placed nearly to the Edge nodes where data is being collected. This way, it is possible to aggregate data not only at the edge of the network but also pre-process data at nearby geographic areas, before sending data to be aggregated globally by global centralization nodes. The results of this work show that the implemented testbed application, through the usage of edge node aggregation, data aggregators geographically distributed and messaging windows, can achieve collection rates above 400 million measurements per second.

## 1 INTRODUCTION

The ubiquity of the Internet of Things (IoT) unleashes the potential of using innovations based on sensor data to improve society's overall quality of life. These data profiles can be used to enable technologies such as large-scale smart cities deployments, smart home monitoring, smart industries and smart energy grids.

Devices that provide these kinds of capabilities are now widespread through our cities and homes on devices such as smartphones, medical devices, home appliances and street signals.

By 2025, researchers estimate that the IoT will have a potential economic impact of $11 trillion per year – which would be equivalent to about 11% of the world economy. They also expect that one trillion IoT devices will be deployed by 2025 (Buyya and Dastjerdi, 2016).

Novel technologies for mobile computing and the Internet of Things (IoT) are shifting the focus of its research and development to computing toward dispersion. In this context, Edge Computing is one of the most prominent areas, and it is a new paradigm in which a high volume of computing and storage resources – generally referred to as cloudlets, micro datacenters or fog nodes – are placed at the Internet's edge close to mobile devices or sensors (Satyanarayanan, 2017).

However, there is a large set of applications that cannot tolerate the latency penalties of sending data to be aggregated on the cloud and then waiting for the response on edge nodes. Also, the act of sending a potentially large number of small packets to the cloud for data processing can saturate the network and decrease the scalability of applications (Dastjerdi and Buyya, 2016).

Smart grids will allow consumers to receive near real-time feedback about household energy consumption and price, allowing consumers to make informed decisions about their spending. For energy producers, it will be possible to leverage home consumption data to produce energy forecasts, enabling near real-time actions and better scheduling of energy generation and distribution (Brown, 2008). In this way, smart grids will save billions of dollars in the long run, for consumers and the generators, as well as to reduce the impact on the environment, according to recent forecasts (Reuters, 2011).

This paper uses a realistic application use case and dataset in order to further understand what is achievable when certain computational tasks are moved from cloud nodes to edge nodes, and the potential



Carvalho, O., Roloff, E. and Navaux, P.
GaruaGeo: Global Scale Data Aggregation in Hybrid Edge and Cloud Computing Environments.
DOI: 10.5220/0007734604370447
In *Proceedings of the 9th International Conference on Cloud Computing and Services Science (CLOSER 2019)*, pages 437-447
ISBN: 978-989-758-365-0

benefits of data aggregation and batching at the edge of the network.

In order to improve over our previous work a new communication layer was introduced, which is called the aggregator layer. This layer has the primary goal of pre-processing data in the same geographic location of edge nodes. This way, it is possible to avoid high-cost communication delays by pre-processing data from multiple edge nodes in the same geographic region, postponing the latency overhead of communicating with potentially distant cloud nodes. Finally, the scalability of the architecture is evaluated by deploying the platform in a large scale deployment in a public cloud environment.

Our results show, after including the aggregator layer, the platform can achieve data collection rates above 400 million measurements per seconds, using 15 geographic distributed datacenters on Microsoft Azure platform, for a total of 1366 machines across the globe. It also presents an eight times speedup improvement in throughput in comparison to the previous architecture, in a scenario using eight aggregators in the same geographic region.

The rest of this paper is organized as follows. The Section 2 presents and compares the related work with our work. In Section 3 our mechanism is presented and explained. Section 4 presents the results of our experimental evaluation. Finally, the Section 5 concludes this work and presents the next research directions.



## 2 RELATED WORK

The main works on the state-of-the-art of edge computing focus on platforms and frameworks aiming to provide scalable processing as close as possible to the network border. These approaches are primarily focused on providing the lowest possible latency results and better utilize the resources available on the network. Multi-access Edge Computing (MECs) and Cloudlets, which can be described as cloud-like deployments at the network edge, are currently the predominant approaches to address these challenges.

The most prominent approaches and how they relate to this work are briefly described below. At the end of Section 2 there are detailed explanations of how these works relate to the proposed solution on the Section 3.

The state-of-the-art works include FemtoClouds (Habak et al., 2015), REPLISOM (Abdelwahab et al., 2016), Cumulus (Gedawy et al., 2016), CloudAware (Orsini et al., 2016), ParaDrop (Liu et al., 2016), HomeCloud (Pan et al., 2016), ENORM (Wang et al., 2017), RT-SANE (Singh et al., 2017), EdgeIoT (Sun and Ansari, 2016) and cloud provider based implementations (Tärneberg et al., 2016), which are examples of applications that explore computational offloading to nearby devices.

One important thing to perceive is that the majority of the related works either rely on offloading computation to edges that are underutilized or offload processing to nearby network centralizers (modified wireless network access points or specialized mobile network base station hardware). EdgeIoT (Sun and Ansari, 2016) stands apart on this aspect by exploring computation to considerably more performance nodes, by relying on virtual machines in a nearby mobile base station.

Several works on the state-of-the-art either rely on hardware specific tools or significant modifications on their underlining communication protocols. This work, on the other hand, relies on standard tools and protocols that add flexibility and turn easier to port it to other platforms.

