NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATION

MSc THESIS

An Intelligent scheme for Outliers' detection on a Cloudlet

Dimitrios S. Milios

Supervisors: **Efstathios Hadjiefthymiades, Associate Professor NKUA**
Kostas Kolomvatsos, Phd, Researcher NKUA

ATHENS

JULY 2018

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ένα έξυπνο σχήμα για ανίχνευση ακραίων τιμών σε ένα Cloudlet

Δημήτριος Σ. Μήλιος

Επιβλέποντες: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής ΕΚΠΑ
Κώστας Κολομβάτσος, Διδάκτωρ, Ερευνητής ΕΚΠΑ

ΑΘΗΝΑ

ΙΟΥΛΙΟΣ 2018

MSc THESIS

An intelligent scheme for Outliers' detection on a Cloudlet

Dimitrios S. Milios
S.N.: M1384

SUPERVISORS: **Efstathios Hadjiefthymiades,** Associate Professor NKUA
Kostas Kolomvatsos, Phd , Researcher NKUA

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ένα έξυπνο σχήμα για ανίχνευση ακραίων τιμών σε ένα Cloudlet

Δημήτριος Σ.Μήλιος
Α.Μ.: Μ1384

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Ευστάθιος Χατζηευθυμιάδης,** Αναπληρωτής Καθηγητής ΕΚΠΑ
Κώστας Κολομβάτσος, Διδάκτωρ, Ερευνητής ΕΚΠΑ

ABSTRACT

A Cloudlet is a computer or a cluster of computers connected at the edge of the network to provide low-latency access to Computing resources for IoT devices. The main aim of this Thesis is to provide an intelligent scheme for detection of Outliers in a Cloudlet simulation environment. For this purpose, Hilout algorithm is used, modified to use, in addition, a temporal approach. The experiments that took place focus on different configurations' values of the input data, data-vectors with multiple dimensions coming from IoT devices. The results are examined by the scope of how the different configurations affect the number of Outliers that are detected by the scheme. The environment that is used, is provided by the CloudSim framework.

SUBJECT AREA: Cloud, Cloudlet, IoT, Cloud Computing, Mobile Cloud Computing

KEYWORDS: IoT, Cloud, Cloudlet, Hilout, Exponential, Gaussian, Variance, PCA, Outliers

ΠΕΡΙΛΗΨΗ

Ένα Cloudlet είναι ένας υπολογιστής ή ένα σύμπλεγμα υπολογιστών που συνδέονται στην άκρη του δικτύου για να παρέχει πρόσβαση χαμηλής καθυστέρησης σε υπολογιστικούς πόρους για συσκευές IoT. Ο κύριος στόχος της παρούσας εργασίας είναι να παράσχει ένα έξυπνο σχέδιο για την ανίχνευση ακραίων τιμών σε περιβάλλον προσομοίωσης Cloudlet. Για το σκοπό αυτό, χρησιμοποιείται ο αλγόριθμος Hilout, τροποποιημένος για να χρησιμοποιήσει επιπρόσθετα μια χρονική προσέγγιση. Τα πειράματα που πραγματοποιήθηκαν επικεντρώνονται στις διαφορετικές τιμές των συνθέσεων των δεδομένων εισόδου, των φορέων δεδομένων με πολλαπλές διαστάσεις που προέρχονται από συσκευές IoT. Τα αποτελέσματα εξετάζονται από το πεδίο εφαρμογής του τρόπου με τον οποίο οι διαφορετικές διαμορφώσεις επηρεάζουν τον αριθμό των ακραίων τιμών που ανιχνεύονται από το σχήμα. Το περιβάλλον που χρησιμοποιείται, παρέχεται από το framework CloudSim.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Cloud, Cloudlet, IoT, Cloud Computing, Mobile Cloud Computing

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: IoT, Cloud, Cloudlet, Hilout, Εκθετική, Gaussian, Διακύμανση, PCA, Ακραίες Τιμές

Αφιερώνω αυτή την εργασία σε όλα τα φιλικά και συγγενικά άτομα που με στήριξαν και με στηρίζουν σε όλη την φοιτητική μου σταδιοδρομία.

ΕΥΧΑΡΙΣΤΙΕΣ

Ευχαριστώ από τα βάθη της καρδιάς μου τον κ.Κολομβάτσο για την στήριξη και την υπομονή που υπέδειξε προς το πρόσωπο μου καθ’όλη την διαδικασία εκπόνησης της εργασίας.

CONTENTS

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF TABLES

PREFACE

This master Thesis was carried out in the postgraduate studies program of "Computer Systems Technology" of the Department of Informatics and Telecommunications of the National and Kapodistrian University of Athens. The goal of my research is to study the CloudSim framework and the concept of a Cloudlet in order to provide an intelligent scheme for detection of Outliers on a Cloudlet simulation environment with the use of the Hilout algorithm with a temporal approach.

I would like at this point to express my warmest thanks to my Thesis supervisor, Assistant Prof. Efstathios Hadjiefthymiades, for guidance and valuable contributions during my study of the master Thesis. I would also like to thank Kostas Kolomvatsos, Ph.D. and Researcher of Department of Informatics and Telecommunications, for his helpful guidance and advice throughout the preparation of this work. Their continuous monitoring of the progress of the master Thesis, their meaningful remarks and their help for resolving any problem have contributed in the final formation of this Thesis.

1. INTRODUCTION

Our days are dominated by the rapidly evolution and progress of technology, and this has affected our everyday life. New devices such as smartphones and tablets have entered in users' lives by helping them to improve their lives, to monitor remotely their home devices, to inform about the traffic or the pollution of a city and a lot of other applications. The next think was to connect all these devices, in order to exchange data and operate more automated, without requiring human-to-human or human-to-computer interaction. This is called Internet of Things (IoT).

The IoT devices can be connected to a Cloud. A Cloud or Cloud Computing is a framework for sharing resources, information and software capabilities to different mobile/IoT devices. The resources will be available on the Cloud and can be shared by the devices on demand. It is actually a model for enabling convenient, on-demand network access to Computing resources that can be rapidly provisioned and released with minimal management effort. A very similar concept that enhances the "bond" between the client and the Cloud is the concept of Mobile Cloud Computing. Mobile Cloud Computing at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile Cloud applications move the Computing power and data storage away from mobile phones and into the Cloud, bringing applications and MCC to not just smartphone users but a much broader range of mobile subscribers.

Nowadays, the concept of Cloudlet appeared. In the Cloudlet concept, mobile device offloads its workload to a resource-rich, local Cloudlet. Cloudlets would be situated in common areas such as coffee shops, libraries or university halls, so that mobile devices can connect and function as a thin client to the Cloudlet. A Cloudlet could be any first hop element at the edge of network.

In this Thesis we use an intelligent scheme for detection of Outliers on a Cloudlet simulation environment with the use of Hilout Algorithm, a bit modified to support a temporal approach. For this purpose, we use the CloudSim simulation framework in order to run our experiments and to make our conclusions regarding the effectiveness of the scheme.

The Thesis is organized as follows: In Chapter 2 we give the definition of the Cloud Computing and of the Mobile Cloud Computing and we explain their architectures and their features. In Chapter 3 we give the definition of a Cloudlet and we explain its architecture and we mention few information regarding the Cooperative Caching concept. In Chapter 4 we explain our Hilout approach, we give some information regarding PCA dimension reduction, which we are using in some of our experiments and we give some information regarding the Variance of dataset, which we also use in our experiments. Last but not least for this chapter, we explain why we use this approach. In Chapter 5 we give few information regarding the CloudSim framework but most important we give the results of the experiments we had. In the last Chapter, we give some conclusions regarding the work we did.

2. Mobile Cloud Computing

In this chapter we are going to give the definition of Cloud Computing, Mobile Cloud Computing, their architectures, advantages-disadvantages and the areas of usage.

2.1 Cloud Computing Definition

Cloud Computing is a framework for sharing resources, information and software capabilities to different mobile devices. The resources will be available on the Cloud and can be shared by the devices on demand. It is actually a model for enabling convenient, on-demand network access to Computing resources that can be rapidly provisioned and released with minimal management effort. [11]

Cloud Computing is described also as a range of services which are provided by an Internet-based cluster system. Such cluster systems consist of a group of low-cost servers or Personal Computers (PCs), organizing the various resources of the computers according to a certain management strategy, and offering safe, reliable, fast, convenient and transparent services such as data storage, accessing and Computing to clients. [1]

The concept behind Cloud Computing is to offload computation to remote resource providers. [13] The main objective behind the Cloud Computing is the delivery of different services, software and processing capacity over the Internet, increasing storage, reducing cost, automating systems and decoupling of service delivery from underlying technology, and providing flexibility and mobility of information in different purposes. [1]

2.2 Cloud Computing Classification

Cloud Computing can be viewed as a collection of services, which can be represented as a layered Cloud Computing architecture. In the upper layers of this paradigm, Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) are stacked:

- **Data centers layer:** This layer provides the hardware facility and infrastructure for Clouds. In data center layer, a number of servers are linked with high-speed networks to provide services for customers. Typically, data centers are built in less populated places, with a high power supply stability and a low risk of disaster.
- **Infrastructure as a Service (IaaS):** IaaS is built on top of the data center layer. It is the delivery of computer infrastructure (typically a platform virtualization environment) as a service. IaaS enables the provision of storage, hardware, servers and networking components. The capability provided to the end users is to provision processing, storage, networks, and other fundamental Computing resources where the end user is able to deploy and run arbitrary software, which can include operating systems and applications. The user does not manage or control the underlying Cloud infrastructure but it has control over operating systems, storage, deployed applications, and possibly limited control of select networking components. The end-user typically pays on a per-use basis. Thus, end-user can save cost as the payment is only based on how much resource they really use. Infrastructure can be expanded or shrunk dynamically as needed. The examples of IaaS are Amazon EC2 (Elastic Cloud Computing) and S3 (Simple Storage Service).

- **Platform as a Service (PaaS):** It is the delivery of Computing platform and solution stack as a service. The capability provided to the end users is to deploy onto the Cloud infrastructure user created or acquired applications created using programming languages and tools supported by the provider. PaaS offers an advanced integrated environment for building, testing and deploying custom applications. PaaS providers offer a predefined combination of OS and application servers, such as WAMP platform (Windows, Apache, MySQL and PHP), LAMP platform (Linux, Apache, MySQL and PHP), and XAMP (X-cross platform) limited to J2EE, and Ruby etc. The examples of PaaS are Google App Engine, Microsoft Azure, and Amazon Map Reduce/Simple Storage Service.
- **Software as a Service (Saas):** SaaS supports a software distribution with specific requirements. In this layer, the users can access an application and information remotely via the Internet and pay only for that they use. Sales force is one of the pioneers in providing this service model. Saas is actually a model of software deployment whereby the provider licenses an application to the customers for use as a service on demand. The capability provided to the end users is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through a thin client interface such as a web browser (e.g., web enabled e-mail). The end users does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user specific application configuration settings. Today SaaS is offered by companies such as Google, Salesforce, Microsoft, Zoho, etc.

