UNIVERSIDADE DO VALE DO RIO DOS SINOS
ACADEMIC GRADUATION UNIT
BACHELOR DEGREE'S COURSE IN COMPUTER SCIENCE

GUILHERME GABRIEL BARTH

PROFOG: A PROACTIVE ELASTICITY MODEL FOR FOG COMPUTING-BASED IOT APPLICATIONS

São Leopoldo
2020

Guilherme Gabriel Barth

PROFOG: A PROACTIVE ELASTICITY MODEL FOR FOG COMPUTING-BASED IOT APPLICATIONS

Graduation Research presented as partial requisite for obtaining a Computer Science's Bachelor degree by Universidade do Vale do Rio dos Sinos — UNISINOS

: Prof. Dr. Rodrigo da Rosa Righi

São Leopoldo
2020

PROFOG: A PROACTIVE ELASTICITY MODEL FOR FOG COMPUTING-BASED IOT APPLICATIONS

Guilherme Gabriel Barth[1]
Rodrigo da Rosa Righi[2]

Abstract: Internet of Things has been without a doubt on the rise over the last few years as more and more companies, in varying fields, start to automate their processes with the help of Internet-Of-Things (IoT). IoT systems are often connected to critical processes of companies, such as production line monitoring, requiring reliable and scalable environments. Cloud Computing has been prevailing as an architecture of choice for IoT implementations due to its highly scalable nature. However, that has brought forth challenges related to security, Quality of Service, network congestion, and latency. Fog Computing has been on the rise as an alternative to address those challenges as it allows not only increased security and reduced latency but also reduced network congestion. Multiple studies suggest different models for Cloud-Fog architectures for IoT implementations, mostly focusing on task execution and job scheduling, but not thoroughly addressing elasticity control mechanisms. In this context, this article presents ProFog, a proactive elasticity model for Cloud-Fog architectures for IoT scenarios. ProFog makes use of the Auto-Regressive Integrated Moving Average (ARIMA) mathematical formalism to predict load behaviors and trigger scaling actions as close to when they are required as possible, allowing the delivery of new resources prior to reaching an overloaded state. We developed a prototype in order to validate ProFog on a simplified use case and its results were compared to those of a reactive model. The results showed an improvement of 11,21% in energy consumption in favor of ProFog due to data smoothing achieved by the ARIMA predictions and open way for additional studies on proactive elasticity for fast deployment environments.

Keywords: Cloud Computing, Fog Computing, IoT, resource management, elasticity

1 INTRODUCTION

The advance of the Internet-of-Things (IoT) industry is connected to the needs of many segments of the economy, such as the automation in industries as part of Industry 4.0 (I4.0) (MASOOD; SONNTAG, 2020), Smart Cities, transportation and healthcare (MOHAMED, 2017). Over the past few years, more industries started to adhere to the automation of processes and the implementation of IoT applications and the tendency is for that only to grow further as the results of such implementations start to be seen. Therefore, it is expected for IoT implementations to grow as well based on the continuous and increasing demand from other industries. According to Cisco (CISCO, 2020), the number of devices connected to the network will be close to 30 billion by the end of 2023.

The applications of IoT are numerous. Internet of Things allows monitoring of machines and processes on manufacturing (Mourtzis; Vlachou; Milas, 2016), stock control on retail, security

[1]Graduando em de Ciência da Computação pela Unisinos. Email: guilherme_barth96@hotmail.com
[2]Mini-currículo do orientador. Email: rrrighi@unisinos.br

and management on cities, patient monitoring on healthcare (FISCHER et al., 2020), and many others. Each of these applications has its own set of peculiarities which makes IoT systems rather complex. These systems may be used from data collection for long-term actions - such as predictive maintenance - to emergency actions - such as healthcare (MAHMUD; KOCH; BUYYA, 2018) and Smart Manufacturing (Yin; Luo; Luo, 2018). The IoT devices and format of data used on each kind of scenario also vary. For this variety of requirements and use cases, IoT systems must be as flexible and manageable as possible. This heterogeneity of scenarios and devices is one of the main challenges of IoT along with the collaboration of systems, data transmission, management of resources, and energy consumption (Hu et al., 2019).

Relying only on a local central processing server to address all the operations of IoT implementations would result in high costs on infrastructure and maintenance as it would require a very robust server and network to avoid overloading the server or flooding the network with data. In most industrial scenarios, IoT systems are connected to critical applications that require high availability from the server given that failure of the system may result in significant financial losses for the company or even damage to machinery. Cloud computing is often used as an approach in such cases to take advantage of its scalability and high availability with reduced infrastructure costs. However, sending requests to a Cloud server adds a constant latency overhead to the communication (STANKOVSKI et al., 2016) that can not be accepted in some cases, such as healthcare scenarios as response time is critical for the system.

As explained by Vaquero, the existence of billions of devices constantly producing data on the edge of the network may cause the network to become a bottleneck (VAQUERO; RODERO-MERINO, 2014). Using Cloud Computing as the main paradigm for IoT scenarios will only further contribute to network congestion (Yin; Luo; Luo, 2018) as the "Internet is not scalable and efficient enough to handle IoT big data" as explained by Xiang (Sun; Ansari, 2016). Submitting massive loads of data to a Cloud Computing server may also result in high costs for the service as providers may charge based on the amount of data going into the Cloud.

To allow further development and implementation of IoT, it is necessary to design new architectures and solutions that allow higher scalability while reducing the number of requests on the network and maintaining the required Quality of Service (QoS). Fog Computing has been gaining position as an architecture of choice for IoT scenarios due to its concept of keeping the processing near to its data sources, thus allowing reduced latency, stable QoS (SUGANUMA, 2018) and addressing another of the key concerns of IoT which is the data security.

Like Cloud Computing, Fog Computing also has its set of disadvantages when it comes to IoT systems. Fog nodes, unlike Cloud servers, are often limited in hardware capabilities, therefore they are not appropriate for CPU intensive tasks such as data analytics, which is one of the key components for turning data into information on IoT systems. In order to address the many requirements of IoT systems and achieve a highly scalable and reliable architecture, Fog Computing may be used in combination with Cloud Computing to provide the best of both architectures into a single environment that is able to cover the multiple needs of IoT systems efficiently.

The majority of studies released to date which address this kind of scenario focuses on task scheduling and resource allocation algorithms based on different criteria, not giving too much attention to the elasticity control technologies behind the maintenance of the system's operations. Elasticity control strategies may be classified as reactive or proactive. The first one is based on if-then rules and pre-set thresholds which scale the system when certain conditions are met, providing a simple approach for elasticity control. This strategy however may result in tardy scaling decisions, delivering resources too late and subjecting the system to an overloaded state in the meantime, or increased energy consumption due to unnecessary scaling. The second strategy on the other hand makes use of different prediction techniques to anticipate the system's behavior and ensure that resources are delivered on time, avoiding overloaded states and unpredicted errors.

In this study, we present ProFog, a proactive elasticity model for resource management on Cloud-Fog environments for IoT implementations. ProFog uses the mathematical formalism ARIMA (AutoRegressive Integrated Moving Average) to predict the system's behavior and deliver new resources as close to when they are required as possible, allowing elasticity with minimum energy consumption and operating costs.

In section 2 of this article, we present the background information that is necessary to understand the concepts later referred to in the study. Section 3 describes related work in the area. Over the course of section 4, we present the ProFog model in detail, explaining its architecture and technologies. Section 5 introduces our prototype, workload, and evaluation metrics used to analyze its results. Section 6 comprises a review of the results of the prototype tests. At last, section 7 provides the scientific contributions, challenges, and future work involving the solution.

2 BACKGROUND

This section explores and defines the most important topics for the understanding and development of Internet of Things systems, covering architecture, commonly used technologies, challenges, and future direction.

2.1 IoT Architecture

Internet-of-Things (IoT) systems are composed of a number of physical objects that are connected to the internet and constantly produce loads of data based on the environment they are applied to (Mourtzis; Vlachou; Milas, 2016). The end devices, data output format, frequency, and application vary across the many different IoT use cases. However, the massive quantity of collected data is a constant for large IoT implementations. IoT has been on the rise over the last few years and its use and research have been expanding throughout several different areas, such as maintenance of factories, supply chain management, agriculture, and healthcare (SUGANUMA, 2018).

In order to address the huge loads of data generated by IoT systems, Cloud Computing has been prevailing as the architecture of choice for those systems as it offers computing resources, storage capacity, scalability, and reliability (MAHMUD; KOCH; BUYYA, 2018). However, many difficulties have emerged with the use of Cloud Computing as a center of IoT systems such as “increased network loads, response delays, and privacy concerns”, as per Takuo (SUGANUMA, 2018). Certain IoT applications can not be subject to such response delays though, like healthcare applications where IoT is implemented mainly for automation of patient monitoring and emergency response. Data security is another stable concern when it comes to IoT systems and Cloud Computing. As IoT devices are directly connected to internal systems and confidential data, having a direct connection between the edge devices and the internet may pose risk for data leakage and Cyber-Physical attacks (HE et al., 2016).

Due to those concerns and difficulties, Fog computing has been gaining space as an alternative to Cloud computing. Fog computing allows the processing of data to be conducted near to its data sources or control object, thus reducing the network load and delays as well as privacy concerns. Many architectural models making use of different technologies were proposed for IoT systems. However, none of those models has become a standard for the industries.