Examples of applications which require significant changes in the underlining communication protocols or specific hardware are ParaDrop, a specific edge computing framework implemented on WiFi Access Points (APs) or other wireless gateways (such as set-top boxes). It uses a lightweight virtualization framework through which third-party developers can create, deploy, and revoke their services in different APs.

HomeCloud (Pan et al., 2016) focus on being an open and efficient new application delivery in edge cloud, by integrating two complementary technologies: Network Function Virtualization (NFV) and Software-Defined Networking (SDN).

EdgeIoT, in its turn, is an architecture to handle data stream at the mobile edge. The central idea consists of fog nodes communicating with a Virtual Machine (VM) positioned at a nearby base station. On the top of the fog nodes, the Software Defined Networking (SDN) based cellular core is designed to facilitate the package forwarding among fog nodes.

Cumulus (Gedawy et al., 2016), FemtoClouds (Habak et al., 2015), CloudAware (Orsini et al., 2016) and RT-SANE (Singh et al., 2017) have a greater focus on task placement, providing dynamic scheduling capabilities for operators and tasks (such as tasks migration functionalities) which are beyond the scope of this work. Cumulus (Gedawy et al., 2016) provides a complete framework that controls task distribution under a heterogeneous set of devices on its cloudlet. FemtoClouds (Habak et al., 2015) and CloudAware (Orsini et al., 2016) monitor device usage on its cloudlets in order to improve

Table 1: Research scope comparison of the state-of-the-art with the proposed work.

| Name | Cloud | Edge | Mobility | Large Scale | Hardware Agnostic |
|---|---|---|---|---|---|
| GaruaGeo (this work) | • | • | | • | • |
| ENORM | • | • | | | • |
| RT-SANE | • | • | | | • |
| Tarneberg et al. | • | • | | | • |
| HomeCloud | • | • | | | • |
| CloudAware | • | • | • | | |
| FemtoClouds | | • | • | | |
| REPLISOM | | • | • | | |
| Cumulus | | • | • | | • |
| ParaDrop | | • | | | • |
| EdgeIoT | | • | • | | |

device usage as part of its scheduling algorithms. CloudAware (Orsini et al., 2016) also provides a specific Application Programming Interface (API) to improve the user experience of software developers. RT-SANE (Singh et al., 2017) evaluates several scheduling heuristics in comparison to a cloud-only scenario.

Although the related works present multiple initiatives on Edge Computing towards computational offloading, only the work of Tärneberg et al. (Tärneberg et al., 2016), ENORM (Wang et al., 2017) and RT-SANE (Singh et al., 2017) explore the potential of combining low latency edge nodes processing with scalable and more powerful sets of commodity machines on public clouds. However, ENORM (Wang et al., 2017) evaluation only considers a small set of edge nodes communicating with a nearby Amazon AWS cloud node in Dublin. RT-SANE (Singh et al., 2017), in its turn, relies on a specific fog simulator to obtain its results, which limits the scope in comparison to a real-world evaluation. Finally, the work of Tärneberg et al. (Tärneberg et al., 2016) directly offload from the edges devices to the cloud infrastructure, which limits the scope of the work in relation to data aggregation.

Apart from that, most works presented in this section either rely on generated datasets or small datasets (tenths of mobile devices) for its evaluations. On the other hand, the present work uses a realistic dataset, based on a real-world dataset from household energy consumption in Germany (Ziekow and Jerzak, 2014).

In Table 1, we present a comprehensive description of the coverage of the state-of-the-art in comparison with this work.

According to characterization, it is possible to perceive that the proposed model (GaruaGeo) is the only work that focuses on large scale hybrid (cloud and edge) data processing.

This work also does not focuses on mobility issues or hardware specific implementations such as several works on the state-of-the-art.

The taxonomy used to group the state-of-the-art works is based on recent surveys on fourth-generation distributed stream processing systems, fog computing and edge computing (de Assuncao et al., 2017) (Atzori et al., 2010) (Mao et al., 2017) (Mahmud et al., 2018).

# 3 ARCHITECTURE AND IMPLEMENTATION

The architectural infrastructure of the testbed application deployment can be described as a composition of four layers, as it is represented on Figure 1: (1) Cloud layer, which executes long-running scalable jobs that can provide more powerful machines for processing with the trade-off of greater latencies; (2) Aggregator layer, which aggregates from multiple edge nodes, usually placed on the same geographic region as edge nodes, in order to aggregate all data for a given geographic region before sending data to the cloud layer, which is potentially placed in a distant geographic region; (3) Edge layer, which is composed of edge nodes that are used to pre-process and aggregating sensor data before sending to the Aggregator nodes; and the (4) Sensor layer, which is composed of the sensors that communicate directly with Edge layer nodes to receive actuation requests and provide measurements to the network.

### 3.1 Cloud Layer

This layer is composed of virtual machines that execute the application in order to aggregate data to be received from Edge layer nodes. The Cloud layer should be composed of elements that can be able to process data as it arrives. It can be instantiated by implementing the same application logic from the layer below, but instead, it should be configured to receive data from multiple edge nodes.

This layer receives data from queues and exchanges through a message hub, so that the inputs can be parallelized through multiple consumers. It can also be configured to support clusters of machines to execute transformations as distributed stream processing jobs over these queues and exchanges.

In the evaluation work, the model is implemented as an application running in a single node inside a Linux VM at Microsoft Azure, which was chosen due to its perceived benefits over other cloud platforms (Roloff et al., 2012) (Roloff et al., 2017). This application was written in the Go programming language and receives the processing request from the layer below through GRPC communication framework. The VM instance utilized was configured as it is described in Table 2.