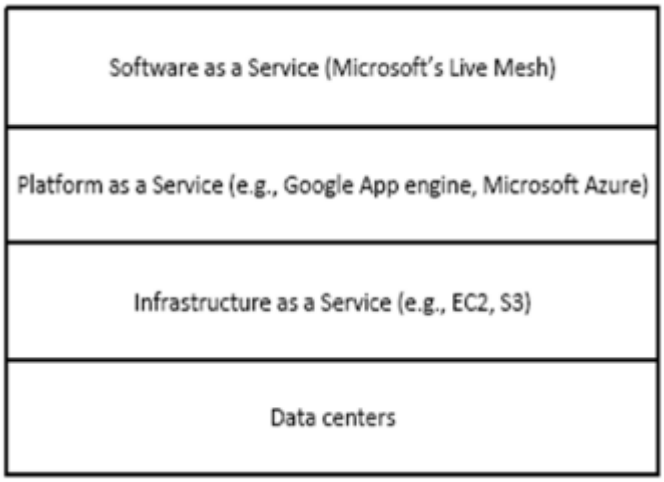

Figure 1: Cloud Layers

- There is also an extra layer, **Monitoring-as-a-Service (MaaS):** It is the outsourced provisioning of security, primarily on business platforms that leverages the Internet to conduct business. MaaS has become increasingly popular over the last decade. Since the advent of Cloud Computing, its popularity has grown even more. Security monitoring involves protecting an enterprise or government client from cyber threats. A security team plays a crucial role in securing and maintaining the confidentiality, integrity, and availability of IT assets. The major functionality of MaaS is to monitor the working of the top three layers SaaS, PaaS and IaaS. [5]

2.3 Cloud Computing Deployment Models

There are three types of Cloud Computing deployment models:

1. **Private Cloud (or Internal Cloud):** A type of Cloud in which the Cloud services are delivered over a network which is open for public usage. It refers to Cloud Computing on private networks. Private Clouds are built for the exclusive use of one client, providing full control over data, security, and quality of service. Private Clouds can be built and managed by a company's own IT organization or by a Cloud provider.
2. **Public Cloud (or External Cloud):** A type of Cloud that is implemented on a Cloud-based secure environment that is safeguarded by a firewall. Private Cloud as it permits only the authorized users can use the data. In this model, Computing resources are dynamically provisioned over the Internet via Web applications or Web services from an off-site third party provider. Public Clouds are run by third parties, and applications from different customers are likely to be mixed together on the Cloud's servers, storage systems, and networks.
3. **Hybrid Cloud (or Mixed Cloud):** A type of Cloud which is integrated. It can be an arrangement of two or more Cloud servers, i.e. private, public or community Cloud that is bound together but remain individual entities. This environment intersects and combines multiple public and private Cloud models. Hybrid Clouds introduce the complexity of determining how to distribute applications across both a public and private Cloud. [5]

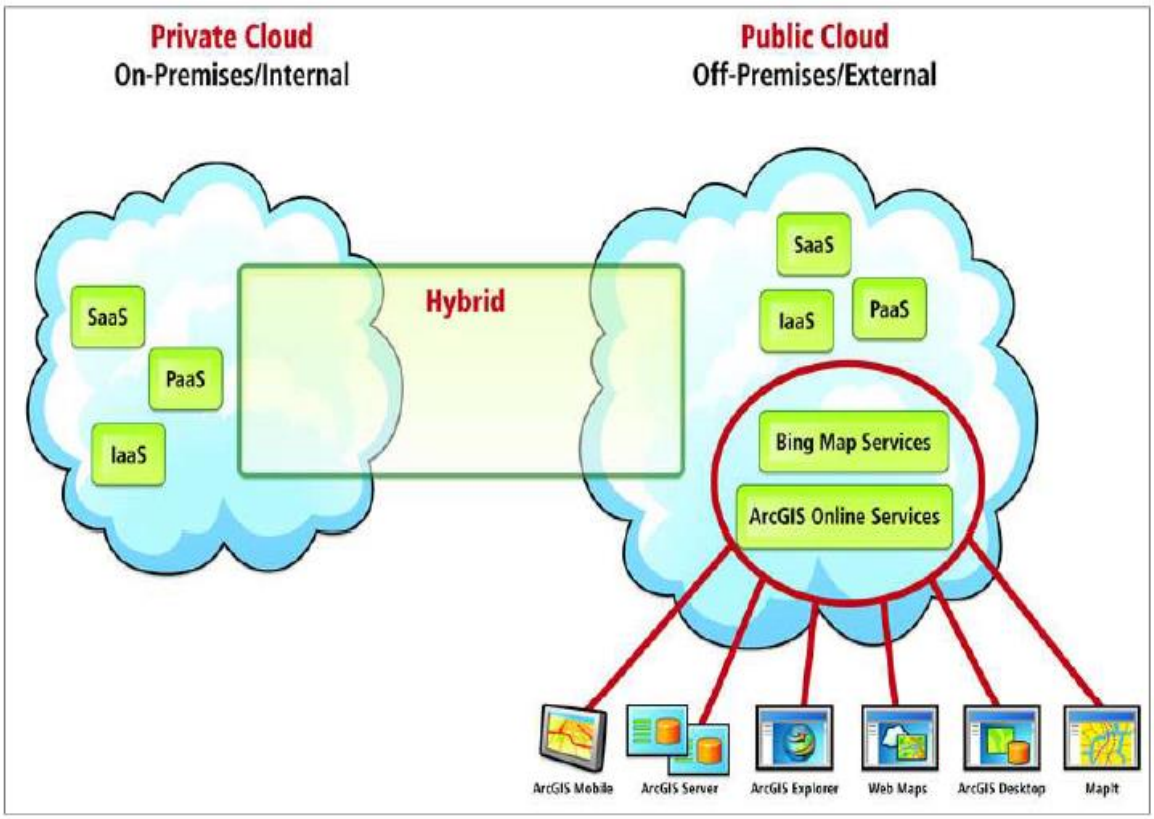

Figure 2: Cloud Models

4. There is an extra type of deployment model, **Community Cloud:** A type of Cloud in which the setup is mutually shared between many organizations that belong to a particular community. [14]

2.4 Cloud Computing Features

Below are the Cloud Computing features:

- **Scalability and On-Demand Services:** Cloud Computing provides resources and services for users on demand. The resources are scalable over several data centers.
- **Quality of Service (QoS):** Cloud Computing can guarantee QoS for users in terms of hardware or CPU performance, bandwidth, and memory capacity.
- **User-Centric Interface:** Cloud interfaces are location independent and they can be accessed by well-established interfaces such as Web services and Web browsers.
- **Autonomous System:** Cloud Computing systems are autonomous systems managed transparently to users. However, software and data inside Clouds can be automatically reconfigured and consolidated to a simple platform depending on user's needs.
- **Pricing – Cloud:** Computing does not require up-front investment. No capital expenditure is required. Users may pay and use or pay for services and capacity as they need them. [5]

2.5 Mobile Cloud Computing Definition

There are several existing definitions for Mobile Cloud Computing. In general, it is a running service on a resource rich Cloud server which is used by a thin mobile client. It can also be referred when mobile nodes play as a resource provider role in a peer-to-peer network. MCC can be considered as a network with certain characteristics. The need for adaptability, scalability, availability and self-awareness in Cloud Computing concept is taken and is expanded to Mobile Cloud Computing. [15]

Alternatively, MCC could be defined in a more comprehensive way as it is quoted as follows: "Mobile Cloud Computing at its simplest, refers to an infrastructure where both the data storage and data processing happen outside of the mobile device. Mobile Cloud applications move the Computing power and data storage away from mobile phones and into the Cloud, bringing applications and MCC to not just smartphone users but a much broader range of mobile subscribers". [5]

Mobile Cloud Computing at its simplest refers to an infrastructure where both the data storage and data processing happen outside of the mobility devices (e.g., tablet PC, smart-phone). Mobile Cloud apps move the Computing power and data storage capacity away from mobile phones and into the Cloud power, bringing applications and MC to not just smartphone users but a much broader range of mobile subscribers. [2]

Mobile Cloud Computing is a paradigm where data processing and storage are moved from mobile device to powerful and centralized Computing platforms located in Clouds over the internet. All these centralized applications are then accessed over the wireless connection based on a thin native client or web browser on the mobile devices. Alternatively, Mobile Cloud Computing can be defined as a combination of mobile web and Cloud Computing, which is the most popular tool for mobile users to access applications and services on the Internet. [6]

Mobile Cloud Computing has three different definitions:

1. The term Mobile Cloud Computing means to run an application for mobile on a remote resource rich server as displayed in while the mobile device acts like a thin client connecting over to the remote server through 3G.
2. Consider other mobile devices themselves too as resource providers of the Cloud making up a mobile peer-to-peer network. This approach supports user mobility, and recognizes the potential of Mobile Clouds to do collective sensing as well.
3. The Cloudlet concept proposed by Satyanarayanan is another approach to Mobile Cloud Computing. The mobile device offloads its workload to a local Cloudlet comprised of several multi-core computers with connectivity to the remote Cloud servers. PlugComputers can be considered good candidates for Cloudlet servers because of their form factor, diversity and low power consumption. They have the same general architecture as a normal computer, but are less powerful, smaller, and less expensive, making them ideal for role small scale servers installed in the public infrastructure. [13]

2.6 Mobile Cloud Computing Architecture

The general architecture of MCC proposed can be shown in the picture below:

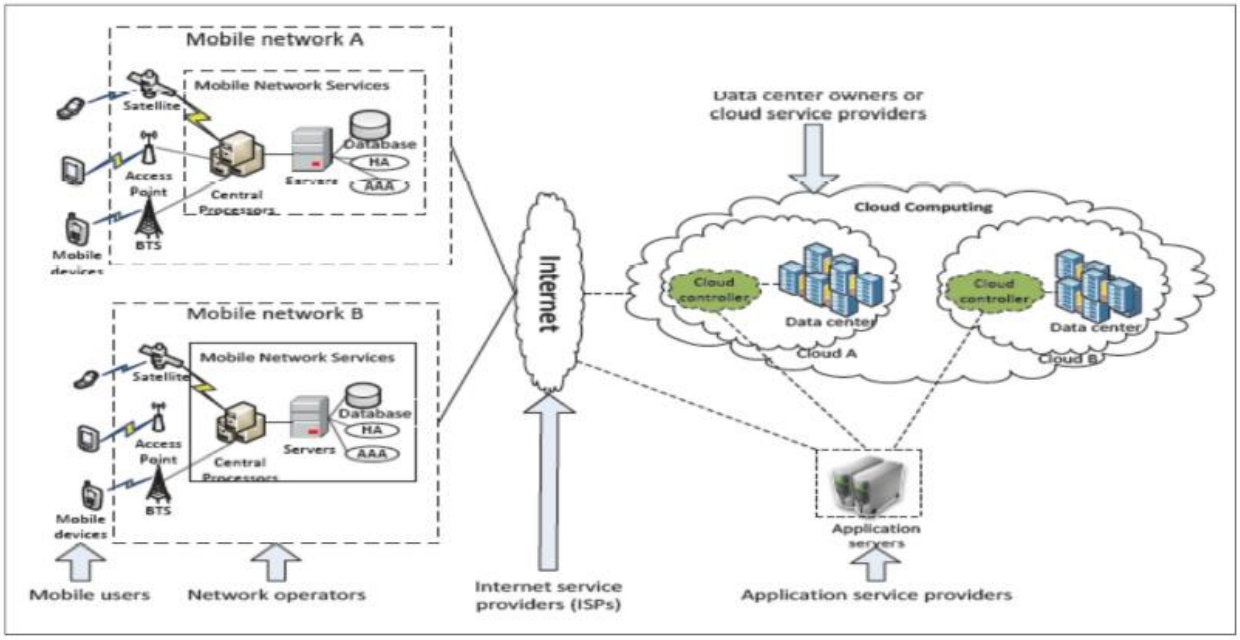

Figure 3: Mobile Cloud Architecture

Mobile devices are connected to the mobile networks via base stations (e.g., base transceiver station (BTS), access point, or satellite) that establish and control the connections (air links) and functional interfaces between the networks and mobile devices. Mobile user's requests and information (e.g., ID and location) are transmitted to the central processors that are connected to servers providing mobile network services. Here, Mobile network operators can provide services to mobile users as AAA (Authentication, Authorization and Accounting) based on the home agent (HA) and subscriber's data stored in databases. After that, the subscriber's requests are delivered to a Cloud through the Internet. In Cloud, the Cloud controllers process the requests to provide mobile users with the corresponding Cloud services. These services are developed with the concepts of utility Computing, virtualization, and service oriented architecture (e.g. web application, and database servers). [5]

Mobile Cloud Computing has three components, mobile device, wireless communication channel and Cloud. Mobile devices have resource constraint in terms of battery power,

memory, processing power and have different types of hardware, operating system, and input -output interface. Wireless communication channel has different radio access technologies such as GPRS, 3G, WLAN and WiMAX with variable network conditions in terms of limited and unstable bandwidth. [7]