2.2 Cloud and Fog Computing

Cloud Computing systems are characterized by the provisioning of On-Demand computing services over the internet, allowing service consumers to pay only for what is used and to use only what is required (Abbasi et al., 2019) without the need to maintain a local server. These services are divided on three categories: 1. Platform as a Service (PaaS), allowing the deployment of applications and use of the Cloud as their platform; 2. Infrastructure as a Service (IaaS), offering computing, storage, and networking resources on-demand to consumers along with control over the operating system and applications, giving more flexibility than PaaS; 3. Software as a Service (SaaS), offering a specific service over the internet on-demand to consumers.

These processes rely on server elasticity which is the virtualization of resources whenever the load the server is submitted to requires so. That can be achieved by virtualizing machines that act as processing environments - known as horizontal elasticity - and dividing the load of a central server in many different nodes, increasing the overall throughput of the server (Agarwal; Yadav; Yadav, 2015), or by adding extra resources to existing virtualized machines - known as vertical elasticity.

Cloud Computing offers a set of benefits to consumers - which highly boosted its acceptance by companies - such as: 1. Reduced infrastructure costs as companies do not need to maintain expensive servers to cover their demands or employees to maintain the same; 2. Scalability is provided by the Cloud provider which allows companies to scale its services on demand rather than maintaining oversized servers to cover for occasions that lead to unusual server load, such as Black Friday for retail stores; 3. Redundancy and reliability against hardware failures are handled by the Cloud provider as they are responsible for the server and the adherence to service level agreements (SLAs) related to server availability; 4. Reduced landscape complexity as the system is maintained by a separate company and so is its security.

Cloud Computing is rather attractive to companies as it allows them to spare expenses on hardware and maintenance and to focus on business, however it is also associated with its own set of challenges, such as: 1. Security and privacy concerns as the data resides outside of the company's firewall; 2. System availability as the system is maintained by a different company and SLAs may not be maintained; 3. Performance is limited by bandwidth and latency on the communication, affecting the user experience with the application.

To address the challenges of Cloud Computing implementation in the presence of the advance of Internet-of-Things, Cisco introduced the concept of Fog Computing. Similarly to Cloud Computing, this technology may provide computing, storage, and networking services in a virtualized platform. Unlike Cloud Computing, Fog Computing is typically positioned close to the end devices, on the edge of the network (BONOMI et al., 2012). One may say that Fog is a Cloud that is closer to the user.

Fog Computing systems are essentially composed of multiple Fog nodes - which are individual processing units - that provide services to edge devices. The services provided by Fog nodes are numerous varying according to the application of the system, from failure detection on assembly lines to streaming services on sports events. Edge devices also vary based on the IoT application, from mere sensors to Smart Cars. Fog Computing nodes are limited on processing capacity, making the use of VMs for virtualization too resource-demanding, fostering the use of containers as a light-weight virtualization option that is also capable of faster deployment incurring in better overall response times in communication (Yin; Luo; Luo, 2018). The positioning of the Fog Computing architecture is strategical and connected to many other characteristics that make it a promising architecture for Internet-of-Things implementations:

- Higher number of nodes;
- Wide-spread geographical distribution;
- Low latency due to edge positioning;
- Increased security;
- Heterogeneity.

The proximity of the Fog to the end user allows lower latency on the communication and facilitates tasks of maintaining the required Quality-of-Service (QoS) - of extreme importance to several IoT applications. Fog Computing also allows higher control and security over the data as Fog nodes may be placed behind the company's firewall and configured as required to meet the company's security standards. Fog Computing was not designed to be a substitute for Cloud Computing, but to extend it and address its challenges, allowing the development of applications and services that could not be achieved with Cloud alone (BONOMI et al., 2012). By properly integrating the two different architecture paradigms, it is possible to achieve high scalability with reduced latency and higher security standards while also reducing the amount of data on the network, preventing network congestion.

2.3 Load Prediction

Performance is a constant concern on large scale systems, especially on IoT implementations as the complexity of scenarios, number of connected devices and aggregated data can easily reach massive numbers. Maintaining application requirements such as QoS is a challenging task that can only be fulfilled by the implementation of proactive resource management on the system and that can only be achieved through the use of intelligent load prediction techniques.

Load prediction consists on forecasting resource consumption within the infrastructure based on aggregated data of resource monitoring. This allows for proactive management actions to be taken before a state of depletion of resources is reached. Working on an overloaded state may result in a series of undesired effects on the system, such as high response times, timeouts, denial-of-service, or unforeseen application errors (Righi et al., 2019). There are two steps in load prediction: the first one is the collection of data on certain metrics that represent the system's state and the second is the analysis of the collected data with the use of algorithms to predict future resource utilization rates.

Data is typically organized as time series - a set of observations that represent sequential events over a period of time - to allow the analysis of the system's state over time. Depending on its characteristics, time series can be classified as deterministic or stochastic processes. Deterministic processes are defined by known mathematical functions, meaning future values can be calculated through the use of formulas. They may be represented by linear models and further classified as stationary or non-stationary. Stationary time series are those that display no tendency over time while non-stationary time series tend to increase or decrease over time, both tendencies may occur at different points as well. Multiple methods are commonly used for linear time series analysis, such as Moving Average (MA), Autoregressive (AR), AR Integrated MA (ARIMA), and Holt-Winters.

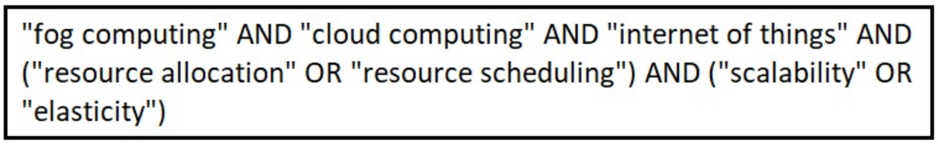
Stochastic processes on the other hand can not be defined by any known mathematical model, so their values can not be formally calculated. Therefore, stochastic processes require non-linear models. Non-linear time series are often analyzed by Artificial Neural Networks (ANN) as they are capable to learn and adapt from the data without prior knowledge of the model (Righi et al., 2019). Neural Networks may produce more accurate results in comparison to time-series models, but they also require more data and extended learning periods to achieve such results which may be prohibitive for use on critical applications, such as medical use cases.

3 RELATED WORK

This section reviews related articles and their research on the subject to determine the most commonly used technologies and architecture organizations for IoT scenarios and Fog Computing. These areas are a constant point of study for researchers and are under constant and quick change. Therefore, in order to filter out possibly outdated information, we used a filter not to return any article released before 2016.

We performed the search for related articles using a query string that encapsulated general terms used on IoT and Fog Computing scenarios: Fog Computing, Cloud Computing, Internet of Things, scalability, elasticity, and resource management. We chose Google Scholar to perform the query for articles as it simultaneously queries multiple scientific repositories, such as IEEE Xplore Digital Library, ScienceDirect, ACM Digital Library, and Springer Library. The following exclusion criteria (EC) were also used to further filter articles returned on the search:

1. EC1: Articles that were not written in English;
2. EC2: Articles that performed surveys on the subject;
3. EC3: Articles that performed reviews on the subject.



"fog computing" AND "cloud computing" AND "internet of things" AND ("resource allocation" OR "resource scheduling") AND ("scalability" OR "elasticity")

Figure 1 – Query string used to search for related work.

3.1 State-of-the-art

Inspired by IoT adoption rate and Cisco's predictions for data traffic and number of devices connected to the network, many researchers aimed efforts at the development of Fog Computing enablement technologies. Yin's work proposed the use of a request evaluator as the first module on Fog Computing nodes, allowing to evaluate the complexity of a task based on its calculated computing time to decide where such task should be addressed to meet its requirements. In case the request evaluator decides the task can not be addressed locally - on the edge device - and requires to be executed by the Fog or Cloud, the same is directed to a task scheduler that is responsible for the decision if the task will be computed on the Cloud or the Fog. The task scheduler evaluates the resource demand of a task and the calculated resource threshold of a period of time for the Fog to determine the allocation. A local resource manager on each of the Fog nodes is responsible for the initialization of tasks as well as resource reallocation to tasks based on delay constraints (Yin; Luo; Luo, 2018). Yin's work focused on proper resource allocation for QoS adherence and reduction of communication to remote servers, minimizing network congestion risks.

Ranesh's study used a resource ranking approach - based on available processing, latency, and bandwidth - for dynamic resource allocation on Cloud-Fog architectures as a mean to achieve optimal response times. The resource provisioning method included data transmission as a metric for the provisioning decision along with a hierarchical approach to minimize latency, response times, and network congestion by giving priority to the lowest members of the hierarchy whenever possible (NAHA et al., 2020).

Chen (Chen et al., 2017) on the other hand proposed a four-layer architecture - consisting of IoT, Middleware, Fog, and Cloud - where the majority of the services are provided by the Cloud initially and the same are allocated to the Fog upon demand. In his work, the middleware is responsible for receiving tasks from IoT devices and performing job classification and resource scheduling. Job classification is based on data privacy and QoS requirements, tasks subject to data privacy are directly assigned to a local Fog node to ensure security while non-privacy-sensitive tasks have their QoS requirements evaluated to determine which nodes of the system are capable of meeting the QoS. Based on the results of the classification, the middleware schedules tasks to the Fog or Cloud based on an operational cost evaluation performed by the resource scheduler with the use of predefined costs. The system monitors resource utilization on the Fog and Cloud layers and determines resource allocation needs and the pool of resources for scheduling based on predefined threshold settings.

Small's work proposed a middleware solution for microservice-based orchestration of applications on multi-tier IoT infrastructures as a mean to allow reduced communication to remote servers (Small et al., 2017). His solution orchestrated the deployment of services to Cloud, Fog, or Mist layers based on the requirements of the services and available resources on the layers. The Cloud layer used OpenStack hosted VMs, having the capacity to boot VMs ahead of time, reducing deployment time, while services on the Fog were provided by containers instantiated on demand.