### 3.2 Aggregator Layer

The aggregator layer represents one or multiple intermediate layers of aggregation that could be potentially used to mitigate the impact of latency between data collection and the collection of global metrics into the Cloud layer.

In previous works, it was possible to observe that is possible to obtain significant throughput gains by aggregating data from sensors on edge nodes.

However, it was perceived that there was room for improvement if it was possible to have an operator in the architecture responsible for aggregate data for specific regions before communicating with the cloud layer. An example of this scenario is when multiple edge nodes in Japan are transferring data to the USA, each one of them paying the latency penalty of

Table 2: Cloud layer configuration: Virtual machine type and toolset description.

| Parameter | Description |
|---|---|
| Instance Type | Basic_A3 (4 cores, 7 GB RAM) |
| Operating System | Ubuntu 16.04 LTS |
| Golang version | 1.8 |
| GRPC version | 1.3.0-dev |
| Protocol Buffers version | 3.2.0 |

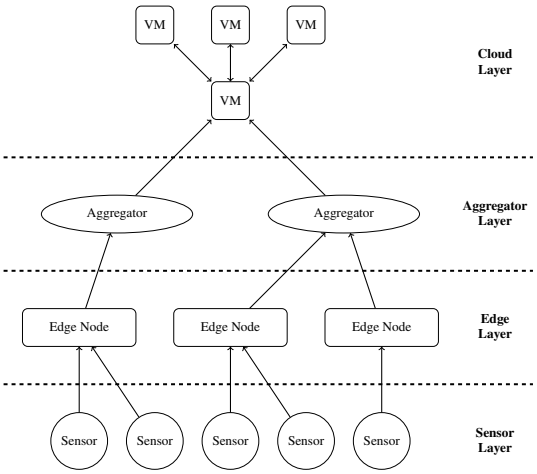


Figure 1: The architecture is composed by 4 layers: Cloud, Aggregator, Edge and Sensor.

communicating with another continent. Instead, aggregators could be placed in this location and aggregate data from multiple edge nodes in Japan, before transferring it to the USA.

In our evaluation, this layer was represented as a single core machine, in order to potentially represent less powerful machines, such as Raspberry Pi's and similar ARM processors which are well-suited for the given scenario (and that were used for the edge layer on previous works).

### 3.3 Edge Layer

The Edge layer is composed of a set of nodes with transformation operators to apply over sensor measurements one-at-time. Operators can be either transformations over results, combinations with sets of measurements received or mappings to machines on the Cloud layer above.

On this layer, the application code will be expressed to define which computation will be done inside of edge nodes and which computation will be managed by VMs on the Cloud Computing environment. The degree of control provided by this level makes it possible to decrease the number of messages sent to the Cloud. In this way, it is possible to decrease the amount of data that is sent to the Cloud.

Table 3: Aggregator layer configuration: Virtual machine type and toolset description.

| Parameter | Description |
|---|---|
| Instance Type | Standard_DS2_v2 (2 cores, 7 GB RAM) |
| Operating System | Ubuntu 16.04 LTS |
| Golang version | 1.8 |
| GRPC version | 1.3.0-dev |
| Protocol Buffers version | 3.2.0 |

Also, by processing certain amounts of data directly on the edge nodes, the latency experienced by actuator sensors is in the order of tenths of milliseconds instead of a couple of seconds of cloud processing latencies.

Actuator sensor logic can also be implemented on this layer, in such a way that when a given condition is matched by an edge node, it can trigger actuators on the Sensor layer in order to act on external applications. For example, a consumer can configure its smart grid energy meter to maintain the energy consumption below a certain level during peak cost energy hours. In this way, the smart grid meter can turn off certain machines when the average consumption reaches a certain threshold.

In our previous works, edge layers were composed by a set of Raspberry Pi Zero W machines. In this paper, in order to stress the simulation in a globally distributed environment, these machines are simulated as VMs on Microsoft Azure. These machines communicate with the sensor layer, which in our evaluation will also be a simulated layer. The configuration of the edge nodes is the same as we have used for the aggregator nodes, which is described in detail on Table 4.

## 3.4 Sensor Layer

The Sensor layer is represented by a given set of sensors that communicate with the Edge nodes. Ideally, sensors should communicate with Edge Nodes through their available input/output hardware interconnections or lightweight wireless connection such as Bluetooth or LTE networks. However, in order to limit the analysis scope of this work, the sensor network dataset is previously loaded into edge nodes before the execution of tests.

Smart grid environments rely on specific meters and plugs on households to collect data, which are provided by the energy grid provider or standardized to support only a set of accepted and verified plugs and meters types. The data types generated by these environments also need to respect a certain schema to be shared, aggregated and analyzed by the energy provider companies.

Table 4: Edge layer configuration: Virtual machine type and toolset description.

| Parameter | Description |
|---|---|
| Instance Type | Standard_DS1_v2 (1 cores, 3.5 GB RAM) |
| Operating System | Ubuntu 16.04 LTS |
| Golang version | 1.8 |
| GRPC version | 1.3.0-dev |
| Protocol Buffers version | 3.2.0 |

The dataset used for the evaluation of this work is based on the dataset provided by the 8th ACM International Conference on Distributed Event-Based Systems (DEBS). This conference provides competitions with problems which are relevant to the industry. In the year 2014, the conference challenge focus was on the ability of Complex Event Processing (CEP) systems to apply on real-time predictions over a significant amount of sensor data. For this purpose, household energy consumption measurements where generated, based on simulations driven by real-world energy consumption profiles, originating from smart plugs deployed in households (Ziekow and Jerzak, 2014). For this challenge, a large number of smart plugs has been deployed in households with data being collected roughly every second for each sensor in each smart plug.