There are two types of Architecture in Mobile Cloud Computing:

1. **Non Cloudlet Architecture:** there are three components Mobile client, Transmission channel and Cloud. Mobile client requests desired service from Cloud and Cloud provides the service. Cloud is owned by an organization or Cloud provider and services thousands of users at time. In this architecture, main disadvantage is communication latency for getting service from distant Cloud.
2. **Cloudlet Architecture:** a local Cloudlet contains cached copy of data. It is installed between client and Cloud. The cost of installation is less as compared to Cloud as it is only a data center at business premises. A Cloudlet services only a few users and has less communication latency as compared to Cloud. Cloudlet is owned by local business. [8]

2.7 Mobile Cloud Computing Models

There are three Mobile Cloud Models:

1. **Client Model:** In this model, mobile device act as client and mobile user access service is offered by Cloud by thin layer of interface web browser. Cloud charges for services till the duration client is connected. Client model depicts Software as a Service model of Cloud Computing.
2. **Client / Cloud Model:** In client /Cloud model, the concept of task partitioning comes in which mobile users give a part of task to Cloud for processing.
3. **Cloud Model:** In Cloud model, mobile device itself is the part of Cloud. One or more mobile devices create the structure of Cloud. [7]

2.8 Mobile Cloud Computing Features and Challenges

2.8.1 Features –Advantages

The main objective of Mobile Cloud Computing is to provide a convenient and rapid method for users to access and receive data from the Cloud, such convenient and rapid method means accessing Cloud Computing resources effectively by using mobile devices. Below there are enlisted the advantages of Mobile Cloud Computing:

- **Extending battery lifetime:** Battery is one of the main concerns for mobile devices. Several solutions have been proposed to enhance the CPU performance, and to manage the disk and screen in an intelligent manner, to reduce power consumption. However, these solutions require changes in the structure of mobile devices, or they require a new hardware that results in an increase of cost and may not be feasible for all mobile devices.
- **Improving data storage capacity and processing power:** Storage capacity is also a constraint for mobile devices. MCC is developed to enable mobile users to store/access the large data on the Cloud through wireless networks.
- **Improving reliability:** Storing data or running applications on Clouds is an effective way to improve the reliability since the data and application are stored

and backed up on a number of computers. This reduces the chance of data and application lost on the mobile devices. In addition, MCC can be designed as a comprehensive data security model for both service providers and users.

- **Dynamic provisioning:** Dynamic on-demand provisioning of resources on a fine-grained, self-service basis is a flexible way for service providers and mobile users to run their applications without advanced reservation of resources.
- **Scalability:** The deployment of mobile applications can be performed and scaled to meet the unpredictable user demands due to flexible resource provisioning. Service providers can easily add and expand an application and service without or with little constraint on the resource usage.
- **Multi-tenancy:** Service providers (e.g., network operator and data center owner) can share the resources and costs to support a variety of applications and large number of users.
- **Ease of Integration:** Multiple services from different service providers can be integrated easily through the Cloud and the Internet to meet the users' demands. [1]

The advantages of Mobile Cloud Computing are:

- Mobile devices allow users access to Cloud services anywhere and anytime.
- Mobile Cloud services can give information about a user's location, context, and requested services to improve user experience.
- Each mobile device has storage, Computing, sensing, and power resources which are advantageous.
- Mobile Computing can help to overcome some problem of Cloud Computing such as solving the problem of WAN latencies by using Cloudlet.
- Major problems faced by MCC are discussed such as stability of wireless connectivity, tackling the unnecessary battery usage etc. Certain barriers such as network availability and bandwidth are focused. Two aspects of security issues such as mobile device security and Cloud security are addressed. [4]

The major characteristics of Mobile Cloud Computing are listed below:

- **Flexibility/Elasticity:** Users can rapidly access provision Computing resources without human interaction. User Capabilities can be rapidly and elastically provisioned, in some cases dynamically, to quickly scale out or up.
- **Scalability of Infrastructure:** In the physical servers, new nodes can be added or dropped from the network with limited modifications to infrastructure set up and software. According to demand mobile Cloud architecture can scale horizontally or vertically easily.
- **Broad Network Access:** User capabilities and ability are available over the network and can be accessed through standard mechanisms that promote use by heterogeneous platforms like mobile phones, laptops, and PDAs etc.
- **Location Independence:** Location independence is another characteristic of Mobile Cloud Computing. There is a sense of different location independence where customer generally has no control or knowledge over the exact location of the provided resources. But it may be able to specify location at a higher level of abstraction from country, state, or datacenter.

- **Reliability:** Through the use of multiple redundant site reliability can be improved and this makes Cloud Computing more worthy for disaster recovery applications and business continuity. [6]

2.8.2 Challenges and strategies to solve them

2.8.2.1 Challenges

The major challenge of Mobile Cloud Computing comes from the characters of mobile devices and wireless networks, as well as their own restriction and limitation, and such challenge makes application designing, programming and deploying on mobile and distributed devices more complicated than on the fixed Cloud devices. The important factors that affect assessing from Cloud Computing are below:

- **Limitations of mobile devices:** While discussing mobile devices in Cloud the first thing is resource constrain. Though smart phones have been improved obviously in various aspects such as capability of CPU and memory, storage, size of screen, wireless communication, sensing technology, and operation systems, still have serious limitations such as limited Computing capability and energy resource, to deploy complicated applications. By contrast with PCs and Laptops in a given condition, these smart phones like iPhone 4S, Android serials, Windows Mobile serials decrease 3 times in processing capacity, 8 times in memory, 5 to 10 times in storage capacity and 10 times in network bandwidth.
- **Quality of communication:** In contrast with wired network uses physical connection to ensure bandwidth consistency, the data transfer rate in Mobile Cloud Computing environment is constantly changing and the connection is discontinuous due to the existing clearance in network overlay. Furthermore, data center in large enterprise and resource in Internet service provider normally is far away to end users, especially to mobile device users. In wireless network, the network latency delay may 200 ms in 'last mile' but only 50 ms in traditional wired network.
- **Division of application services:** In Mobile Cloud Computing environment, due to the issue of limited resources, some applications of compute-intensive and data-intensive cannot be deployed in mobile devices, or they may consume massive energy resources. Therefore, we have to divide the applications and use the capacity of Cloud Computing to achieve those purposes, which is: the core Computing task is processed by Cloud, and those mobile devices are responsible for some simple tasks only. In this processing, the major issues affecting performance of Mobile Cloud Computing are: data processing in data center and mobile device, network handover delay, and data delivery time. The following strategies can be used to response to the above challenges:
    - Upgrade bandwidth for wireless connection, make the web content more suitable for mobile network using regional data centers.
    - Deploy the application processing node at the 'edge' of the Cloud in order to reduce data delivery time.
    - Duplicate mobile devices to Cloud using virtualization and image technologies, to process Data-Intensive Computing (DIC) and Energy-Intensive Computing, such as virus scanning in mobile devices. Dynamically optimize application push in Cloud and the division with mobile terminals. [1]

There are several challenges in the process of consuming Web Services (WS) from mobile clients. The following are enlisted:

- **Loss of connection:** The interaction between clients and service requires a steady connection. However, due to the mobility of the clients and the wireless network setup, mobile clients can be temporarily removed from the previous connected network and later may enter to another network. In such occurrences, either service requests or responses may fail to be delivered to their destination.
- **Bandwidth/Latency:** Cell networks have limited bandwidth and are often billed based on the amount of data transferred. However, even a simple SOAP message often contains a large amount of XML data/information, which consumes a lots of bandwidth and the transmission can cause major network-latency. In addition, the SOAP messages contain mostly XML tags that are not all necessary for the mobile clients.
- **Limited resources:** Mobile clients are "thin clients" with limited processing power. The boundaries are essential to mobility and not just the failings of current technology. For example, a service mash up involves parsing and combining different WS results requires a lot of computation. The challenges are minimizing the data processing on mobile clients and extending processing power beyond mobile clients. In addition, several mobile platforms do not include necessary libraries for SOAP Web Services.

Most of the challenge of Mobile Cloud Computing comes from the characters of mobile devices and wireless networks and their own restriction and limitation. All these challenge makes application more complicated than on the fixed Cloud devices. The entire limitations of mobile devices, quality of wireless communication and support from Cloud Computing to mobile are all important factors that affect accessing from Cloud Computing. [2]

Major limitations and solutions of Mobile Cloud Computing is listed below:

- **Low Bandwidth:** Since mobile network resource is much smaller compared with the traditional networks bandwidth is the one of major important issues in Mobile Cloud environment. Therefore, P2P Media Streaming for distributing small bandwidth among the subscriber who are located nearby in the same area for the similar content such as the same video. Using this procedure, each user can transmit or exchanges parts of the same content with second users, which is resulted in improvement of content quality, especially for videos transmission.
- **Security and Privacy in the Cloud:** In Mobile Cloud Computing security and privacy has become the biggest concern. When establishing a remote Cloud base infrastructure certainly any organization will give away private data and information which might be sensitive and confidential. Then it gives to the Cloud service provider to manage, protect and retain them. The existence of the company might be jeopardous, so before taking any decision all the possible alternatives should be explored. Therefore, users might feel uncomfortable surrendering their data to a third party.
- **Prone to Attack:** It is more vulnerable to external hack attacks and threats to store information in the Cloud. Nothing on the internet is completely protected. Sensitive data and information may be stealth on the internet as many hackers and malicious users always lurk for the chances.
- **Dependency and Vendor Lock-In:** One of the major disadvantages of Mobile Cloud Computing is the implicit dependency on the internet service provider. It is

really painful and cumbersome if one user wants to switch from one provider to some other provider as he has to transfer large number of data from the previous provider to the new one. This is another main reason why have to carefully and thoroughly contemplate in all options when picking a vendor.

- **Limited Control and Flexibility:** Since all the applications and services run on remote or third party virtual environments, users have limited control over the whole function and execution of the hardware and software. In addition, since remote software is being used for Mobile Cloud Computing, it usually lacks the features of an application running locally.
- **Increased Vulnerability:** Privacy and security related Cloud based solutions is more vulnerable target for hackers and malicious users as all Cloud based solutions are exposed on the public internet. Many biggest players suffer from serious attack and security breakage in the internet. Nothing on the internet is fully secured.

2.8.2.2 Strategies

The following strategies can be used to reduce to the challenges in Mobile Cloud Computing: for wireless connection upgrade bandwidth and make the web content more usable for mobile network using regional data centers. In order to reduce data delivery time, deploy the application processing node at the 'edge' of Cloud. Duplicate mobile devices to Cloud using virtualization and image technologies, to process Data-Intensive Computing (DIC) and Energy-Intensive Computing, such as virus scanning in mobile devices. Optimize application push in Cloud dynamically and the division with mobile terminals. [6]

There are some Data Security Issues concerning the Mobile Cloud:

- **Privacy and Confidentiality:** Once the client host data to the Cloud there should be some guarantee that access to that data will only be limited to the authorized access. Inappropriate access to customer sensitive data by Cloud personnel is another risk that can pose potential threat to Cloud data. Assurances should be provided to the clients and proper practices and privacy policies and procedures should be in place to assure the Cloud users of the data safety. The Cloud seeker should be assured that data hosted on the Cloud will be confidential.
- **Data Integrity:** With providing the security of data, Cloud service providers should implement mechanisms to ensure data integrity and be able to tell what happened to a certain data set and at what point. The Cloud provider should make the client aware of what particular data is hosted on the Cloud, the origin and the integrity mechanisms put in place.
- **Data Location and Relocation:** Cloud Computing offers a high degree of data mobility. Consumers do not always know the location of their data. However, when an enterprise has some sensitive data that is kept on a storage device in the Cloud, they may want to know the location of it. They may also wish to specify a preferred location (e.g. data to be kept in India). This, then, requires a contractual agreement, between the Cloud provider and the consumer that data should stay in a particular location or reside on a given known server. Also, Cloud providers should take responsibility to ensure the security of systems (including data) and provide robust authentication to safeguard customers' information. Another issue is the movement of data from one location to another. Data is initially stored at an appropriate location decide by the Cloud provider.