Cisco's initial Fog Computing model definition did not specify hardware-specific details for Fog nodes, such as computing and storage power. To this day, many authors have proposed different kinds of hardware configurations to allow different Fog use cases. Jianhua's work proposed a multi-tier Fog architecture model to allow complex analytical tasks to take place on Fog level and reduce the load of requests directed to Cloud servers for data analytics scenarios. The proposed multi-tier model was composed of two layers of Fog nodes, A-Fog (ad-hoc) and D-Fog (dedicated), which operated on analytics tasks based on their complexity (He et al., 2018). A-Fog was composed of low computing power devices while D-Fog was formed by a cluster of servers, allowing the execution of complex data analytics without a Cloud server. Resource allocation was based on utility cost as a measure of decision rather than QoS adherence as in some of the previous studies.

Instead of focusing on resource allocation on Fog or Cloud, Alsaffar proposed a collaboration strategy between Fog Computing and Cloud Computing to allow the execution of more complex tasks. On his work, a Fog broker evaluated received requests and in case its node was not capable of processing it within its SLAs, it contacted a service monitor server located on the Cloud which hosted information about VM availability on all of the Cloud and Fog environments. Based on the availability information retrieved from the service monitor server, the Fog broker divided the job into multiple blocks for distribution across available VMs. Upon the conclusion of the processing, each remote VM returned the data for aggregation to the initial Fog broker. The requests were evaluated based on predefined categories. The proposed approach supported the collaboration of Fog nodes for the execution of CPU intensive tasks, offering an option for big data management. However, this process may also take too much processing of the Fog layer for scheduling, data division, and aggregation processes (ALSAFFAR, 2016).

Mohammed proposed a Fog-2-Fog collaboration system to allow better resource utilization and load distribution across Fog nodes. Mohammed designed algorithms for resource reallocation decisions using multiple delay metrics such as service delay, propagation delay, and computational delay (AL-KHAFAJIY et al., 2019). The decision of when to offload work from a Fog node to another was based on two conditions: 1. Evaluation if one or more services in the queue would miss their deadline; 2. Comparison of service arrival rate to service output on the node.

Nguyen's study (Nguyen et al., 2020) pointed out that merely monitoring server metrics, such as CPU load and memory consumption, is not an appropriate solution to address the heterogeneity of applications comprised by IoT. On that note, Nguyen proposed ElasticFog, a framework built on top of Kubernetes for dynamic resource allocation of container-based applications on Fog Computing. ElasticFog monitors network traffic at each Fog node location and uses that as an affinity rule on Kubernetes to improve resource allocation decisions. The evaluation of the solution showed significant improvements in throughput and latency in favor of ElasticFog when compared to Kubernetes' default scheduling mechanism.

3.2 Analysis and research opportunities

By analyzing the related works - refer to table 1 - we can see that the majority of studies focus on task scheduling and resource allocation algorithms based on different criteria, from security to utility cost and QoS adherence. This approach allows the evaluation of tasks and their deployment to the most suitable environment for their execution. However, it does not fully address the scalability management side of resource provisioning.

In order to create a highly reliable environment that allows the maintenance of QoS for a wide range of IoT implementations, it is important to implement load prediction algorithms and make use of proactive elasticity as a mean to scale the system before the same reaches an overloaded state, which may affect the QoS and create unpredicted errors on the server.

4 PROFOG MODEL

This section describes the ProFog model, a model that uses proactive elasticity to improve resource allocation for IoT applications on Fog Computing. Figure 3 depicts the composition of the model. The following sections address its design decisions and underlying algorithms of the core components.

4.1 Design Decisions

Cloud computing models face many challenges when it comes to IoT implementation, such as data security, latency, network congestion, and cost. Fog Computing offers a way to address these challenges as the positioning of the Fog allows for increased security, reduced latency, and reduced risk of network congestion. However, Fog Computing systems do not have as much processing and storage capacity as Cloud Computing systems, thus we can not expect them to be able to perform all the operations that rely on Cloud Computing. Considering the varying requirements of different Internet of Things scenarios, a Cloud-Fog architecture is currently the most promising solution for IoT implementation in the long term as it allows for a wider variety of scenarios to be implemented while also providing additional security and control over the data, reducing network congestion and facilitating QoS adherence.

In order to meet QoS requirements and create a stable and reliable environment for the execution of IoT services, it is essential for the system to make use of elasticity control techniques and scale itself on demand. Reactive elasticity models trigger scaling operations once the server load reaches a pre-set threshold. Scaling operations are not instant, though - they require a certain deployment and start-up time until the resources are available for use. This may cause the applications to operate in an overloaded state until the resources become available, possibly resulting in a series of undesired effects on the system, like high response times, timeouts, denial-of-service, or unforeseen application errors (Righi et al., 2019).

Proactive elasticity models on the other hand apply the collected load data to predictive algorithms to estimate future load and trigger scaling operations before such load is reached, allowing for resources to be available before the server reaches an overloaded state, avoiding system errors. Given the importance of stability and QoS for IoT applications, we have decided to make use of proactive elasticity on ProFog. Unlike Cloud servers, Fog nodes are limited on processing capacity, limiting the use of vertical elasticity. For that reason, we have chosen to apply horizontal elasticity instead by instantiating new service instances on idle Fog nodes.

The proposed architecture is divided on three different physical layers: Internet-of-Things, Fog Computing, and Cloud Computing. The IoT layer is composed of network edge devices that collect data to be analyzed, interact with its surroundings or require services made available

Article	Focus	Elasticity on Cloud	Elasticity on Fog	Resource collaboration
(Yin; Luo; Luo, 2018)	Resource scheduling and allocation	NA	Vertical elasticity using resource reallocation w/ custom algorithms	NA
(NAHA et al., 2020)	Resource scheduling and allocation	NA	NA	NA
(Chen et al., 2017)	Job Classification + Resource scheduling and allocation	Reactive	Reactive	NA
(Small et al., 2017)	Application deployment orchestration with resource scheduling	Proactive	NA	NA
(He et al., 2018)	Resource scheduling for data analytics	NA	NA	NA
(ALSAFFAR, 2016)	Resource allocation based on collaboration between Fog and Cloud Computing	NA	NA	X
(AL-KHAFAJIY et al., 2019)	Load balancing based on collaboration between Fog nodes	NA	NA	X
(Nguyen et al., 2020)	Resource allocation	NA	Reactive	X

Table 1 – Related works comparison. NA is used to indicate that the item was not addressed by the author of the work. X indicates the item is addressed on the related work.

on the Fog Computing layer- these devices may vary from simple sensors to smart devices, like Smart Cars. The Fog layer is composed of multiple Fog nodes which perform initial data treatment and provide time sensitive services for the IoT layer and other service consumers. The Cloud layer is responsible for the monitoring and scaling of the system as a whole - the Cloud and the Fog nodes - and also for performing non-time-sensitive and CPU intensive tasks, such as data analytics. Figure 2 provides a high-level overview of a Cloud-Fog architecture for IoT implementation.

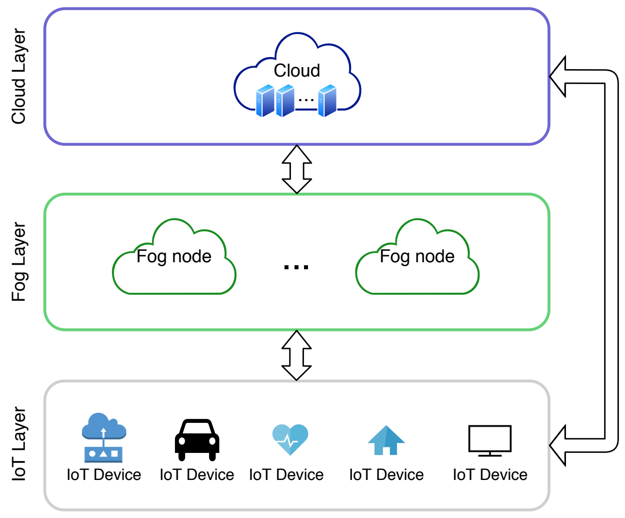


Figure 2 – High-level view of a Cloud-Fog architecture.

4.2 Architecture

ProFog focuses on elasticity for the execution of time-critical applications on Fog Computing. For that purpose, we apply load prediction algorithms to time series data in order to determine resizing needs and trigger proactive scaling. ProFog acts as a middleware by managing the deployment and execution of different services and applications in a seamless manner. Figure 3 provides an overview of the core components of the model, highlighting in red the ones that are part of ProFog. The model is composed of three physical layers - Cloud Computing, Fog Computing, and Internet-of-Things. However, ProFog is distributed only on the Cloud Computing and Fog Computing layers.