## 3.5 Communication Protocol

Although multiple protocols for communication in IoT systems have been proposed in the recent years, the protocols in use today are still being evaluated and are subject of discussion and standardization initiatives, mainly due to advancements of internet protocols to support mobile and IoT applications. The most widely adopted protocols in use today are, respectively, MQTT (Banks and Gupta, 2014) and CoAP (Bormann et al., 2012).

One of the most prominent proposals on this area is the HTTP/2 protocol. The standard was finished in 2005 and provides several improvements over previous protocols, mainly due to the capability of multiplexing data, avoiding handshake overhead and their data compression capabilities (Belshe et al., 2015) (Ruellan and Peon, 2015).

As an alternative to broker-centric communication protocols and synchronization costly protocols such as REST (Richardson and Ruby, 2008), Google Inc. has adopted a RPC protocol and service discovery framework Stubby/Chubby (Burrows, 2006). The open source version of its tool is called GRPC (Google, 2015), which relies on HTTP/2 in order to avoid handshake overhead, and Protocol Buffers (Gligorić et al., 2011) to communicate using a binary method, which provides better data compaction by reducing the message size.

Due to the performance benefits reported from the usage of HTTP/2 protocols over standard HTTP, and their ease of use for flexible prototyping of distributed applications, GRPC was used to build a reliable and fast communication channel for all of the communication layers implemented on this work.

GRPC as a communication platform presents several advantages over TCP only connections and communication protocols such as REST. It has a simple interface which hides configuration complexity but is still able to provide high-level features, such as long-lived connections (to avoid unnecessary communication handshakes) and communication multiplexing inside a small number of channels (reducing the number of required connections open at a given point in time). However, their usage is still subject of evaluation, mainly on networks with high package loss percentages, which are a limiting factor not only for HTTP/2 but also for AMQP based applications (Goel et al., 2016) (Lee et al., 2013) (Chowdhury et al., 2015) (Thangavel et al., 2014).

### 3.6 Measurement Algorithm

In order to proper extract insights from sensor data, it was selected a broadly used approach to aggregate and generate Short-Term Load Forecasting (STLF) for smart grids. This algorithm is not only well-known by the research community, but also provides the potential to be applied at multiple aggregation layers.

Smart grids bring a rich set of tools to better control and balance energy supply and demand in near real-time, as well as continuous visibility of consumption patterns about energy generation and consumption. This way, methods to extract knowledge from near real-time observations are critical to the extraction of value from infrastructure investment.

In this scenario, Short-Term Load Forecasting (STLF) describes the prediction of power consumption levels in several time frames, from minutes and hours up to a week ahead. It considers variables such as date, temperature (including weather forecasts), humidity, temperature-humidity index, wind-chill index and most importantly, historical load. Residential versus commercial or industrial uses are rarely specified.

Multiple approaches for time series modeling for STLF have been developed over the last thirty years. There methods (Kyriakides and Polycarpou, 2007) could be briefly summarized into the following groups:

- Regression models that represent electricity load as a linear combination of multiple parameters e.g. weather factors, day type and customer class.
- Methods based on linear time series including those derived from Autoregressive Integrated Moving Average (ARIMA) model and State-Space Models (SSMs).
- Methods derived from SSMs that rely on a filtering-based (e.g., Kalman filters) techniques and a characterization of dynamical systems.
- Nonlinear time series modeling through machine learning methods such as non-linear regression.

According to the research of Shawkat Ali et al. (Ali, 2013), the three most accurate models for load prediction are respectively, Multiplayer Perceptron (MLP), Support Vector Machines and Least Mean Squares. In this work, it is implemented an approach which mixes MLP and ARIMA, bringing together characteristics from both linear time series methods and SSMs (Bylander and Rosen, 1997). This approach was chosen not only due to the model suitability to distributed architectures, but also due to its similarity to the model proposed at the DEBS 2014 conference (Ziekow and Jerzak, 2014). This approach is schematically described in Equation (1).

More specifically, the set of queries provide a forecast of the load for: (1) each house, i.e., house-based and (2) for each individual plug, i.e., plug-based. The forecast for each house and plug is made based on the current load of the connected plugs and a plug specific prediction model.

$$L(s_{i+2}) = \frac{avgL(s_i) + median(avgL(s_j))}{2} \quad (1)$$

In the Equation (1), $avgL(s_i)$ represents the current average load for the slice $s_i$. The value of $avgL(s_i)$, in case of plug-based prediction, is calculated as the average of all load values reported by the given plug with timestamps $\in s_i$. In case of a house-based prediction the $avgL(s_i)$ is calculated as a sum of average values for each plug within the house. $avgL(s_j)$ is a set of average load value for all slices $s_j$.

$$s_j = s_{i+2-n*k} \quad (2)$$

In the Equation (2), $k$ is the number of slices in a 24 hour period and $n$ is a natural number with values between 1 and $floor(\frac{i+2}{k})$. The value of $avgL(s_j)$ is calculated analogously to $avgL(s_i)$ in case of plug-based and house-based (sum of averages) variants.