However, it is often moved from one place to another. Cloud providers have contracts with each other and they use each other's resources.

- **Data Availability:** Customer data is normally stored in chunk on different servers often residing in different locations or in different Clouds. In this case, data availability becomes a major legitimate issue as the availability of uninterruptible and seamless provision becomes relatively difficult. [7]

The following issues have not been sufficiently solved:

- **Supporting continuous mobility while ensuring connectivity to the Cloud**: Even if the reception is sufficient, data costs and latency has a huge impact on these kinds of Mobile Cloud Computing apps. When supporting mobility and connectivity, some of the questions we need to contemplate are; How can a user device know of impending dis-connectivity? In what ways can the most 'stable' and 'efficient' surrogates be chosen so as to ensure seamless connectivity? What fault-tolerance mechanisms can be employed to minimize potential failures?
- **Security in Mobile Clouds:** Although an issue of paramount importance, little research has been carried out in this regard. Although many of the reviewed frameworks mention the need for security and trust, very few of them have actually implemented it and have left the implementation for future directions.
- **Incentives for surrogates:** If users are to be persuaded to collaborate and share their resources with others, there needs to be motivation either through monetary or social incentives to do so. An interesting method is using common goals, but in the absence of common activities this will not prevail. In the case of monetary incentives, several questions need to be answered such as: how is credit represented in a Mobile Cloud? how will monetary transactions proceed in a secure method? how will the price of resources be decided? Using social incentives such as suggested in also raises challenges such as preventing free riding and enforcing standards. [13]

The research challenges are defined as the issues that include how to abstract the complex heterogeneous underlying technology, how to model all the different parameters that influence the performance and interactivity of the application, how to achieve optimal adaptation under different constraints, how to integrate computation and storage with the Cloud while preserving privacy and security.

The full potential of Mobile Cloud applications can only be unleashed, if computation and storage is offloaded into the Cloud, but without hurting user interactivity, introducing latency or limiting application possibilities. The applications should benefit from the rich built-in sensors which open new doorways to more smart mobile applications. As the mobile environments change, the application has to shift computation between device and Cloud without operation interruptions, considering many external and internal parameters. [12]

2.9 Mobile Cloud Computing Applications

Various applications based on Mobile Cloud Computing have been developed and served to users, such as Google's Gmail drive, Maps and Navigation systems for Mobile, I- Cloud from Apple Moto Blur from Motorola (with a special feature called remote wipe) Amazon 's new "Cloud-accelerated" Web browser Silk. Silk is a "split browser whose software resides both on Kindle Fire and EC2. The applications reinforced by mobile Cloud Computing include mobile commerce, mobile learning, and

mobile healthcare and other areas. Mobile applications extended extensive share in a global mobile market. Various mobile applications have engaged the recompenses of Mobile Cloud Computing. The following are the few inferences:

- **m-Commerce:** Mobile commerce (m-commerce) is a buying and selling of products using mobile devices. The m-commerce applications normally used to achieve some tasks that necessitate mobility (e.g., mobile transactions and payments, mobile messaging, and mobile ticketing). The m-commerce applications have to face various complications (e.g., low network bandwidth, high complexity of mobile device configurations, and security). Subsequently, m-commerce applications are integrated into Cloud Computing environment to solve these issues (X. Yang et al,2010).
- **m-Learning:** Mobile learning (m-learning) is an electronic learning (e-learning) and mobility. However, traditional m-learning applications have limitations in terms of high cost of devices and network, low network transmission rate, and limited educational resources (X. Chen et al, 2010; H. Gao et al, 2010; Jian Li, 2010). Cloud based m-learning applications are presented to solve these limitations, for example utilizing a Cloud with the large storage capacity and powerful processing ability, the applications offer learners with much comfortable services in terms of information size, processing speed.
- **m-HealthCare:** MCC in medical applications is used to minimize the limitations of traditional medical treatment [e.g., small physical storage, security and privacy, and medical errors (D. Kopec et al, 2013)]. Mobile healthcare (m-healthcare) offers mobile users with appropriate help to access resources easily. m-Healthcare provides healthcare organizations a diversity of on-demand services on Clouds rather than standalone applications on local servers.
- **m-Banking:** M-Banking is an uprising in traditional banking services, where user can avail the bank services provided to them through their mobile despite of location and time (Z. Li et al, 2001). Transaction can be done even if user is busy in his routine work via SMS or the mobile Internet but can also use special programs, called mobile applications, downloaded to the mobile device.
- **m-Game:** Mobile game (m-game) is a prospective market producing incomes for service providers. M-game can completely offload game engine requiring large Computing resource (e.g., graphic rendering) to the server in the Cloud, and gamers only interact with the screen interface on their devices (Jasleen et al, 2013) demonstrates that offloading (multimedia code) can save energy for mobile devices, thereby increasing game playing time on mobile devices. [7]

Applications of Mobile Cloud Computing:

- **Image processing:** If user/subscriber visit foreign museum, he can't perceive the language written in each object of the museum. He can take picture of the object and using Mobile Cloud Computing can understand the language written over the object. An optical character recognition (OCR) program on a collection of mobile devices can give this ability to users.
- **Natural language processing:** Language translation is one possible application for Mobile Cloud Computing. Translation is a viable candidate for language processing since different sentences and paragraphs can be translated independently, and this is experimentally explored in using Pangloss-Lite.

- **Crowd Computing:** Video recordings from multiple mobile devices can be spliced to construct a single video that covers the entire event from different angles, and perspectives.
- **Sharing GPS/Internet data:** It is more efficient to share data among a group of mobile devices that are near each other, through local-area or peer-to-peer networks. It is not only cheaper, but also faster.
- **Sensor data applications:** Now-a-days almost every mobile device is built with sensors which are used to read data. Some sensors such as GPS, accelerometer, thermo sensor, light sensor, clock and compass may be time stamped and associated with other phone readings. In order to gather precious information in different situation different queries can be executed.
- **Multimedia search:** Mobile phones may store different types of multimedia content such as videos, photos, and music. Shazam is a music identification service for mobile phones, that searches for similar songs in a central database, in the context of the mobile Cloud, the searching could be executed on the contents of nearby phones.
- **Social networking:** Since sharing user content is a popular way we interact with friends on social networks such as Facebook, integrating a Mobile Cloud into social networking infrastructure could open up automatic sharing and p2p multimedia access and this will also reduce the need to back up and serve all of this data on huge servers. [13], [6]

3. Cloudlets and Cooperative Caching

In this chapter we are going to describe the Cloudlet's architecture and its usefulness if it is integrated in Mobile Cloud Computing architecture. Also we are going to define what is cooperative caching and how this can be used best in order to make Mobile Cloud Computing more efficient.

3.1 Cloudlet's Architecture

Despite a lot of achievements that MCC provides, there are still issues like low bandwidth, high latency, service availability, quality of service (QoS) and service cost to be addressed. These concerns arise mostly from rapid growth in the number of mobile users and their expectations of MCC services. Bandwidth is limited in wireless networks compared to normal wired networks. Users need more availability despite mobile devices lack of connectivity and they demand higher QoS with less service cost. Also network latency is a big burden in improving QoS and user experience while using a distant Cloud. These problems are more tangible in applications that offer cognition or virtual reality services which demand low latency and high bandwidth.

Considering these problems, researchers realized utilizing resources and services with more locality is more cost efficient with better availability, faster connectivity and less latency. This has led to the concept of the Cloudlet. A Cloudlet is a computer or a cluster of computers connected at the edge of the network to provide low-latency access to Computing resources for mobile devices. The mission of Cloudlets is to alleviate resource constraints of mobile devices and also to reach better network latency. Speech recognition, natural language processing, computer vision and graphics, machine learning, augmented reality and other computation-intensive applications would benefit the most from the Cloudlet approach.

Cloudlet is considered as the middle tier of a 3-tier hierarchy: mobile device, Cloudlet and Cloud. A Cloudlet can also be viewed as a resource rich center at the proximity of users.

Cloudlet Architecture taken from [15]:

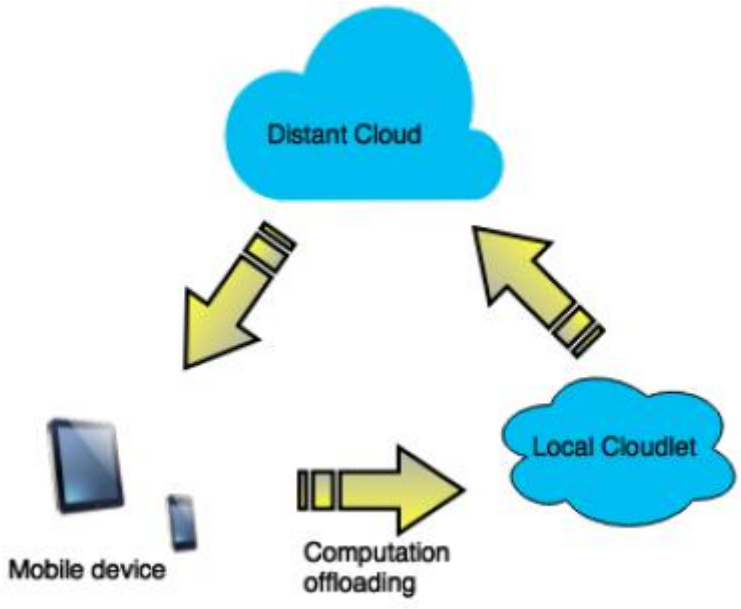

Figure 4: Cloudlet Architecture

In the Cloudlet concept, mobile device offloads its workload to a resource-rich, local Cloudlet. Cloudlets would be situated in common areas such as coffee shops, libraries or university halls, so that mobile devices can connect and function as a thin client to the Cloudlet. A Cloudlet could be any first hop element at the edge of network while it has four key attributes. It has only soft state, it should be resource rich and well-connected, with low end-to-end latency and also it follows a certain standard for offloading (e.g. Virtual machine migration). In other words, a Cloudlet's failure is not critical, it has strong internal connectivity and high bandwidth wireless LAN and it should be in logical and physical proximity of user to reduce the network latency.

There are two main approaches to implement Cloudlet infrastructure using Virtual Machine (VM) technology. In both of these architectures it is important that Cloudlet could go back to its beginning state after being used (e.g. by post-use clean up). A VM based approach is broadly used since it can cleanly encapsulate and separate the transient guest software environment from the Cloudlet infrastructure's permanent host software and it's less brittle than other approaches like process migration or software virtualization.