ProFog is composed of three different modules: 1. Cloud Manager; 2. Fog Manager; 3. Elastic Manager. The Cloud Manager is responsible for triggering the deployment of services to the Fog layer based on requests from the IoT devices, load balancing, and for the communication with Fog nodes about load and triggering scaling actions on the Fog and Cloud layers. The Fog Manager downloads application images from the Cloud and instantiates them as containers on the Cloud Manager's request while also constantly monitoring its load and updating the Cloud Manager about the same. The Elastic Manager receives data about the load of the

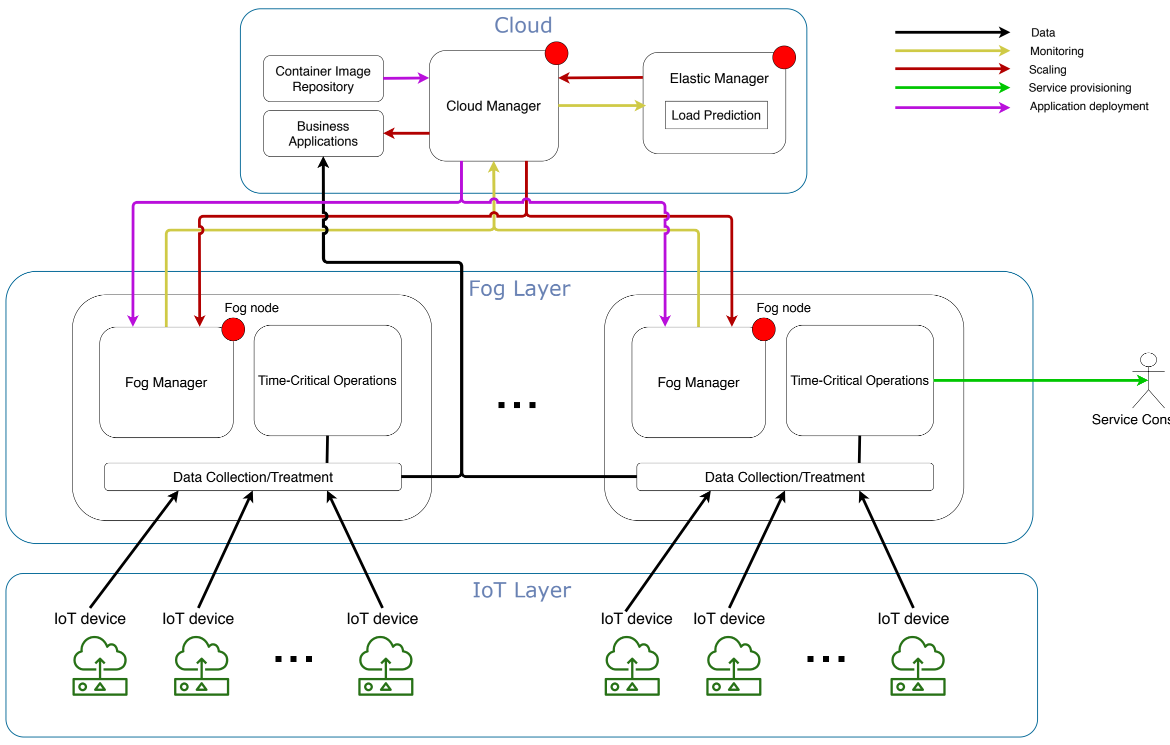


Figure 3 – ProFog model applied to Cloud-Fog environment. Elements highlighted with a red dot represent the main modules of ProFog.

Fog nodes and Cloud applications from the Cloud Manager, applies the collected data to load prediction algorithms, and informs the Cloud Manager about its scaling decisions.

ProFog requires the deployment of a Fog Manager to all Fog nodes of the architecture and of a Cloud Manager and an Elastic Manager to the selected Cloud server. The number of Fog nodes on the system depends on the complexity of the implemented scenario. Applications and services to be deployed on the Fog layer must also follow certain formats for compatibility with ProFog. On the runtime, IoT devices initially request services to the Cloud Manager that redirects them to the Fog node which provides the service, depicted in Figure 4. From there on out, the IoT device communicates directly with the service provider on the Fog. These services may perform data treatment and forward the data to the Cloud for analytics - or other processing intensive task - or provide time-critical functionalities to the IoT layer or external service consumers. This allows for reduced network congestion - which may reach critical points on IoT scenarios - and QoS adherence for time-critical services benefitting from Fog's reduced latency.

All management activities are centralized to a single point - the Cloud - facilitating system management and reducing landscape complexity. The Cloud Manager also monitors the load of local applications and collects information about the load of Fog applications from each Fog Manager, this data is then forwarded to the Elastic Manager which applies load prediction algorithms based on the collected data and provides proactive elasticity to both Cloud and Fog layers with no human intervention.

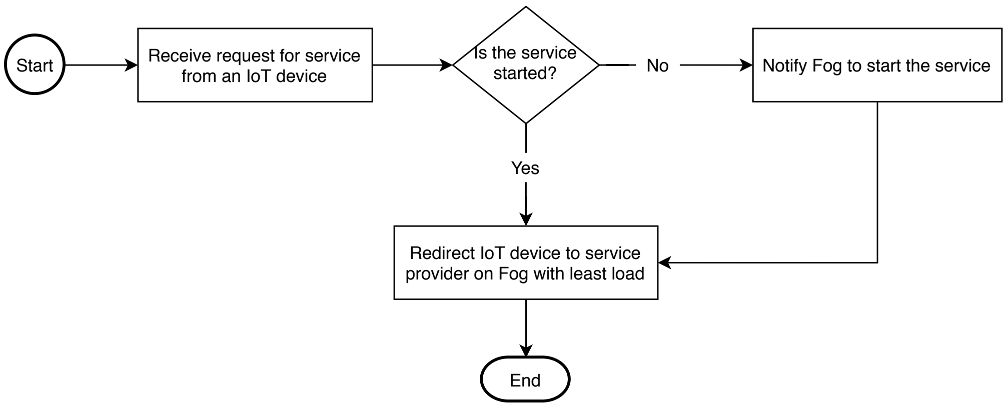

Start
Receive request for service from an IoT device
Is the service started?
No
Notify Fog to start the service
Yes
Redirect IoT device to service provider on Fog with least load
End

Figure 4 – Service request routing process performed by the Cloud Manager.

4.3 Application model for Fog deployment

Applications are exposed for deployment as Docker container images which include all the necessary services for its operations, in a similar way to what was proposed by Nguyen in his study (Nguyen et al., 2020). In order to start the application, ProFog creates an instance of the container image for that application, requiring the image to be built and configured to run the application upon initialization. Any necessary parameters for the application start-up can be provided during the instantiation of the container as long as the image was previously built with an ENTRYPOINT or CMD statement to support command line argument consumption through Docker. Deploying applications as isolated containers simplifies the deployment process while also providing more flexibility to the application development as it becomes possible to install any necessary resources for the application directly on its container.

As applications are deployed as containers, they are isolated from the Fog Manager, requiring them to implement internally any functionalities that are necessary for consuming data or providing services. For instance, an application that receives data from edge devices, performs operations on such data, and then forwards it to the Cloud must implement an HTTP server to receive data from edge devices through HTTP requests. The Fog Manager provides the application with the port number it should use to run its HTTP server upon initialization. The application must implement a shutdown service and a process to handle the rerouting of any connected clients back to the Cloud. The Fog Manager calls the shutdown service to notify the application of any eventual shutdown due to down-scaling, providing it the time left for the shutdown and the address of the Cloud Manager which may be used to reroute clients to another active application instance. All monitoring and elasticity control related tasks are managed by the Fog Manager and Cloud Manager, relieving application developers of any concerns of elasticity control at application level.

4.4 Elasticity Algorithm

ProFog uses proactive elasticity to provide stable QoS and prevent having the system on an overloaded state, which may compromise its operations. Proactive elasticity is not only capable of achieving more consistent and reliable scaling decisions with better rates of false-positive results in case of sudden peaks of load when compared to reactive elasticity, but it also allows to include relevant metrics such as deployment and start-up time into consideration for scaling decisions. This makes possible to have the necessary resources ready before they are required, preventing the application to go into an overloaded state.

Proactive elasticity depends on constant monitoring of the system's state, followed by the prediction of future load based on the collected data through the use of time-series forecasting algorithms. We selected the mathematical model ARIMA (AutoRegressive Integrated Moving Average) for this purpose as it is widely used for time series based prediction and it can provide predictions in fewer boot cycles than the machine learning models (da Rosa Righi et al., 2020). The term AutoRegression (AR) means that it is a regression of the variable against itself, thus the variable of interest is forecasted using a linear combination of its past values. Moving Average models use past forecast errors rather than past values of the variable. ARIMA takes both the previous values of the variable and the previous forecast errors into consideration for its predictions.

ARIMA has several configurations that are dependent on three parameters: *d* which is the minimum number of differencing that is required to make the time series stationary, *p* which is the order of autoregressive terms (AR) and *q* which is the order of the moving average (MA) term. We have used Auto Arima, a function from the time series analysis library used, to determine the most appropriate values for these parameters. As ARIMA's predictions depend on the analysis of time series data, it is necessary to collect a given number of load observations to start the load predictions. Righi's study (da Rosa Righi et al., 2020) suggests the use of at least six cycles - or six monitoring observations - for the calculation of the predictions. We have chosen to use the data of ten cycles for our predictions, meaning ARIMA will take *10 x monitoring_observation_period* to initialize and provide its first prediction. For example, if there is an interval of 10 seconds between each observation of the load of the system, it will take 100 seconds to collect enough values to make the first prediction.