## 4 EVALUATION

The main idea of adding aggregators to the platform was that, by including a layer near to the edge nodes, the latency experienced by the edge nodes would decrease and it would be possible to decide, on aggregator nodes, when it was necessary to suffer the latency penalties to communicate with potentially distant cloud nodes.

After adding the new layer, some scenarios would be required to be validated in order to verify that our platform was able to provide its expected benefits in comparison to the previous version without aggregators.

Aggregators have the ability to receive, buffer, pre-process and group messages before sending data to the cloud layer nodes. This additional layer gives the platform the ability to answer edge layer node requests faster than the cloud layer nodes, which could potentially be placed at other countries and continents, providing slower responses due to the greater latencies implied in communication between machines in distant geographical regions.

## 4.1 Infrastructure Setup and Operators Placement

Before the execution of the tests, it was essential to setup a minimal set of tools to help on the automation of the deployment process in multiple regions around the globe.

Since we are using Microsoft Azure, it was possible to rely on ARM (Azure Resource Manager) templates to describe the infrastructure to setup. By using ARM templates, it was possible to define generic templates to describe the number of nodes, size of virtual machines, operating system version and default tools that were important to have pre-installed on all machines.

The decision of which regions to use in our analysis scenarios came from a previous latency measurement evaluation done on previous works.

In this evaluation, it was analyzed the communication between machines placed in multiple regions and one node placed on US West, which in this paper was the region where the master node of the cloud layer was placed. After this analysis was made, these regions were classified regions into three categories:

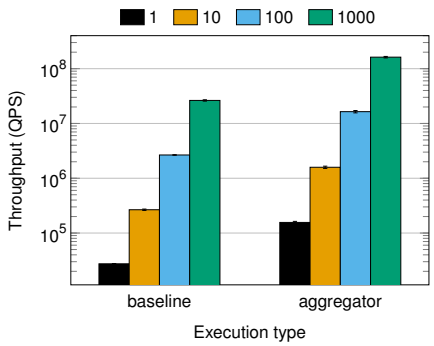


Figure 2: Aggregator stress comparison analysis: 90 nodes maximum throughput with and without an aggregation layer (1 to 1000 message batch sizes).

Table 5: Region profiles: Description of the latency profiles between Azure regions selected for evaluation

| Region | Low latency | Medium latency | High latency |
|---|---|---|---|
| westus2 | • | | |
| brazilsouth | | • | |
| ukwest | • | | |
| southeastasia | | • | |
| eastasia | | | • |
| japaneast | | • | |
| westeurope | • | | |
| australiasoutheast | | | • |
| northeurope | | • | |
| centralus | • | | |
| southindia | | | • |
| canadacentral | • | | |
| centralindia | | • | |
| koreasouth | | | • |
| francecentral | | • | |

Low latency, medium latency, and high latency. The results with regions selected and their latency profiles are displayed on Table 5.

## 4.2 Exploring the Impact of Adding Aggregators into the Infrastructure

In this testbed evaluation, aggregators nodes are placed in the same regions (datacenters) as the simulated edge nodes. Using this approach messages are buffered into a nearby aggregator, where they can be aggregated or pre-processed and sent to a centralized node which is potentially in a distant region.

The first experiment we have made was to evaluate the impact of adding another moving piece to our infrastructure. The rational behind it was to evaluate how much it would impact the performance in a stress scenario. In Figure 2 we evaluate a stress scenario with a single master, before and after adding an aggregator.

From this experiment, it was possible to perceive that the performance improves for smaller groups of messages, but in general, including an aggregator does not impact the performance negatively in comparison to the previous architecture. In comparison to the previous architecture, it is a worst-case evaluation scenario, since with an aggregator there is an extra layer of communication overhead. Apart from that, in this scenario the aggregator could not benefit from batching of multiple messages since batching,

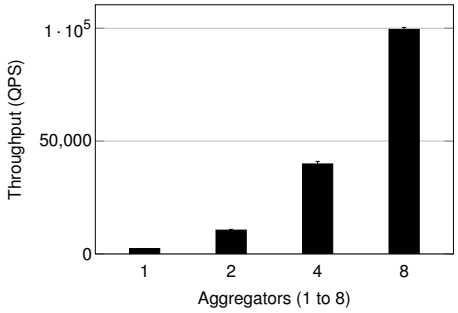


Figure 4: Aggregator groups evaluation: 40 edge nodes distributed between distinct groups of aggregators.

in this case, is done at edge node level. The potential performance benefits of using one or multiple aggregators should be perceived only in the communication between aggregator and master nodes (where it is possible to wait or batch multiple edge node messages before communicating with nodes on the cloud layer).

### 4.3 Multiple Aggregators in a Given Global Region

In our previous version of the architecture, which was composed by a master node on the cloud layer and edge nodes that received sensor data, it was possible to validate that it the architecture was able to scale linearly up to 90 nodes (regarding throughput). In this experiment, after the addition of our new processing layer, it was collected data to understand if this behavior has not changed.

As it can be seen on Figure 3, regarding batches of messages, it is still possible to perceive that the architecture can scale linearly. However, regarding the number of nodes, it is not possible to perceive any performance gains when we increase the number of nodes from 15 to 90. This behavior could be explained by the fact that our aggregators are now in the same geographic region as our edge nodes.

In our previous analysis (which considered cloud nodes and edges nodes only), cloud nodes were placed in a region and edge nodes were placed in another. In this analysis, from the standpoint of the master node a stress scenario was never achieved (the upper bound of the system regarding throughput). However, in the current scenario, the only piece of the architecture which is not in the same region is the master node. Hence, the latency experienced by our edge nodes is much lower, and the upper bound of the system is experienced by all scenarios from 15 to 90 nodes.