Cloudlets utilize rapidly deployed VMs which the client can customize freely upon their need to make the VM image or VM overlay which has the application and all necessary requirements to run properly. In both types of implementations, the VM image or overlay is created at runtime by user which is quite flexible for offloading the workload to the Cloudlet. Nevertheless, despite this flexibility, the procedure of creating an image or a VM overlay and also application status encapsulation could be quite time taking. At the end, it is totally dependent on application design, needs and environment whether to choose using Cloudlets as resource rich sources or not. [15]

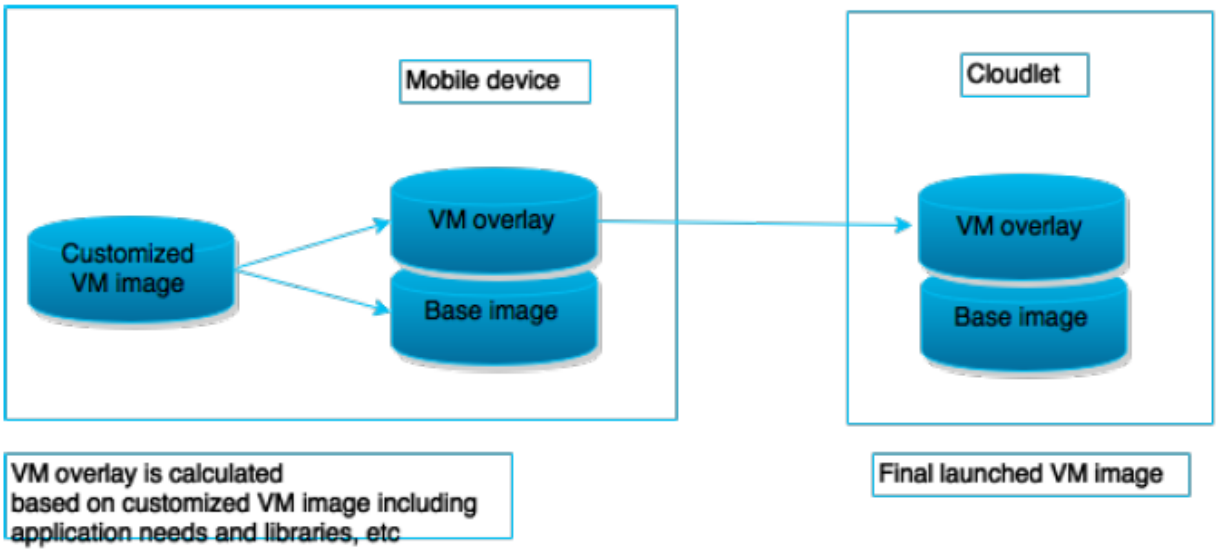

Figure 5: Cloudlet VM Synthesis

In this architecture a mobile user exploits virtual machine (VM) technology to rapidly instantiate customized service software on a nearby Cloudlet and then uses that service over a wireless LAN, the mobile device typically functions as a thin client with respect to the service. A Cloudlet is a trusted, resource-rich computer or cluster of computers that's well-connected to the Internet and available for use by nearby mobile devices.

Using a Cloudlet also simplifies the challenge of meeting the peak bandwidth demand of multiple users interactively generating and receiving media such as high-definition video and high-resolution images.

Cloudlets are decentralized and widely dispersed Internet infrastructure components whose compute cycles and storage resources can be leveraged by nearby mobile computers. Essentially, a Cloudlet resembles a "data center in a box": it's self-managing, requiring little more than power, Internet connectivity, and access control for setup. This simplicity of management corresponds to an appliance model of Computing resources and makes it trivial to deploy on a business premises such as a coffee shop or a doctor's office. Internally, a Cloudlet resembles a cluster of multicore computers, with gigabit internal connectivity and a high-bandwidth wireless LAN. For safe deployment in unmonitored areas, the Cloudlet can contain a tamper-resistant or tamper-evident enclosure with third-party remote monitoring of hardware integrity.

A future in which Cloudlet infrastructure is deployed much like Wi- Fi access points today is something that is achievable ambition. Indeed, it would be relatively straightforward to integrate Cloudlet and Wi-Fi access point hardware into a single, easily deployable entity. A key challenge is to simplify Cloudlet management. Widespread deployment of Cloudlet infrastructure won't happen unless software management of that infrastructure is trivial— ideally, it should be totally self-managing. Tightly restricting software on Cloudlets to simplify management is unattractive because it constrains application innovation and evolution. Instead, an ideal Cloudlet would support the widest possible range of mobile users, with minimal constraints on their software.

The proposal is transient customization of Cloudlet infrastructure using hardware VM technology. The emphasis on "transient" is important: pre-use customization and post-use cleanup ensures that Cloudlet infrastructure is restored to its pristine software state after each use, without manual intervention. A VM cleanly encapsulates and separates the transient guest software environment from the Cloudlet infrastructure's permanent host software environment. The interface between the host and guest environments is narrow, stable, and ubiquitous, which ensures the longevity of Cloudlet investments and greatly increases the chances of a mobile user finding compatible Cloudlets anywhere in the world.

This Cloudlet's physical proximity is essential: the end-to-end response time of applications executing within it must be fast (a few milliseconds) and predictable. If no Cloudlet is available nearby, the mobile device can gracefully degrade to a fallback mode that involves a distant Cloud or, in the worst case, solely its own resources. Full functionality and performance can return later, when the device discovers a nearby Cloudlet. A VM-based approach is less brittle than alternatives such as process migration or software virtualization.6 It's also less restrictive and more general than language-based virtualization approaches that require applications to be written in a specific language such as Java or C#.

The other approach is called dynamic VM synthesis. A mobile device delivers a small VM overlay to the Cloudlet infrastructure that already possesses the base VM from which this overlay was derived. The infrastructure applies the overlay to the base to derive the launch VM, which starts executing in the precise state in which it was suspended. To appreciate its unique attributes, it's useful to contrast dynamic VM synthesis with the alternative approach of assembling a large file from hash-addressed chunks. Researchers have used variants of this alternative in systems such as LBFS, Casper, Shark, the Internet Suspend/Resume system, the Collective and KeyChain. All these variants have a probabilistic character to them: chunks that aren't available nearby (in the local cache, on portable storage, and so on, depending on the specific variant) must be obtained from the Cloud. Thus, bandwidth to the Cloud and the hit ratio on chunks are the dominant factors affecting assembly speed. Dynamic VM synthesis

differs in two key ways. First, its performance is determined solely by local resources: bandwidth to Cloudlet and the Cloudlet's compute power. Local hardware upgrades can thus translate directly to faster VM synthesis. Second, WAN failures don't affect synthesis. Even a Cloudlet that's totally isolated from the Internet is usable because the mobile device delivers the overlay. In this case, provisioning the Cloudlet with base VMs could be done via physical storage media. [8]

There are two types of Cloudlets: the ad hoc Cloudlet and the elastic Cloudlet. The ad hoc Cloudlet consists of dynamically discovered nodes in the LAN network. These nodes run a Node Agent that can spawn Execution Environments to deploy components in. When nodes join or leave the Cloudlet, the Cloudlet Agent will recalculate the deployments, migrating components if needed. The elastic Cloudlet runs on a virtualized infrastructure, where nodes run in virtual machines. Here, the Cloudlet Agent can spawn new nodes when more resources are needed, or stop nodes when too much resources are allocated. This type of Cloudlet comes close to the VM based Cloudlet envisioned by Satyanarayanan, but with extra middleware in the VM (NA and EE) that manages the application. [9]

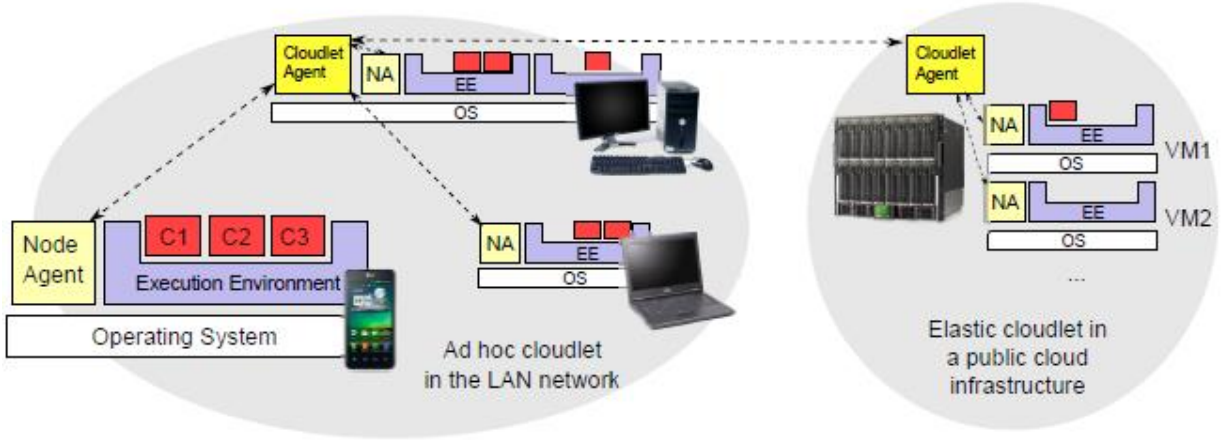

Figure 6: Cloudlet Types

We read that in Cloudlet architecture mobile users can access Cloudlet one hop away, thus reduces bandwidth utilization and efficiency. Computation and data storage mostly happens outside the mobile device, to a remote server which will complete the computation task and send the results back to the client. Offloading techniques currently available are client server communication, virtualization and mobile agents.

Cloudlet architecture reduces the gap between mobile devices and remote servers by offloading workload to a local Cloudlet with connectivity to remote servers. It uses VM technology. It uses dynamic VM synthesis. The mobile device transmits a small VM overlay to the Cloudlet and applies it to a compatible base VM to generate the launch VM-temporarily created for a mobile client to execute the task and then restored to its previous state after each execution. [11]

3.2 Cloudlet's Challenges

Although Cloudlets may solve the issue of latency, there are still two important drawbacks of the VM based Cloudlet approach. First, one remains dependent on service providers to actually deploy such Cloudlet infrastructure in LAN networks. To alleviate this constraint, the authors of [9] propose a more dynamic Cloudlet concept, where all devices in the LAN network can cooperate in the Cloudlet. A second drawback of VM based Cloudlets is the coarse granularity of VMs as unit of distribution. Instead of executing the whole application remotely in the VM and using a thin client protocol,

better performance can be achieved by dynamically partitioning the application in components. As resources in the Cloudlet will still be limited, chances are that even the Cloudlet runs out of resources when many users execute their VM simultaneously on the Cloudlet infrastructure. With component offloading, a more flexible allocation of the Cloudlet resources is possible, so that priority is given for latency-critical parts of the application, while non real-time parts can be offloaded to a more distant Cloud. [9]

In Satyanarayanan's architecture a mobile user exploits VMs to rapidly instantiate customized service software on a nearby Cloudlet and uses the service over WLAN. A Cloudlet is a trusted, resource-rich computer or a cluster of computers well connected to the Internet and available for use by nearby mobile devices. Rather relying on a distant Cloud, the Cloudlets eliminate the long latency introduced by wide-area networks for accessing the Cloud resources. Cloudlets allow high abstraction and personalization of the Computing environment by using VMs, but lack from fine-grained execution adaptation. [13]

3.3 Cooperative Caching Definition

The concept of cooperative caching is based on the idea of demanding the necessary data from a neighbor node in the network instead of the original resource. Different approaches have been proposed for cooperative caching, such as, caching on mobile nodes, caching on intermediate or proxy nodes or caching on the edge of network.

By technological improvements in smart phones and other mobile devices, mobile clients are capable of sharing data between themselves as peers. In this way they can stay independent from the origin server where the data comes from. Mobile Cooperative Caching is an aggregation of this alternative with the concept of caching for mobile devices. In Mobile Cooperative Caching, mobile devices try to form an ad hoc network with other mobile nodes in the proximity to share the relevant data. To develop this kind of network, one should consider proper policies and select efficient algorithms regarding cache records invalidation, consistency level, cache record placing and searching. [15]

Cooperative caching improves the response time by reducing VM synthesis time by caching the previous state. Cooperative caching consists of multiple distributive caches to improve system response time. [14]

Having distributed caches permits a system to deal with concurrent client request as well as sharing contents. With cache different VM synthesis states the users can get the service from cache otherwise request is given to the base layer to get the corresponding launch VM. If the corresponding base VM is not present we have to contact distant Cloud for the service. Data caching increases battery life in mobile devices by reducing wireless communication. [11]

4. Cloudlet Scenario with Hilout Algorithm

In this chapter we are going to give the description of the Cloudlet scenario that is implemented in this Thesis as long as the way that Hilout algorithm logic is embedded to it in order to make the decision which data vectors are considered as Outliers and can't be handled by the Cloudlet.

4.1 Cloudlet Scenario

As we saw in the previous chapter, Cloudlet is defined as a computer or a cluster of computers connected at the edge of the network to provide low-latency access to Computing resources for mobile devices. Cloudlet is considered as the middle tier of a 3-tier hierarchy: mobile device, Cloudlet and Cloud. A Cloudlet can also be viewed as a resource rich center at the proximity of users.