We have also defined a *ahead* parameter based on the equation 1 - which was proposed by Righi on his article (da Rosa Righi et al., 2020) - to determine how many cycles ahead into time ARIMA should make its prediction to allow the deployment of new resources in time. A proper definition of the *ahead* parameter is of crucial importance to avoid two possible pitfalls: 1. Too high of a value may cause the prediction to miss short-term changes on the application behavior incurring in false-negative or false-positive elasticity; 2. If the value is too low, the elasticity action may deliver the resources after they start to be necessary, causing the application to run on an overloaded state for some time, affecting its operations (da Rosa Righi et al., 2020). Figure

6 depicts the flow of actions performed by ProFog for elasticity management. Like reactive models, proactive elasticity models also require the definition of upper and lower threshold settings. After performing load prediction, the predicted values are compared to the threshold values to determine if scaling actions are required. Figure 5 illustrates the behavior of proactive elasticity models during execution.

$$ahead = \frac{Abs(Max(scaling\_out\_time))}{monitoring\_observation\_period} \tag{1}$$

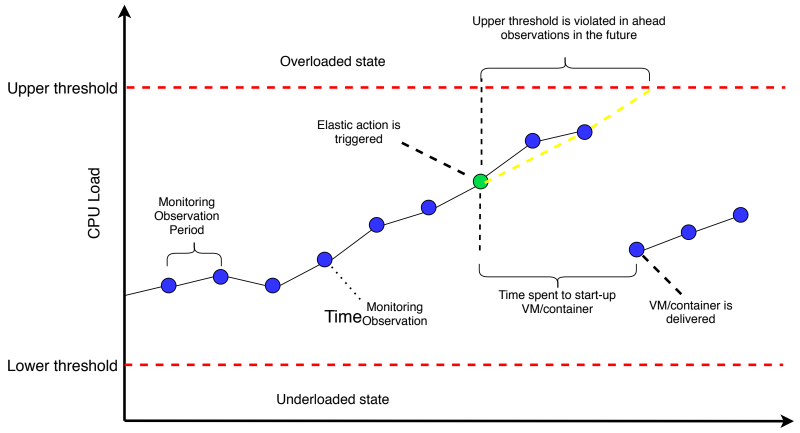

Upper threshold is violated in ahead observations in the future
Overloaded state
Upper threshold
Elastic action is triggered
CPU Load
Monitoring Observation Period
Monitoring Observation
Time
Time spent to start-up VM/container
VM/container is delivered
Lower threshold
Underloaded state

Figure 5 – Server elasticity managed by a proactive elasticity model.

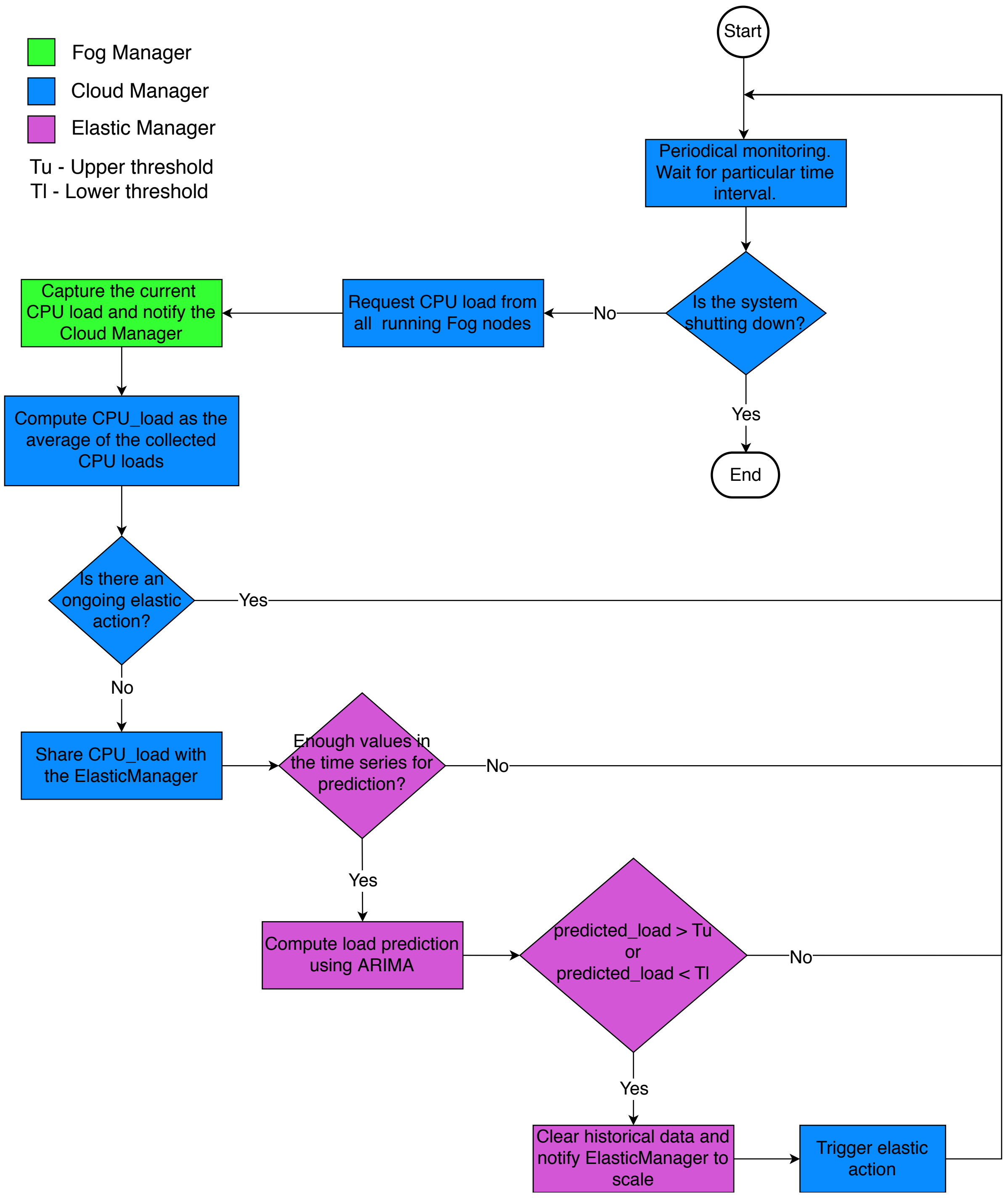

🟩 Fog Manager
🟦 Cloud Manager
🟪 Elastic Manager
Tu - Upper threshold
Tl - Lower threshold
Start
Periodical monitoring. Wait for particular time interval.
Is the system shutting down?
No
Yes
End
Request CPU load from all running Fog nodes
Capture the current CPU load and notify the Cloud Manager
Compute CPU_load as the average of the collected CPU loads
Is there an ongoing elastic action?
Yes
No
Share CPU_load with the ElasticManager
Enough values in the time series for prediction?
No
Yes
Compute load prediction using ARIMA
predicted_load > Tu or predicted_load < Tl
No
Yes
Clear historical data and notify ElasticManager to scale
Trigger elastic action

Figure 6 – ProFog elasticity management flowchart.

5 EVALUATION METHODOLOGY

The following sections describe the use case that we selected to evaluate the model, the infrastructure and workload used for the tests, and metrics used for their evaluation.

5.1 Application

The domain of Internet-of-Things is composed of a variety of scenarios, each having its own set of demands and peculiarities. The proposed solution is designed to address the demand for low-latency and continuous Quality-of-Service which is present in a series of solutions and industries, such as Smart Manufacturing, Smart Cities, and healthcare. Therefore, in order to properly evaluate the results of the proposed architecture, a scenario presenting such a set of demands is required to be used as a case. After analyzing a series of possible IoT scenarios that were documented by the OpenFog Consortium, an association for the advance of Fog Computing as a connected and interoperable architecture, we decided to make use of a video streaming use case for the prototype's test. Cases such as healthcare and Industry 4.0 require very specific knowledge of the industry to recreate scenarios properly as the variables, data, and the analysis of such data are very specific to their environment. However, streaming scenarios are far more versatile as they are by themselves more connected to the information technology area and do not involve as many unfamiliar end devices and sensors.

Streaming of sports' events has become popular as even those attending the event itself are simultaneously watching it on their smartphones to have access to different angles of view. This generates a demand for higher availability on streaming services and lower latency to maintain the feeling of real-time to the clients. In such scenarios, Fog nodes may be used to execute pre-processing and encoding of video as well as its distribution to nearby clients with low latency and constant QoS. Not only maintaining the feeling of real-time but also reducing network congestion and distributing the load on more service providers.

There are no applications or benchmarks designed to evaluate the performance of such scenarios, so we have modeled an application to allow us to evaluate the prototype. The official version of the streaming scenario proposed by the Industrial Internet Consortium is composed of multiple different Fog levels, each responsible for a specific task in order to achieve the highest level of performance for live-event streaming. As this project's solution is not specific for streaming, but for a general Cloud-Fog architecture, the streaming scenario was simplified to a single Fog level which provides Video-On-Demand (VOD) to clients while monitoring its state and performing scaling operations proactively.

5.2 Infrastructure

In order to evaluate the results of ProFog, we have deployed our solution to a Cloud-Fog environment. For the Cloud layer, we have used an Azure Container instance, configured with 2 CPUs and 4GB of RAM, to host the Cloud Manager and Elastic Manager. The Fog layer was composed of three Raspberry PI 4 Model B microcomputers as Fog nodes, hosting application services.

On the Industrial Internet Consortium's proposition of the video broadcasting scenario, which we have based our case on, the IoT layer is composed of HD video cameras that collect data for transmission. As our goal is to validate the elastic capacity of ProFog and not the streaming scenario, we have removed the HD video cameras in exchange for pre-recorded video files (VOD), allowing us to simplify the application and test infrastructure while maintaining the elastic behavior. By doing that, we have removed the IoT layer from the test infrastructure. In our test scenario, the services provided by the Fog layer are consumed by Video-On-Demand clients which are generated by JMeter on a separate machine, creating service load for the Fog and scaling needs. The machine we used to emulate the clients is configured with an Intel I7 7th Generation processor and 16GB of RAM. Figure 7 shows a diagram of the test infrastructure.