This analysis has shown three crucial facts: 1) The throughput from the standpoint of the edge nodes has significantly improved since aggregator nodes now handle requests in the same geographic region with lower latencies; 2) By adding aggregator nodes in the same geographic regions as the edge nodes which collect sensor data, we are now bounded by the communication between the aggregator nodes and cloud nodes (which will always be slower than the communication between edge nodes and aggregator nodes); 3) In order to evaluate the overall performance (regarding throughput and latency) of the system, it is not possible anymore to aggregate the number of requests made from edge nodes to their counterparts (cloud nodes in the previous architecture or aggregator nodes in the current), but it is required to evaluate the performance of the communication between aggregator nodes and the cloud node.

### 4.4 Groups of Aggregators into a Single Region

In this experiment, it was evaluated the impact of adding multiple aggregators into the same local region. The analysis was made using VMs on the uswest region of Azure with a fixed number of edge nodes (40 edge nodes) and a variable number of aggregators (1 to 8 aggregators).

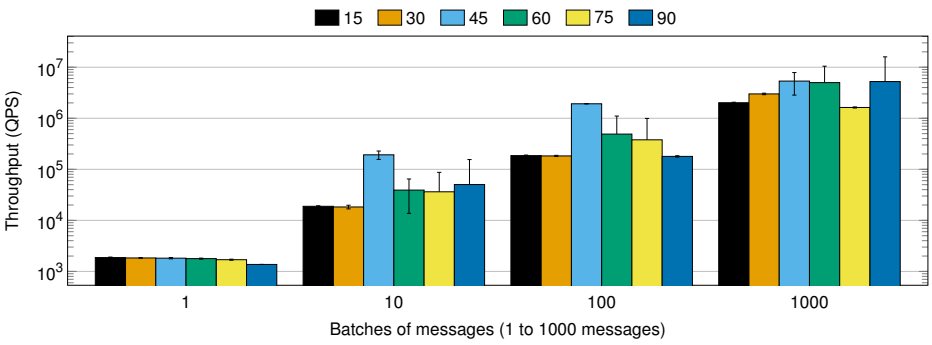


Figure 3: Aggregator stress evaluation: Throughtput analysis from 15 to 90 nodes.

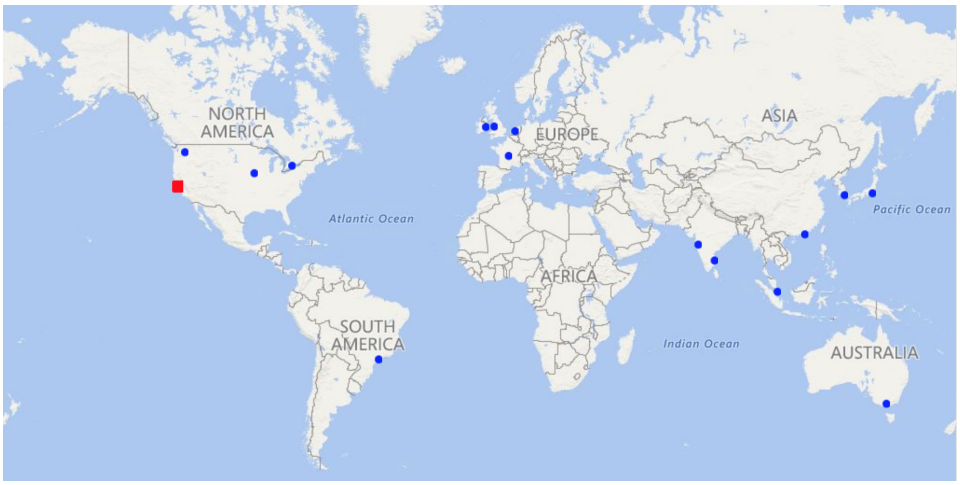


Figure 5: Global scale deployment: One global master (red square) and 15 regions (blue dots, where each region contains 1 aggregator node and 90 edge nodes).

In order to execute this analysis, the number of existent edge nodes was spread evenly between aggregators. In Figure 4, it was possible to verify that by adding 8 aggregators it was possible to achieve exponential increase on throughput up to 8 aggregators, where the aggregator has achieved a throughput of around 100k messages per second received from the edge nodes.

Multiple aggregators perform better probably due to the number of concurrent requests that one aggregator can answer at the same time. Given a certain amount of edge nodes, spreading their requests between multiple aggregator nodes decreases contention level in each aggregator. This way, they can answer more requests per second since the number of nodes for each aggregator is more balanced. However, introducing a more significant amount of aggregators will require that more elements pay the extra latency needed to communicate with cloud layer nodes, but these trade-offs were not explored in this scenario.

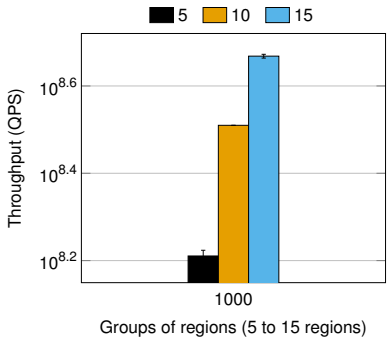


Figure 6: Global performance evaluation: 90 nodes per regions, 1000 messages batches and variable number of regions (5 to 15).