In this Thesis, we are using a Cloudlet simulation framework, which is called CloudSim and it is going to be described in the next chapter, in order to implement and evaluate a specific Cloudlet scenario. In this scenario the Cloudlet is considered as a repository, practically a MySQL schema table, in which data vectors, data from IoT devices, arrive with multi-dimensions' values sequentially. At this point Hilout Algorithm takes place. Each time that a data vector wants to be processed by the Cloudlet, the algorithm decides if it can be processed or not. In case that it can't be processed, it is marked as an outlier and stored in the repository.

A simplistic overview of the scenario is given below. The details are going to be explained in the next sections of this chapter.

Scenario:

- We have a Cloudlet where m (maximum 500) devices are connected. In this Cloudlet, data are stored in a repository R. We consider that the selected data should be in accordance with the current data that devices report. We should decide which data will be stored. These multivariate data (n variables) should be 'similar' with the data present in the Cloudlet otherwise they are transferred in the Cloud for storage and further processing (practically they stored in the repository as Outliers).
- Vectors arrive in the Cloudlet accompanied by a timestamp.
- The Cloudlet processor (a component responsible to manage the incoming vectors) decides if each vector is similar with the dataset.
- We execute the Hilout algorithm to identify if the incoming vector is an outlier compared to the stored vectors.
- Each vector gets a weight which is the sum of the distance with its k-neighbors.
- The weight of the k-neighbors is updated only if the incoming vector is not an outlier.
- Based on the timestamps, we extend the Hilout algorithm to take into consideration a window W where the stored vectors are considered for the calculations (details about this in the Hilout algorithm section)
- When a stored vector is out of the window, its weight is penalized and reduced. The higher the difference with W the higher the penalty becomes.
- The final weight of the neighbors-stored vectors is taken into consideration for deciding if the incoming vector will be stored or sent to the Cloud.

- Below it is a picture that gives a snapshot of how the Cloudlet is conceived in this scenario:

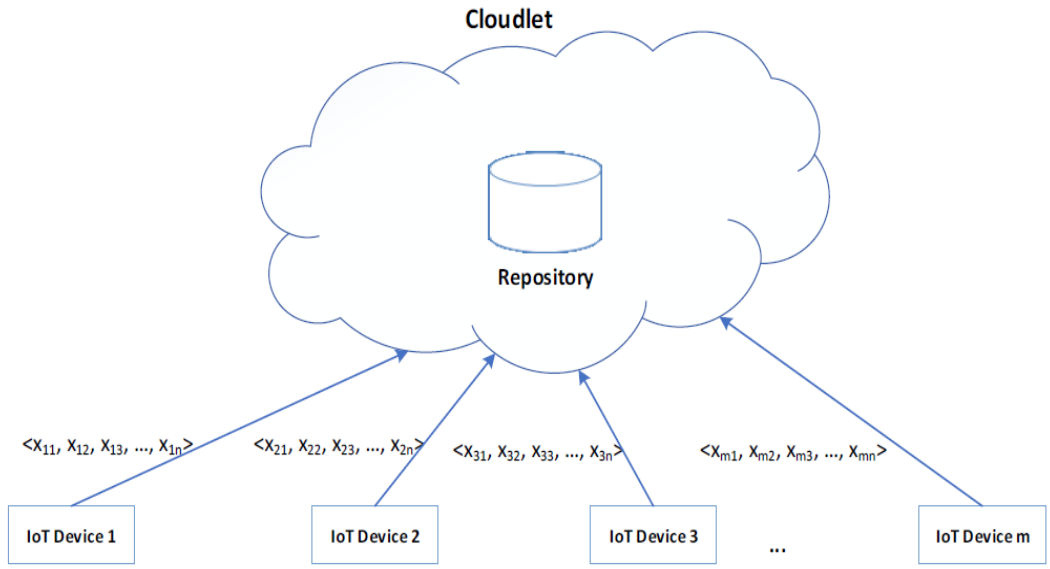

Figure 7: Cloudlet

Each data vector that is stored in the Cloudlet(repository) has the below attributes:

1. nodeID: the ID of the IoT device of the data vector. The ID can be within the range 0-500.
2. recordID: the ID of the vector that it stored in the repository. This number is unique for each record.
3. dimensions(a-j): each time the simulation runs the program decides with a randomized way how many dimensions the data vectors have. The range of the dimensions can be from 1 to 10. The names of the dimensions follow the alphabet sequence (a, b, c, ...)
4. time: it is the timestamp of a data vector when this is stored in the repository.
5. weight: the weight that is stored for an arriving data vector is actually the sum of the distances between the dimensions of the data vector and its neighbors' dimensions, data vectors that are already stored in the repository. The logic with which the neighbors are selected is going to be explained in the section of Hilout Algorithm.
6. marked: This boolean attribute is true for a data vector when it is stored as an outlier and false when it is not.

4.2 Hilout Algorithm

In this section we are going to describe the logic of the Hilout algorithm that is applied every time a data vector wants to enter the Cloudlet. Before we do, we need to say few words about the Hilout algorithm itself.

Hilout finds distance-based Outliers, but uses the ranks of distance instead of the absolute distance in outlier detection. Specifically, for each object, o, Hilout finds the k-nearest neighbors of o, denoted by *nn1(o), ... nnk(o)*, where k is an application-dependent parameter. The weight of object o is defined as:

$$w(o) = \sum_{i=1}^{k} dist(o, nn_i(o)).$$

Equation 1: weight

All objects are ranked in weight-descending order. The top-l objects in weight are output as Outliers, where l is another user-specified parameter. Computing the k-nearest neighbors for every object is costly and does not scale up when the dimensionality is high and the database is large. To address the scalability issue, Hilout employs space-filling curves to achieve an approximation algorithm, which is scalable in both running time and space with respect to database size and dimensionality.

In our scenario Hilout is a little bit differentiated because in the process of detecting if the incoming vector is an outlier it takes into consideration also the temporal factor. Not only the spatial proximity with the neighbors is taken into account for the decision but also the temporal proximity. If the neighbors' timestamps are not within a time window W, it is defined from the timestamp of the incoming vector minus a fixed interval, an extra penalty is added to the weight of each of the neighbors-vectors accordingly, which is taken into consideration for the decision for the incoming vector. The higher the difference with W the higher the penalty becomes.

In order to see how Hilout algorithm works for the scenario, we describe the algorithm in physical steps below:

- the number of the dimensions(d) can be between 2-10 and the number of the data vectors(v) can be between 1-1000.The data vectors try to enter the Cloudlet sequentially.
- Incoming vector's data follow the Gaussian distribution, which means that every dimension's value for each vector follows this distribution. The mean of the distribution is initially 100 but every 10 vectors that are inserted it deviates from this value left or right (- or +) according to a deviation that is given by the exponential distribution. Deviation of the Gaussian distribution is set to the value from the {5,25,50}. Details about the creation of the input data is given to the Input Data section.
- A decision of how many neighbors(k) we are going to look into for the incoming vector is taken. The number of the neighbors(k) is randomized between the half and the total number of vectors of the current dataset that is stored in the repository.
- After the neighbors' number is set we find the neighbors according to the proximity of them comparing to the input vector's position, as the Hilout algorithm defines. This means that the k stored vectors with the lowest relative distance to the incoming vector are considered as neighbors.
- The decision if the incoming vector is an outlier or not is taken comparing the weights of the neighbors with the weight of it. If the weight of the incoming vector is the highest, then it is marked as an outlier and is stored in the repository with weight 0. If it is not an outlier, then the weights of the neighbors are updated

(plus the relative distance with the incoming vector) and the incoming vector is stored.

- At this point we need to clarify how the weight of the neighbors is calculated each time and how the incoming vector's. When the incoming vector arrives to the Cloudlet, the timestamp of it minus 1 millisecond becomes the time window (W) we mentioned in the beginning of this section. If the timestamp of each neighbor is out of range of the W, then a penalty is added to its weight. This penalty is set as the absolute difference of one of the W's edges and the neighbor's timestamp. On the other side there is no penalty for the incoming vector because its timestamp is one of the edges of the W. Summarizing the weight of the incoming vector is only the sum of the relative distance with its neighbors but the weight of the neighbors is the sum, relative distance with the incoming vector is not taken into account, plus the possible penalty due to the time window. Last but not least, the penalty is taken into account for each neighbor every time an incoming vector wants to enter the Cloudlet but is not added in the weight value which is stored for every vector in the repository.

4.2.1 Hilout Application Example

In this section we are going to see an example of Hilout applied on the Cloudlet repository for a very small dataset.

Input's information: number of vectors =5, number of dimensions = 3, Gaussian's mean = 100.0, variance = 50.

Results and Analysis of them:

Table 1: Hilout Example with 5 vectors

vectorID	Dimension a	Dimension b	Dimension c	Timestamp(long value)	weight	outlier
299	152.6	179.78	100.53	1529156644964	402.94	false
487	64.04	175.69	120.45	1529156645011	0	true
440	106.63	144.1	195.46	1529156645089	369.18	false
465	135.1	32.26	206.7	1529156645264	298.55	false
150	72.11	118.28	60.29	1529156645509	250.87	false

Iteration 1:

The first vector (vectorID: 299) will be stored without any check because the Cloudlet repository is empty. The weight of the vector is 0.

Iteration 2:

For the second vector (vectorID:487) the number of neighbors that is randomly generated is 1, so the Hilout takes place:

- Find 1 neighbor, there is only one stored vector anyway, $distance_{487,299} = 90.86$
- Timestamps of the W at this point is edge1: 1529156645011 and edge2: 1529156645010 (long values), $weight_{vectorID:487} = distance_{487,299}$ + penalty. The penalty for the vector with vectorID 487 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:487} = distance_{299,487} = 90.86$. For vector with

vectorID 299: $weight_{vectorID:299}$ = 0 + penalty = 0 + (edge2 – $timestamp_{vectorID:299}$) = (1529156645010-1529156644964) = 46. $weight_{vectorID:487}$ > $weight_{vectorID:299}$. This means that the current vector is an outlier and it will be stored with weight=0 and marked as outlier.

Iteration 3:

For the fourth vector (vectorID:440) the number of neighbors that is randomly generated is 1, so the Hilout takes place:

- Find 1 neighbor, there is only one stored vector, not outlier, anyway. $distance_{440,299}$ = 111.34.
- Timestamps of the W at this point is edge1: 1529156645089 and edge2: 1529156645088 (long values), $weight_{vectorID:440}$ = distance440,299 + penalty. The penalty for the vector with vectorID 440 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:440}$ = $distance_{440,299}$ = 111.34. For vector with vectorID 299: $weight_{vectorID:299}$ = 0 + penalty = 0 + (edge2 – $timestamp_{vectorID:299}$) = (1529156645088-1529156644964) = 124, $weight_{vectorID:440}$<$weight_{vectorID:299}$. This means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 111.34 and of the vector with vectorID 299 the same.

Iteration 4:

For the fourth vector (vectorID:465) the number of neighbors that is randomly generated is 2, so the Hilout takes place:

- Find 2 neighbors, there are only two stored vectors, not Outliers, anyway. $distance_{465,299}$ = 182.59, $distance_{465,440}$ = 115.95.
- Timestamps of the W at this point is edge1: 1529156645264 and edge2: 1529156645263 (long values), $weight_{vectorID:465}$ = $distance_{465,299}$ + $distance_{465,299}$+penalty. The penalty for the vector with vectorID 465 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:465}$ = 182.59+115.95 = 298.55. For the vector with vectorID 299: $weight_{vectorID:299}$ = 111.34 + penalty = 111.34 + (edge2 – $timestamp_{vectorID:299}$) = 111.34 + (1529156645263 – 1529156644964) = 111.34+299 = 410.34, $weight_{vectorID:440}$ = 111.34 + penalty = 111.34 + (edge2 – $timestamp_{vectorID:440}$) =111.34 + (1529156645263 – 1529156645089) = 111.34 + 174 = 285.34. $weight_{vectorID:440}$ < $weight_{vectorID:465}$ < $weight_{vectorID:299}$, this means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 298.55, of the vector with vectorID 299 is 293.94 and of the vector with vectorID 440 is 227.3.