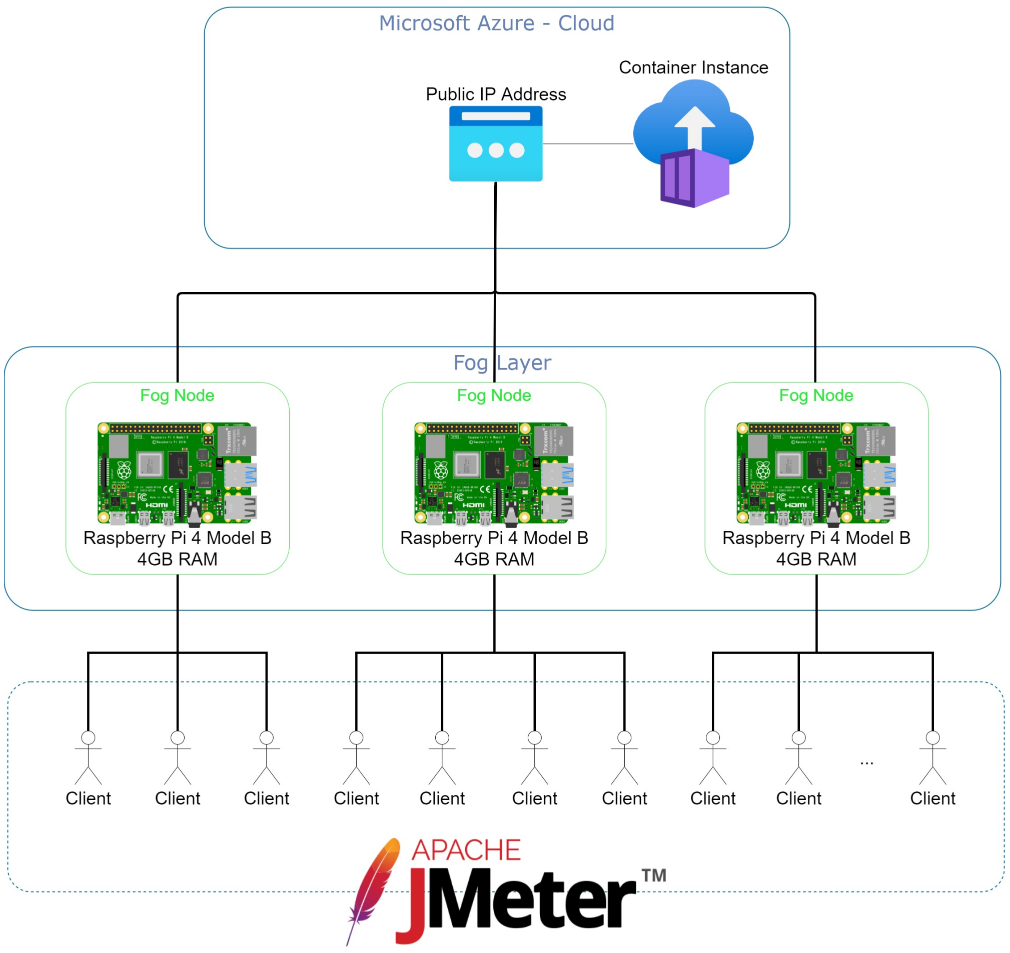


Figure 7 – Infrastructure used for the prototype evaluation.

We initially aimed to connect the machine that emulates the clients to the network over WiFi to more accurately recreate a real scenario. However, we ruled out that option upon testing as any relatively high load of clients affected the download times over WiFi, which could be caused by a machine limitation or a router limitation. All the devices had to be connected to the network via cable in order to emulate the necessary load to evaluate the scenario.

5.3 Prototype

The Cloud Manager and the services on the Fog layers were built on top of Node.js as it provides native asynchronous request handling and a high performance and lightweight backend, which contributes to the overall performance of the system. This is especially important for the Fog nodes as they are limited in hardware capacity. The Elastic Manager was created with Python and it uses the pmdarima library for time series analysis. Pmdarima delivers Python data science capabilities to substitute the use of R code. Using Python instead of R allows us more for communication options between the Cloud Manager and Elastic Manager as Python is an all-purpose high-level programming language while R is focused on data science.

The Fog nodes provide the streaming service through an HTTP server running on Node.js which streams video content to the connected clients using the HTTP Live Streaming (HLS) protocol. This protocol was developed by Apple in 2009 to address the difficulties in accessing streaming services from different resolution devices and constrained bandwidth connections. HLS consists of first chopping up the video content into small encoded chunks (MPEG2 Transport Stream) that are later on stored and provided to clients by an HTTP server. During the encoding of the video into transport streams, a playlist file (M3U8) is also created. The playlist is interpreted on the client-side and works as an index to determine the existing video chunks and the resolution/bandwidth configuration they are meant to. Due to its adaptive bitrate capacity, native support on Apple devices, and support on HTML5 Video Players, HLS has become not only Apple's standard streaming technology but also one of the most popular technologies for streaming lately (Jain; Shrivastava; Moghe, 2020).

For our prototype, we have used FFMpeg to encode a nine-minute long video file and stored the generated index file and video stream chunks locally for consumption, refer to figure 8 for the preparation of the video files. Streaming clients initially connect to the Cloud Manager which then redirects them to an active Fog node which provides the requested service, seamlessly to the client - refer to figure 9. Upon receiving a service request, the Cloud Manager performs load balancing between different active Fog nodes, always directing clients to the node with the least load. The prototype does not contain any application on the Cloud side for the evaluation of the Cloud elasticity.

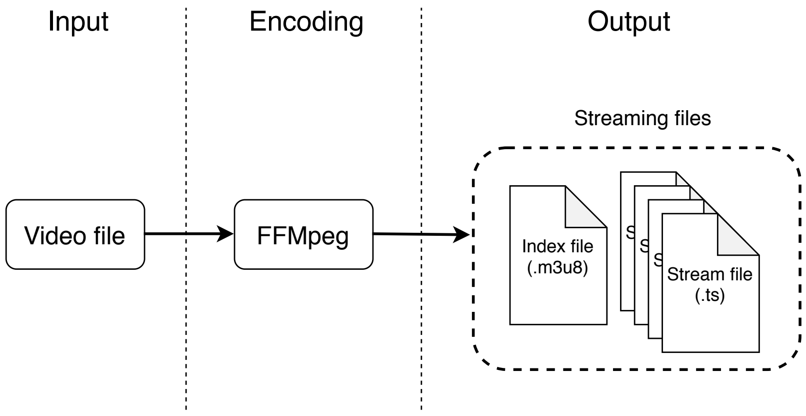
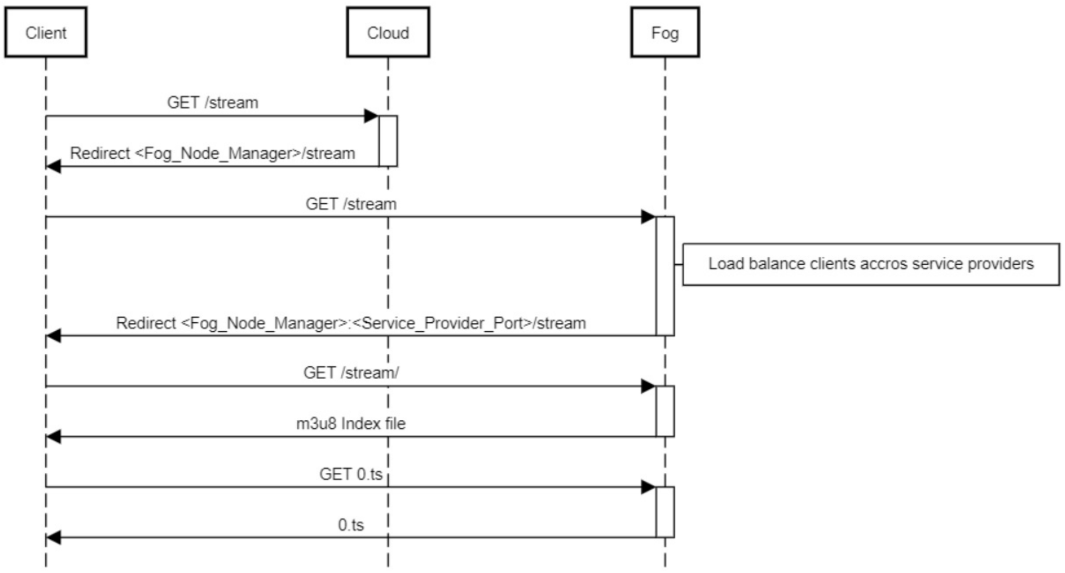


Figure 8 – Video encoding for HLS streaming.



Client
Cloud
Fog
GET /stream
Redirect <Fog_Node_Manager>/stream
GET /stream
Load balance clients accros service providers
Redirect <Fog_Node_Manager>:<Service_Provider_Port>/stream
GET /stream/
m3u8 Index file
GET 0.ts
0.ts

Figure 9 – Sequence diagram of the flow of requests prior to starting the streaming service.

5.4 Workload and scenarios

In order to test the solution in the selected scenario, it is necessary to create and manage hundreds of simultaneous streaming clients. We have used JMeter, a load testing tool maintained by the Apache Foundation, for this purpose as it is a stable load testing tool with an active community that has the capacity to handle massive amounts of threads. We have used different JMeter elements to emulate the required streaming logic, allowing the redirect of streaming clients to different servers during the streaming process, along with a few custom Groovy scripts which perform validations of the request responses and emulate video buffer control.

During the test execution, each client generated by JMeter makes HTTP requests to a service provider on the Fog layer for transport stream files - video chunks. Replying back to the HTTP requests requires the service provider to execute the TCP algorithm, resulting in CPU load and network connections which generate resizing needs as the number of clients increases. We have configured JMeter to emulate clients at two different points in time. The first client emulation created 400 clients over 180 seconds. The second client emulation created 200 more clients over 100 seconds. The first load was triggered at the start of the test and the second load at 250 seconds into the test.