## 4.5 Multiple Region Edge Analysis

In this evaluation scenario, it was explored the maximum achievable performance from the standpoint of data collection on edge nodes. In order to evaluate it, we have deployed a large number of machines, into multiple geographic regions, and collected data about the aggregated data collection throughput of its edge nodes.

Each scenario evaluated on these tests relies on regions (datacenters) on Microsoft Azure across the globe. In each region, there were placed 90 Edge nodes and a single aggregator node. The analysis was made based on data collected from at least 20 executions for 5, 10 and 15 regions. In each execution scenario, each edge node sends at least 100k energy measurements to its respective aggregator nodes.

In Figure 5 it is possible to visualize the extension of the largest execution. For this analysis it was used: 15 regions on Azure; one Global aggregator node on the cloud layer; 15 aggregator nodes on the aggregator layer; 1350 Edge nodes, for a grand total of 1366 machines aggregating data across the globe. In this experiment, it was possible to achieve data collection rates above 400 million measurements per seconds on the scenario with 15 aggregator nodes.

Another important aspect captured by this analysis is the discrepancy in performance between distinct regions. From the standpoint of the Edge nodes, it was not possible to perceive significant performance discrepancies between regions for each one of the analyzed regions, as it is displayed in Figure 7.

Linear scalability is also obtained as we increase the batch factor on the number of messages which are aggregated before being sent to the aggregator nodes. Hence, it was possible to validate in this scenario that

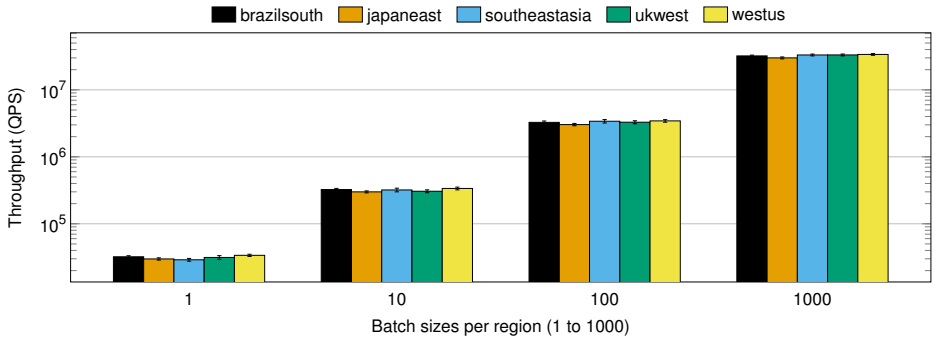


Figure 7: Global performance evaluation: 90 nodes per region, 5 regions and variable batch sizes (1 to 1000 messages).

the pattern analyzed in the scenario with a single region in Figure 3 holds significant for multiple regions.

Finally, it is shown in Figure 6 a summary of the highest throughput scenarios observed, which are those that display the largest amount of messages per batch evaluated. The results show that the performance does not only increases linearly with batch sizes, but also data collection rates, which increase linearly as more regions are added globally. From these tests, up to 15 regions (which use 90 nodes and one aggregator per region), it was not possible to achieve an inflection point where data collection rates start to decrease.

## 5 CONCLUSION AND FUTURE WORK

In this work, it was analyzed a model for workload distribution and data aggregation using a large-scale smart grid application dataset. The main contribution was the addition of new processing layer, responsible for aggregating data from the geographic regions they were placed before communicating with remote cloud nodes. In summary, the application was able to achieve higher throughput by leveraging processing on edge nodes, batching techniques and data aggregation to reduce the communication overhead with distant nodes.

Our results show, after including the aggregator layer, the platform can achieve data collection rates above 400 million measurements per seconds. These results were obtained leveraging 15 geographic distributed regions on the Microsoft Azure platform, for a total of 1366 machines around the globe. In comparison to the previous architecture, the new architecture presents an eight times throughput improvement in a scenario using 8 aggregators in the same geographic region. The platform was able to scale linearly according to the number of messages on window batches, as well as in relation to the number of nodes and regions added to the platform, up to 90 nodes per region.

The experiments show that the impact of windowing and aggregation on edge nodes is not negligible and needs further investigation by the research community. Although it has similarities to data stream processing research; the topic is still being initially explored by researchers on the fields of the Internet of Things, Fog Computing and Edge Computing.

The future works should focus on the exploration of other scheduling, windowing and aggregation techniques for edge processing.

Apart from that, a critical line of research would be to explore how to evolve this testbed application and its middleware into a generic framework for applications that need to distribute processing through edge and cloud nodes. Finally, it would be essential to explore other communication protocols to understand their suitability to multiple scenarios based on hybrid computations on cloud and edge environments.

## ACKNOWLEDGMENTS

This research received partial funding from CYTED for the RICAP Project. It has also received partial funding from the EU H2020 Programme and from MCTI/RNP Brazil under the HPC4E project, grant agreement no. 689772.

Additional funding for this research was provided by FAPERGS in the context of the GreenCloud Project.

Microsoft has provided cloud computing instances on Microsoft Azure to the execution of the experiments.

# REFERENCES


Abdelwahab, S. et al. (2016). Replisom: Disciplined tiny memory replication for massive iot devices in lte edge cloud. *IEEE Internet of Things Journal*.

Ali, A. B. M. S. (2013). *Smart Grids: Opportunities, Developments, and Trends*. Springer.

Atzori, L., Iera, A., and Morabito, G. (2010). The internet of things: A survey. *Computer networks*, 54(15):2787–2805.