Iteration 5:

For the fifth vector (vectorID:150) the number of neighbors that is randomly generated is 2, so the Hilout takes place:

- Find 2 neighbors between the 3 stored-not Outliers- vectors. $distance_{150,299}$ = 109, $distance_{150,440}$ = 141.87, $distance_{150,465}$ =181.11. So neighbors are the vectors with vectorID 299 and 440.
- Timestamps of the W at this point is edge1: 1529156645509 and edge2: 1529156645508 (long values), $weight_{vectorID:150}$ = $distance_{150,299}$ + $distance_{150,440}$+penalty. The penalty for the vector with vectorID 150 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:150}$ = 109+141.87 = 250.87. For the vector with vectorID 299: $weight_{vectorID:299}$ = 293.94 + penalty = 293.94 + (edge2 – $timestamp_{vectorID:299}$) = 293.94 + (1529156645508 – 1529156644964) = 293.94+ 544 = 837.94, $weight_{vectorID:440}$ = 227.3 + penalty =

227.3 + (edge2 – $timestamp_{vectorID:440}$) =227.3 + (1529156645508 – 1529156645509) = 227.3+ 419 = 646.3. $weight_{vectorID:150}$ < $weight_{vectorID:440}$ < $weight_{vectorID:299}$, this means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 250.87, of the vector with vectorID 299 is 402.94 and of the vector with vectorID 440 is 369.18.

4.3 Input Data

The input data, the values of the dimensions for each data vector, are not completely randomized. They follow the Gaussian distribution. Each dimension's value is given, in the code, by the formula: ***mean * randomNumber * deviation***. The deviation is set from the set {5,25,50}, the randomNumber is given by the Random number generator of Java and the mean is not a fixed value all the times. For every 10 data vectors that are inserted, Outliers or not, the mean deviates by a value that is given by the Exponential distribution, see section 4.3.2 about the Exponential distribution. This deviation can be positive or negative (+ or -). Initially the mean's value is 100. The deviation's is given, in the code, by the formula: ***(log (1- randomNumber)) / (-lambda)***. The lambda is set from the set {0.2,5}, the randomNumber is given by the Random number generator of Java and it must be double type and the log is the logarithm with base e.

We apply this input logic in order to see if this phenomenon, the deviation of the mean in the Gaussian distribution, affects the number of the Outliers that are detected.

At this point it should be useful to mention some basic characteristics of the Gaussian distribution and Exponential distribution in a theoretical perspective.

4.3.1 Gaussian Distribution

Normal distribution, also called Gaussian distribution, is the most common distribution function for independent, randomly generated variables. Its familiar bell-shaped curve is ubiquitous in statistical reports, from survey analysis and quality control to resource allocation.

The graph of the normal distribution is characterized by two parameters: the mean, or average, which is the maximum of the graph and about which the graph is always symmetric and the standard, which determines the amount of dispersion away from the mean. A small standard deviation (compared with the mean) produces a steep graph, whereas a large standard deviation (again compared with the mean) produces a flat graph.

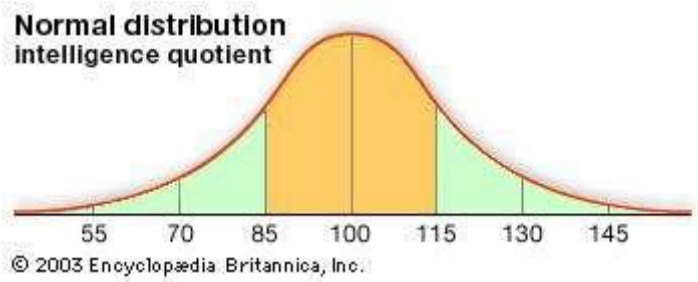

Figure 8: Normal Distribution

The normal distribution is produced by the normal density, the function is shown below:

$$f(x;\mu,\sigma^2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Equation 2: density function

In this exponential e is the constant 2.71828…, is the mean, and σ is the standard deviation. The probability of a random variable falling within any given range of values is equal to the proportion of the area enclosed under the function's graph between the given values and above the x-axis. Because the denominator, the denominator is shown below:

$$(\sigma\sqrt{2\pi})$$

Equation 3: denominator

, known as the normalizing coefficient, causes the total area enclosed by the graph to be exactly equal to unity, probabilities can be obtained directly from the corresponding area—i.e., an area of 0.5 corresponds to a probability of 0.5. Although these areas can be determined with calculus, tables were generated in the 19th century for the special case of = 0 and σ = 1, known as the standard normal distribution, and these tables can be used for any normal distribution after the variables are suitably rescaled by subtracting their mean and dividing by their standard deviation, (x − μ)/σ. Calculators have now all but eliminated the use of such tables.

The term "Gaussian distribution" refers to the German mathematician Carl Friedrich Gauss, who first developed a two-parameter exponential function in 1809 in connection with studies of astronomical observation errors. This study led Gauss to formulate his law of observational error and to advance the theory of the method of least squares approximation. Another famous early application of the normal distribution was by the British physicist James Clerk Maxwell, who in 1859 formulated his law of distribution of molecular velocities—later generalized as the Maxwell-Boltzmann distribution law.

A this point we have to mention that the most real-world data are NOT normally distributed. A paper by Micceri (1989) called "The unicorn, the normal curve and other improbable creatures" examined 440 large-scale achievement and psychometric measures. He found a lot of variability in distributions w.r.t. their moments and not much evidence for (even approximate) normality. In a 1977 paper by Steven Stigler called "Do Robust Estimators Work with Real Data" he used 24 data sets collected from famous 18th century attempts to measure the distance from the earth to the sun and 19th century attempts to measure the speed of light. He reported sample skewness and kurtosis. The data are heavy-tailed. On the contrary, in statistics, we assume normality oftentimes because it makes maximum likelihood convenient. This is what we do also in this Thesis. Although, convenience is not the only reason, the other reason we chose Gaussian distribution is to investigate the behavior of the data under a controlled range of values because a completely randomized input would be chaotic and not safe conclusions could be made of it.

4.3.2 Exponential Distribution

The exponential distribution (also called the negative exponential distribution) is a probability distribution that describes time between events in a Poisson Process. A Poisson process gives you a way to find probabilities for random points in time for a process. A "process" could be almost anything:

- Accidents at an interchange.
- File requests on a server.
- Customers arriving at a store.
- Battery failure and replacement.

The Poisson process can tell you when one of these random points in time will likely happen. For example, when customers will arrive at a store, or when a battery might need to be replaced. It's basically a counting process; it counts the number of times an event has occurred since a given point in time, like 1210 customers since 1 p.m., or 543 files since noon. An assumption for the process is that it is only used for independent events.

There is a strong relationship between the Poisson distribution and the Exponential distribution are intertwined. For example, let's say a Poisson distribution models the number of births in a given time period. The time in between each birth can be modeled with an exponential distribution (Young & Young, 1998).

The most common form of the pdf (Probability Density Function):

$$F(x;\lambda) = e^{-\lambda x} \; x > 0.$$

Equation 4: Probability Density Function

Where:

- e = the natural number e,
- λ = mean time between events,
- x = a random time.

For x less than 0, F (x; λ) = 0

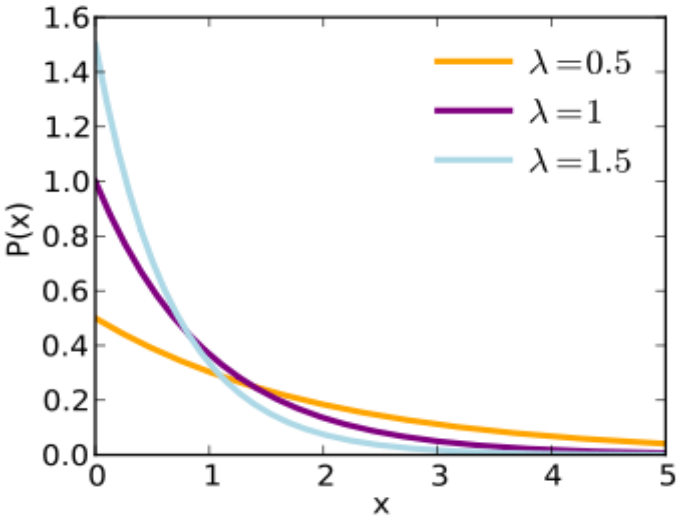

Figure 9: Exponential distribution

The formula for the cumulative distribution of the exponential distribution is:

$$F(x) = 1 - e^{-\lambda x}$$

Equation 5: Cumulative distribution function

$x \geq 0; \lambda > 0$

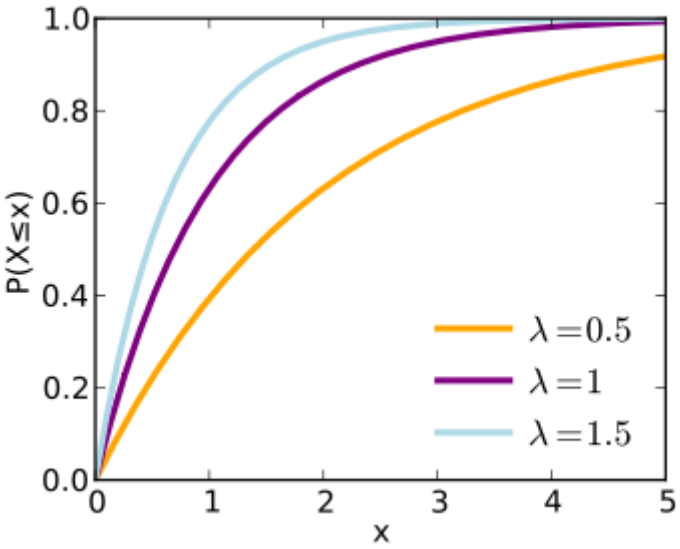

Figure 10: Cumulative distribution

In our case we use the quantile function for the Exponential distribution.

The quantile function for Exponential(λ) is derived by finding the value of Q for which $1-e^{-\lambda Q} = p$:

$$Q(p;\lambda) = \frac{-\ln(1-p)}{\lambda},$$

Equation 6: Quantile function

for $0 \leq p < 1$.

The quantile function is one way of prescribing a probability distribution, and it is an alternative to the probability distribution (pdf) or the cumulative distribution function. The quantile function, Q, of a probability distribution is the inverse of its cumulative distribution function F. The derivative of the quantile function, namely the quantile density function, is yet another way of prescribing a probability distribution. It is the reciprocal of the pdf composed with the quantile function.

We chose quantile function in order to provide the quantity of how much is the variance of the mean each time we decide to change it and if this movement of the range of values affects somehow the Outliers' detection.

4.4 PCA dimension reduction

4.4.1 Principal Component Analysis

Except the logic we described in the Hilout Algorithm section above, we thought that it could be also intriguing to add another factor that can be taken into account for the Outliers' detection. We thought to use PCA dimension reduction. Before we explain more how we added this feature in the algorithm it could be useful to say few words about what PCA is.

Principal component analysis (PCA) does what it says, finds the principal components of data. It is often useful to measure data in terms of its principal components rather than on a normal x-y axis. So what are principal components then? They're the underlying structure in the data. They are the directions where there is the most variance, the directions where the data is most spread out.

Principal component analysis (PCA) is the main linear technique for dimension reduction. Performs a linear mapping of the data to a lower-dimensional space in such a way that the variance of the data in the low-dimensional representation is maximized.

Dimension reduction is analogous to being philosophically reductionist: It reduces the data down into its basic components, stripping away any unnecessary parts.

When we get a set of data points, we can deconstruct the set into eigenvectors and eigenvalues. Eigenvectors and values exist in pairs: every eigenvector has a corresponding eigenvalue. An eigenvector is a direction. An eigenvalue is a number, telling you how much variance there is in the data in that direction, in the example above the eigenvalue is a number telling us how spread out the data is on the line. The eigenvector with the highest eigenvalue is therefore the principal component.

4.4.2 Hilout Algorithm with PCA

We saw in the above subsection a short description of what PCA is. Now we are going to explain how we applied this method in Hilout logic. We apply this method on each incoming vector's dimensions in order to see if this method, the dimension reduction, affects the percentage of the Outliers that are detected.