In order to evaluate the prototype, we have configured ProFog to accept two different execution modes which allow us to observe two different scenarios in the same environment:

- Scenario 1: Streaming service with ProFog running on reactive elasticity mode;
- Scenario 2: Streaming service with ProFog running on proactive elasticity mode.

5.5 Evaluation Metrics

We observed the system's average load in order to determine the efficiency of ProFog on handling resizing upon demand. Average load is a metric that considers CPU load, network connections, and I/O operations, being a more reliable metric for certain applications that do not rely exclusively on CPU intensive operations. This metric is a calculated numerical value based on multiple factors that is present on Unix-based operational systems, requiring a deeper understanding of the system's purpose and use for its evaluation than the CPU usage.

One of the benefits achieved by the use of proactive elasticity is a more efficient management of resources, so we are also observing energy consumption as a metric to evaluate this behavior. We are able to estimate the energy consumption by analyzing the deployment times of containers and for how long they were active (da Rosa Righi et al., 2020), as shown in equation 2. On equation 2, n is the maximum number of containers and T(i) is the time spent running with i containers. For example, suppose the following scenario: 1 container for 40 seconds, 2 containers for 25 seconds, 3 containers for 50 seconds; this would result in energy = 1 x 40 + 2 x 25 + 3 x 50 = 240.

$$Energy = \sum_{i=1}^{n}(i\,x\,T(i)) \tag{2}$$

6 RESULTS

This section presents the results that were observed when running the prototype. We first present the graphs of the results and review the elastic decisions for both evaluated scenarios, proactive and reactive, followed by a review of the energy consumption for each scenario.

6.1 Load and resource allocation

We have configured ProFog to collect data from each of the Fog nodes every 5 seconds. Calculating the prediction on the Elastic Manager takes up to 4 seconds. With these metrics, the total interval data for each collection is around 9 seconds. The download of the container image from Dockerhub takes up to 20 seconds, depending on network congestion, and the initialization of the application using the container image less than 2 seconds, adding up to close to 22 seconds. Based on the equation 1, predictions should be performed for 2.44 measures ahead, resulting from 22 divided by 9. We have decided to round this value up to 3 to compensate for unaccounted communication delays.

Figure 10 illustrates the system behavior using the reactive elasticity approach over the course of a twelve-minute load test. Analyzing the graph, we can see that a load peak at 1:45min has triggered the start-up of the second Fog node. After that, the same situation repeats itself at 2:35min. At 2:40min, three Fog nodes were already active and remained active until the load dropped below the lower threshold at 10:25min. The upper and lower threshold values for elasticity control were set to 0.9 and 0.5, respectively, based on the observation of the system's behavior over multiple test runs.

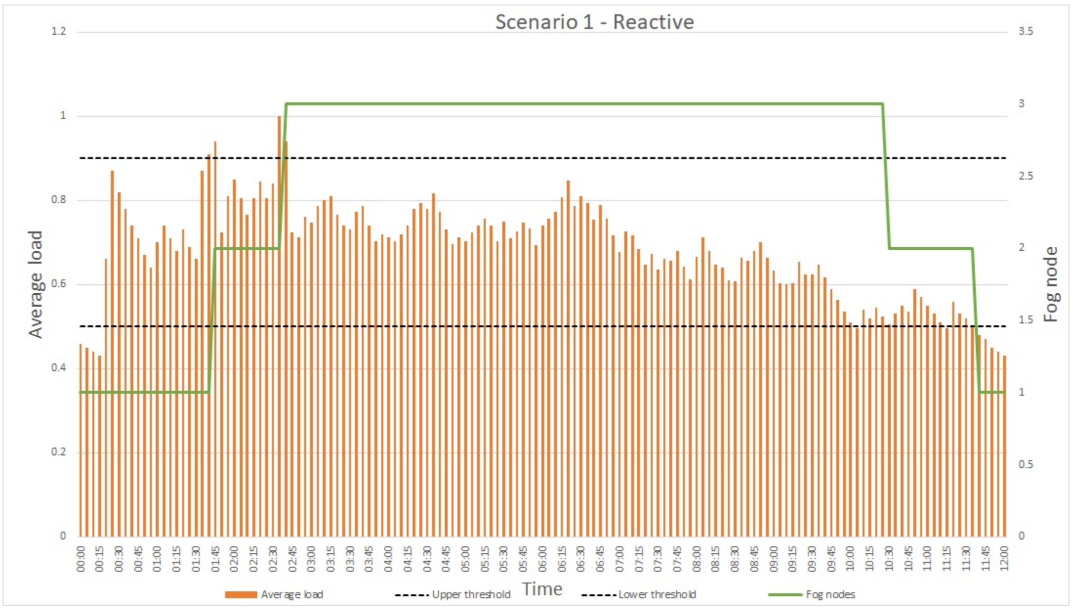


Figure 10 – Scenario 1: Streaming service with ProFog running on reactive elasticity mode.

By analyzing figure 11 - which illustrates the system behavior with ProFog running on proactive mode, we can see that the start-up of the second Fog node was delayed as the predicted values were lower than the observed load value, which was a peak. In this scenario, the start-up of the second Fog node took place at 2:05min and of the third node at 4:25min. We can also see that the predicted values are very close to the observed load values, this behavior may be related to the parameterization used for ARIMA or to the fact that the predictions are only taking place three measures ahead, which is quite low. Increasing the ahead value could result in improvements in the predictions, but it could also incur in the allocation of a resource for longer than it is necessary. The drops to zero in the predicted values mean that the Elastic Manager has requested an elastic action to be taken and cleared its buffer of load values, this action could be followed or ignored by the Cloud Manager depending on resource availability at the moment.

We have opted not to calculate a standard deviation or mean at this point as the application used for the test is of dynamic nature and subject to multiple variables of the emulation environment which would affect the results. Furthermore, the most important point for us to monitor at this moment is the elastic behavior of the system and its impact on resource allocation. As the current prototype does not contain any applications which run on the Cloud, we were not able to verify the elastic behavior on the Cloud side either.

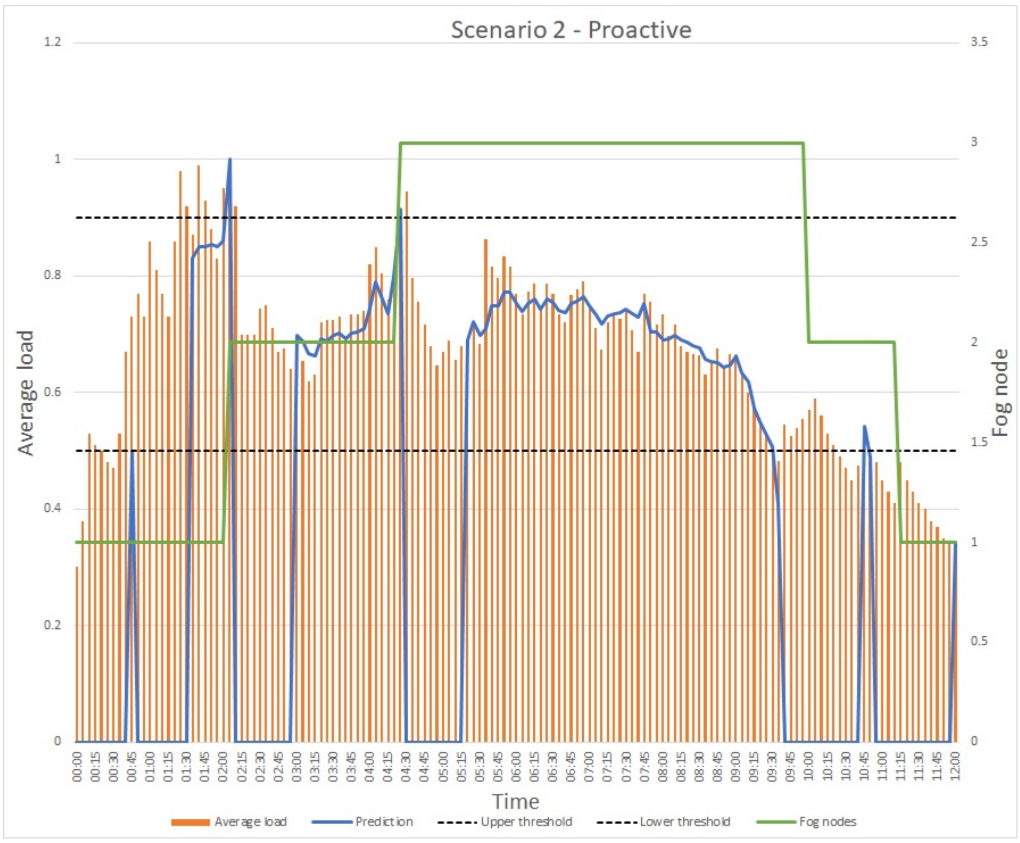


Figure 11 – Scenario 2: Streaming service with ProFog running on proactive elasticity mode.

6.2 Energy consumption

We have mapped the Fog node initialization times for each of the scenarios to calculate an estimate of energy consumption. Table 2 presents the results for scenario 1 and table 3 the results for scenario 2. By comparing the total values on the tables, we see that scenario 2 has presented lower energy consumption. Even though the predictions were not far enough ahead to trigger scaling actions in advance on the tested scenario, we can see that the use of time series analysis has helped to smooth the data and minimize unnecessary scaling on sudden peaks. This has improved energy consumption by 11.21%, resulting from (1785/1605 - 1).