Banks, A. and Gupta, R. (2014). MQTT Version 3.1. 1. *OASIS standard*.

Belshe, M. et al. (2015). Hypertext transfer protocol version 2 (HTTP/2). *Internet Engineering Task Force (IETF) - RFC-7540*.

Bormann, C. et al. (2012). CoAP: An application protocol for billions of tiny internet nodes. *Internet Computing*.

Brown, R. E. (2008). Impact of Smart Grid on Distribution System Design. In *Power and Energy Society General Meeting-Conversion*. IEEE.

Burrows, M. (2006). The Chubby lock service for loosely-coupled distributed systems. In *Proceedings of the 7th symposium on Operating systems design and implementation*. USENIX.

Buyya, R. and Dastjerdi, A. V. (2016). *Internet of Things: Principles and paradigms*. Elsevier.

Bylander, T. and Rosen, B. (1997). A Perceptron-like Online Algorithm for Tracking the Median. In *Neural Networks, 1997., International Conference on*. IEEE.

Chowdhury, S. A. et al. (2015). Is HTTP/2 more energy efficient than HTTP/1.1 for mobile users? *PeerJ PrePrints*.

Dastjerdi, A. V. and Buyya, R. (2016). Fog Computing: Helping the Internet of Things Realize Its Potential. *Computer*.

de Assuncao, M. D., da Silva Veith, A., and Buyya, R. (2017). Distributed data stream processing and edge computing: A survey on resource elasticity and future directions. In *Journal of Network and Computer Applications*. Elsevier.

Gedawy, H. et al. (2016). Cumulus: A distributed and flexible computing testbed for edge cloud computational offloading. In *Cloudification of the Internet of Things (CIoT)*. IEEE.

Gligorić, N. et al. (2011). Performance evaluation of compact binary XML representation for constrained devices. In *Distributed Computing in Sensor Systems and Workshops, International Conference on*. IEEE.

Goel, U. et al. (2016). HTTP/2 Performance in Cellular Networks. In *ACM MobiCom*.

Google (2015). gRPC Motivation and Design Principles.

Habak, K. et al. (2015). Femto clouds: Leveraging mobile devices to provide cloud service at the edge. In *Cloud Computing (CLOUD), IEEE 8th International Conference on*.

Kyriakides, E. and Polycarpou, M. (2007). Short Term Electric Load Forecasting: A Tutorial. In *Trends in Neural Computation*. Springer.

Lee, S. et al. (2013). Correlation analysis of MQTT loss and delay according to QoS level. In *Information Networking (ICOIN), International Conference on*. IEEE.

Liu, P. et al. (2016). ParaDrop: Enabling Lightweight Multi-tenancy at the Network's Extreme Edge. In *Edge Computing (SEC), IEEE/ACM Symposium on*.

Mahmud, R., Kotagiri, R., and Buyya, R. (2018). Fog computing: A taxonomy, survey and future directions. In *Internet of everything*, pages 103–130. Springer.

Mao, Y., You, C., Zhang, J., Huang, K., and Letaief, K. B. (2017). A survey on mobile edge computing: The communication perspective. *IEEE Communications Surveys & Tutorials*, 19(4):2322–2358.

Orsini, G. et al. (2016). CloudAware: A Context-Adaptive Middleware for Mobile Edge and Cloud Computing Applications. In *Foundations and Applications of Self* Systems, IEEE International Workshops on*.

Pan, J. et al. (2016). HomeCloud: An edge cloud framework and testbed for new application delivery. In *Telecommunications (ICT), 23rd International Conference on*. IEEE.

Reuters (2011). U.S. Smart Grid to Cost Billions, Save Trillions.

Richardson, L. and Ruby, S. (2008). *RESTful web services*. O'Reilly Media, Inc.

Roloff, E. et al. (2012). High Performance Computing in the Cloud: Deployment, performance and cost efficiency. In *4th International Conference on Cloud Computing Technology and Science (CloudCom)*. IEEE.

Roloff, E. et al. (2017). HPC Application Performance and Cost Efficiency in the Cloud. In *25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*.

Ruellan, H. and Peon, R. (2015). HPACK: Header Compression for HTTP/2. *Internet Engineering Task Force (IETF) - RFC-7541*.

Satyanarayanan, M. (2017). The Emergence of Edge Computing. *Computer*.

Singh, A., Auluck, N., Rana, O., Jones, A., and Nepal, S. (2017). Rt-sane: Real time security aware scheduling on the network edge. In *Proceedings of the10th International Conference on Utility and Cloud Computing*, pages 131–140. ACM.

Sun, X. and Ansari, N. (2016). EdgeIoT: Mobile Edge Computing for the Internet of Things. *IEEE Communications Magazine*.

Tärneberg, W., Chandrasekaran, V., and Humphrey, M. (2016). Experiences creating a framework for smart traffic control using aws iot. In *Proceedings of the 9th International Conference on Utility and Cloud Computing*, pages 63–69. ACM.

Thangavel, D. et al. (2014). Performance evaluation of MQTT and CoAP via a common middleware. In *9th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*. IEEE.

Wang, N., Varghese, B., Matthaiou, M., and Nikolopoulos, D. S. (2017). Enorm: A framework for edge node resource management. *IEEE Transactions on Services Computing*.

Ziekow, H. and Jerzak, Z. (2014). The DEBS 2014 Grand Challenge. In *Proceedings of the 8th ACM DEBS Conference*, volume 14.