Energy consumption		
Number of Fog Nodes	Period of time (s)	Energy consumption
1	125	125
2	125	250
3	470	1410
	Total	1785

Table 2 – Energy consumption estimate based on equation 2 for the scenario 1.

Energy consumption		
Number of Fog Nodes	Period of time (s)	Energy consumption
1	170	170
2	215	430
3	335	1005
	Total	1605

Table 3 – Energy consumption estimate based on equation 2 for the scenario 2.

6.3 Discussion

The proximity between the actual load values and the predicted load values shows us that the use of proactive elasticity for fast deployment environments, such as the ones based on containers, may be challenging. The lower the time required for deployment is, the more difficult it is to make a prediction that is capable of scaling the system in time. By modifying the ahead parameter used for the predictions, we could further anticipate scaling needs, however, at the same time, we may deliver resources before they are required, which may incur unnecessary energy consumption and additional costs.

Even though the predictions performed by ProFog were too close to the actual load values for us to see preemptive scaling actions take place, we can see that the use of time series analysis has brought other benefits to the system. By analyzing the system load over time for scenario 2, we see that the load predictions have smoothed load peaks, helping to avoid unnecessary deployment of new Fog nodes - as we can see that occurred in scenario 1. By preventing unnecessary deployment of Fog nodes caused by load peaks, ProFog has reduced the number of active machines for a period of time, consequently reducing energy consumption and operating costs for the system.

We believe the research on proactive elasticity for Fog Computing is of great importance for the advance of IoT systems. Therefore, we may, in the future, perform further research on an accurate way to determine how far ahead in time predictions should be made for proactive elasticity applied to Fog Computing implementations. It would also be interesting to include additional metrics for the predictions, such as network traffic, that may allow us to have a more reliable view of the system as a whole and further support the heterogeneity of IoT systems that exists.

7 CONCLUSION

As the use of IoT grows across multiple industries, it becomes necessary to review how IoT systems are designed. Cloud data centers are often used for the implementation of IoT scenarios as they offer scalability and reliability, but such configuration poses many challenges - from security to QoS and network congestion - that may prevent certain use cases or the adoption in certain industries. These challenges have fostered the advance of another system architecture named Fog computing. This architecture brings the processing of data closer to the resource, allowing better response times, lower latency, and increased security.

Considering the aforementioned scenario, this article addressed proactive elasticity for resource allocation on Fog computing for IoT implementations by presenting a model named ProFog. This model manages resource allocation and provides proactive elasticity to applications without any user intervention or elasticity control logic from the application end. In order to validate the model, we have built a prototype using Microsoft Azure - as the Cloud - and three Raspberry Pi 4 microcomputers - which operated as Fog nodes. We evaluated our prototype using a Video-On-Demand streaming scenario. However, ProFog may be applied to a wide range of scenarios, such as manufacturing, healthcare and Smart Cities.

The prototype validation showed positive results when analyzing energy consumption - presenting an improvement of 11.21%. However, the load predictions made by ProFog were not capable of scaling the system preemptively. This result may be associated with a number of different reasons, such as the technologies used for the prototype - containers for instance which create an extremely fast deployment environment - our selected use case, our application, or the parameterization of ARIMA. Therefore, we believe that the observed results do not make ProFog as a model an unviable solution, but open way for further research on the subject and additional improvements. From a contribution perspective, ProFog offers seamless proactive elasticity control - through the use of time series analysis - for applications and services which operate on top of Fog computing systems.

Future research includes further analysis of different prediction methodologies that may be more suitable for fast deployment environments, the monitoring and analysis of different metrics for individual applications, and the configuration of application operation requisites to allow the deployment of applications to Fog nodes that match certain criteria.

References

Abbasi, A. A. et al. Software-defined cloud computing: A systematic review on latest trends and developments. **IEEE Access**, v. 7, p. 93294–93314, 2019.

Agarwal, S.; Yadav, S.; Yadav, A. K. An architecture for elastic resource allocation in Fog Computing. **IJCSC**, v. 6, n. 2, p. 201–207, 2015.

AL-KHAFAJIY, M. et al. Improving fog computing performance via fog-2-fog collaboration. **Future Generation Computer Systems**, v. 100, p. 266 – 280, 2019. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X18331868>.

ALSAFFAR, A. A. An Architecture of IoT Service Delegation and Resource Allocation Based on Collaboration between Fog and Cloud Computing. 2016.

BONOMI, F. et al. Fog computing and its role in the internet of things. In: **Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing**. New York, NY, USA: Association for Computing Machinery, 2012. (MCC '12), p. 13–16. ISBN 9781450315197. Disponível em: <https://doi.org/10.1145/2342509.2342513>.

Chen, Y. et al. Cloud-fog computing for information-centric internet-of-things applications. In: **2017 International Conference on Applied System Innovation (ICASI)**. [S.l.: s.n.], 2017. p. 637–640.

CISCO. **Cisco Annual Internet Report (2018–2023) White Paper**. 2020. Disponível em: <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>.

da Rosa Righi, R. et al. Enhancing performance of iot applications with load prediction and cloud elasticity. **Future Generation Computer Systems**, v. 109, p. 689 – 701, 2020. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X17329229>.

FISCHER, G. S. et al. Elhealth: Using internet of things and data prediction for elastic management of human resources in smart hospitals. **Engineering Applications of Artificial Intelligence**, v. 87, p. 103285, 2020. ISSN 0952-1976. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0952197619302465>.

HE, H. et al. The security challenges in the iot enabled cyber-physical systems and opportunities for evolutionary computing & other computational intelligence. In: **2016 IEEE Congress on Evolutionary Computation (IEEE CEC)**. [s.n.], 2016. p. 1015–1021. Disponível em: <http://eprints.bournemouth.ac.uk/24677/>.

He, J. et al. Multitier fog computing with large-scale iot data analytics for smart cities. **IEEE Internet of Things Journal**, v. 5, n. 2, p. 677–686, 2018.

Hu, P. et al. Software-defined edge computing (sdec): Principle, open iot system architecture, applications and challenges. **IEEE Internet of Things Journal**, p. 1–1, 2019.

Jain, N.; Shrivastava, H.; Moghe, A. A. Production-ready environment for hls player using ffmpeg with automation on s3 bucket using ansible. In: **2nd International Conference on Data, Engineering and Applications (IDEA)**. [S.l.: s.n.], 2020. p. 1–4.

MAHMUD, R.; KOCH, F. L.; BUYYA, R. Cloud-fog interoperability in iot-enabled healthcare solutions. In: **Proceedings of the 19th International Conference on Distributed Computing and Networking**. New York, NY, USA: Association for Computing Machinery, 2018. (ICDCN '18). ISBN 9781450363723. Disponível em: <https://doi.org/10.1145/3154273.3154347>.

MASOOD, T.; SONNTAG, P. Industry 4.0: Adoption challenges and benefits for smes. **Computers in Industry**, v. 121, p. 103261, 2020. ISSN 0166-3615. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0166361520304954>.

MOHAMED, N. SmartCityWare: A Service-Oriented Middleware for Cloud and Fog Enabled Smart City Services. **NEW ERA OF SMART CITIES: SENSORS COMMUNICATION TECHNOLOGIES AND APPLICATIONS**, 2017.

Mourtzis, D.; Vlachou, E.; Milas, N. Industrial big data as a result of iot adoption in manufacturing. **Procedia CIRP 55**, p. 290–295, 2016.

NAHA, R. K. et al. Deadline-based dynamic resource allocation and provisioning algorithms in fog-cloud environment. **Future Generation Computer Systems**, v. 104, p. 131 – 141, 2020. ISSN 0167-739X. Disponível em: <http://www.sciencedirect.com/science/article/pii/S0167739X19319016>.

Nguyen, N. D. et al. Elasticfog: Elastic resource provisioning in container-based fog computing. **IEEE Access**, v. 8, p. 183879–183890, 2020.

Righi, R. d. R. et al. A survey on global management view: Toward combining system monitoring, resource management, and load prediction. **J Grid Computing**, v. 17, p. 473–502, 2019.

Small, N. et al. Niflheim: An end-to-end middleware for applications on a multi-tier iot infrastructure. In: **2017 IEEE 16th International Symposium on Network Computing and Applications (NCA)**. [S.l.: s.n.], 2017. p. 1–8.

STANKOVSKI, V. et al. Implementing time-critical functionalities with a distributed adaptive container architecture. In: **Proceedings of the 18th International Conference on Information Integration and Web-Based Applications and Services**. New York, NY, USA: Association for Computing Machinery, 2016. (iiWAS ’16), p. 453–457. ISBN 9781450348072. Disponível em: <https://doi.org/10.1145/3011141.3011202>.

SUGANUMA, T. Multiagent-Based Flexible Edge Computing Architecture for IoT. **IEEE Network**, 2018.

Sun, X.; Ansari, N. Edgeiot: Mobile edge computing for the internet of things. **IEEE Communications Magazine**, v. 54, n. 12, p. 22–29, 2016.

VAQUERO, L. M.; RODERO-MERINO, L. Finding your way in the fog: Towards a comprehensive definition of fog computing. **SIGCOMM Comput. Commun. Rev.**, Association for Computing Machinery, New York, NY, USA, v. 44, n. 5, p. 27–32, out. 2014. ISSN 0146-4833. Disponível em: <https://doi.org/10.1145/2677046.2677052>.

Yin, L.; Luo, J.; Luo, H. Tasks scheduling and resource allocation in fog computing based on containers for smart manufacturing. **IEEE Transactions on Industrial Informatics**, v. 14, n. 10, p. 4712–4721, 2018.