At this point we have to clarify that Hilout has been implemented as it is described in the Hilout Algorithm section and also with the addition of PCA dimension reduction method. Therefore, in the next chapter we show results for both the implementations of Hilout.

In order to see how Hilout algorithm with PCA works for the scenario, we describe the algorithm in physical steps below:

- the number of the dimensions(d) can be between 2-10 and the number of the data vectors(v) can be between 1-1000.The data vectors try to enter the Cloudlet sequentially.
- Incoming vector's data follow the Gaussian distribution, which means that every dimension's value for each vector follows this distribution. The mean of the distribution is initially 100 but every 10 vectors that are inserted it deviates from this value left or right (- or +) according to a deviation that is given by the exponential distribution. Deviation of the Gaussian distribution is set to the value from the {5,25,50}. Details about the creation of the input data is given to the Input Data section.
- A decision of how many neighbors(k) we are going to look into for the incoming vector is taken. The number of the neighbors(k) is randomized between the half and the total number of vectors of the current dataset that is stored in the repository.
- After the neighbors' number is set we find the neighbors according to the proximity of them comparing to the input vector's position, as the Hilout algorithm defines. The difference this time is that we apply the PCA method on all the records, which are not Outliers, of the dataset, expect the incoming vector, and we find the principal components of the dataset. These components define the correlation between the dimensions. The way we choose to decide which

dimensions we keep is simple. We check the first principal component and for the dimensions that have value > 0.4 we keep them. If the first component doesn't have such values, we move to the next one. If none of them has for at least one of their dimension values > 0.4 then we keep all of the dimensions and no dimension reduction is applied. In the end, the k stored vectors with the lowest relative distance to the incoming vector are considered as neighbors. Relative distance after the dimension reduction is considered only for the dimensions we chose.

- At this point we need to clarify how the weight of the neighbors is calculated each time and how the incoming vector's. When the incoming vector arrives to the Cloudlet, the timestamp of it minus 1 millisecond becomes the time window (W) we mentioned in the beginning of this section. If the timestamp of each neighbor is out of range of the W, then a penalty is added to its weight. This penalty is set as the absolute difference of one of the W's edges and the neighbor's timestamp. On the other side there is no penalty for the incoming vector because its timestamp is one of the edges of the W. Summarizing the weight of the incoming vector is only the sum of the relative distance with its neighbors, relative distance after the dimension reduction is considered only for the dimensions we chose through the PCA method , but the weight of the neighbors is the sum ,relative distance with the incoming vector is not taken into account, plus the possible penalty due to the time window. Last but not least, the penalty is taken into account for each neighbor every time an incoming vector wants to enter the Cloudlet but is not added in the weight value which is stored for every vector in the repository.

Just to make it more clear about the PCA logic we apply, we can give an example: Let's say we have 4 data vectors stored already in the repository and a 5th one arrives. We have 4 dimensions for each vector. The dataset of the repository for the example is the below:

1st record: {96.93,95.04,96.15,99.04}

2nd record: {93.94,106.34,101.99,94.39}

3rd record: {92.98,96.68,101.29,103.75}

4th record: {108.43,92.9,94.11,92.56}

When we apply the PCA we take the below PCA components with order from the highest (PCA1) to the lowest (PCA4):

PCA1 PCA2 PCA3 PCA4

dim1: [[-0.50, -0.50, -0.50, -0.50],

dim2: [0.86, -0.36, -0.29, -0.21],

dim3: [0.09, 0.64, 0.04, -0.77],

dim4: [-0.01, -0.47, 0.81, -0.35]]

According to the logic we described in the Hilout algorithm with PCA physical steps we scan the PCA1 and we check for values > 0.4. We see that dim2 has value that follows this criteria, so we choose the dim2 and we stop the procedure. If PCA1 didn't have such a value, we should move and check PCA2 etc.

4.4.3 Hilout Application Example with PCA

In this subsection we are going to see an example of Hilout with PCA applied on the Cloudlet repository for a very small dataset.

Input's information: number of vectors =5, number of dimensions = 3, Gaussian's mean = 100.0, variance = 50.

Results and Analysis of them:

Table 2: Hilout Example with PCA , with 5 vectors

vectorID	Dimension a	Dimension b	Dimension c	Timestamp(long value)	weight	outlier
225	40.36	87.93	71.34	1529234110627	191.36	false
107	132.78	150.17	119.84	1529234110686	0	true
378	140.08	40.1	50.77	1529234110745	287.02	false
279	59.83	144.73	29.09	1529234110989	196.16	false
477	59.97	110	142.69	1529234115804	126.7	false

Iteration 1:

The first vector (vectorID: 225) will be stored without any check because the Cloudlet repository is empty. The weight of the vector is 0.

Iteration 2:

For the second vector (vectorID:107) the number of neighbors that is randomly generated is 1, so the Hilout takes place:

- Find 1 neighbor, there is only one stored vector anyway. PCA is not applicable in this case because PCA requires at least 2 components, so in this case Hilout without PCA is applied. $distance_{107,225} = 121.52$
- Timestamps of the W at this point is edge1: 1529234110686 and edge2: 1529234110685 (long values), $weight_{vectorID:107} = distance_{107,225}$ + penalty. The penalty for the vector with vectorID 107 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:107} = distance_{107,225} = 121.52$. For vector with vectorID 225: $weight_{vectorID:225} = 0 + penalty = 0 + (edge2 – timestamp_{vectorID:225}) =$ (1529234110685-1529234110627) = 58. $weight_{vectorID:107} > weight_{vectorID:225}$. This means that the current vector is an outlier and it will be stored with weight=0 and marked as outlier.

Iteration 3:

For the second vector (vectorID:378) the number of neighbors that is randomly generated is 1, so the Hilout takes place:

- Find 1 neighbor, there is only one stored vector, not outlier, anyway. PCA is not applicable in this case because PCA requires at least 2 components, so in this case Hilout without PCA is applied. $distance_{378,225} = 112.49$
- Timestamps of the W at this point is edge1: 1529234110745 and edge2: 1529234110744 (long values), $weight_{vectorID:378} = distance_{378,225}$ + penalty. The penalty for the vector with vectorID 378 is 0 because its timestamp is one of the

edges of the W so $weight_{vectorID:378} = distance_{378,225}$ = 112.49. For vector with vectorID 225: weightvectorID:225 = 0 + penalty = 0 + (edge2 – $timestamp_{vectorID:225}$) = (1529234110744-1529234110627) = 117. $weight_{vectorID:378} < weight_{vectorID:225}$. This means that the current vector is an outlier and it will be stored with weight=0 and marked as outlier. This means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 112.49 and of the vector with vectorID 299 the same.

Iteration 4:

For the fourth vector (vectorID:279) the number of neighbors that is randomly generated is 2, so the Hilout takes place:

- Find 2 neighbors, there are only two stored vectors, not Outliers, anyway. The PCA is applied and the reduced dimensions' number is 1 and is the second dimension (b) that is returned, so the distance with the neighbors will be based on the second dimension. $distance_{279,225}$ = 56.8, $distance_{279,378}$ = 104.63.
- Timestamps of the W at this point is edge1: 1529234110989 and edge2: 1529234110988(long values), $weight_{vectorID:225} = distance_{279,225} + distance_{279,378}$+penalty. The penalty for the vector with vectorID 279 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:279}$ = 56.8+104.63 = 161.43. For the vector with vectorID 225: $weight_{vectorID:225}$ = 112.49 + penalty = 112.49 + (edge2 – $timestamp_{vectorID:225}$) = 112.49 + (1529234110988 – 1529234110627) = 112.49+361 = 473.49, $weight_{vectorID:378}$ = 112.49 + penalty = 112.49 + (edge2 – $timestamp_{vectorID:378}$) =104.63 + (1529234110988 – 1529234110745) = 104.63 + 243 = 347.63. $weight_{vectorID:279} < weight_{vectorID:378} < weight_{vectorID:225}$, this means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 161.43, of the vector with vectorID 225 is 169.29 and of the vector with vectorID 378 is 217.12.

Iteration 5:

For the fifth vector (vectorID:477) the number of neighbors that is randomly generated is 3, so the Hilout takes place:

- Find 3 neighbors, there are only three stored vectors, not Outliers, anyway. The PCA is applied and the reduced dimensions' number is 1 and is the second dimension (b) that is returned, so the distance with the neighbors will be based on the second dimension. $distance_{477,225}$ = 22.07, $distance_{477,378}$ = 69.9, $distance_{477,279}$ = 34.73.
- Timestamps of the W at this point is edge1: 1529234115804 and edge2: 1529234115803 (long values), $weight_{vectorID:477} = distance_{477,225} + distance_{477,378}$+$distance_{477,279}$+penalty. The penalty for the vector with vectorID 477 is 0 because its timestamp is one of the edges of the W so $weight_{vectorID:477}$ = 22.07+69.9+34.73 = 126.7. For the vector with vectorID 225: $weight_{vectorID:225}$ = 169.29 + penalty = 169.29 + (edge2–$timestamp_{vectorID:225}$) = 169.29 + (1529234115803–1529234110627) =169.29+5176 = 5345.29, $weight_{vectorID:378}$ = 217.12 + penalty = 217.12 + (edge2 – $timestamp_{vectorID:378}$) =217.12 + (1529234115803–1529234110745) = 217.12 + 5058 = 5275.12, $weight_{vectorID:279}$ =161.43+4814=4975.43. $weight_{vectorID:477} < weight_{vectorID:279} < weight_{vectorID:378} < weight_{vectorID:225}$, this means that the current vector enters the Cloudlet and it is not an outlier. The weight of the vector is 126.7, of the vector with vectorID 225 is 191.36, of the vector with vectorID 378 is 287.02 and of the vector with vectorID 279 is 196.16.

4.5 Variance per dimension

Another characteristic we study in this Thesis is the Variance per dimension. The results per case will be shown analytically in the next Chapter. In this section we see the definition of the variance of a data set and a simple example in which we see how it is calculated.

4.5.1 Variance of a Data Set: Definition

Variance (commonly denoted σ^2) is a very useful measure of the relative amount of 'scattering' of a given set. In other words, knowing the Variance can give you an idea of how closely the values in a set cluster around the mean. The greater the Variance, the more the data values in the set are spread out away from the mean.

Variance is an important calculation to become familiar with because, like the arithmetic mean, Variance is used in many other more complex statistical evaluations. The calculation of Variance is slightly different depending on whether you are working with a population (you do not intend to generalize the results back to a larger group) or a sample (you do intend to use the sample results to predict the results of a larger population). The difference is really only at the end of the process, so let's start with the calculation of the population.

To calculate the Variance of a population:

1. First, identify the arithmetic mean of your data by finding the sum of the values and dividing it by the number of values.
2. Next, subtract each value from the mean and record the result. This value is called the deviation of each score from the mean.
3. For each value, square the deviation.
4. Finally, divide the sum of the squared deviations by the number of values in the set. The resulting quotient is the Variance (σ^2) of the set.

4.5.2 Variance: Example

Let's calculate the Variance of set x:

$$x= \{12,7,6,3,10,5,18,15\}$$

We follow the steps from above:

- 1) First, calculate the arithmetic mean:

  $\mu= (12+7+6+3+10+5+18+15) /8=9.5$

- 2,3) Subtract each value from the mean to get the deviation of each value, square the deviation of each value:

Table 3: Variance example

Value−Mean=Deviation	$Deviation^2$
12−9.5=2.5	6.25
7−9.5=−2.5	6.25

6−9.5=−3.5	12.25
3−9.5=−6.5	42.25
10−9.5=0.5	0.25
5−9.5=−4.5	20.25
18−9.5=8.5	72.25
15−9.5=5.5	30.25
TOTAL (sum of deviation^2):	190.00

- 4) Finally, divide the sum of the squared deviations by the count of values in the data set:190/8 = 23.75. The Variance of set x is 23.75

In our case the set x is each dimension. More analysis will be given, with practical results per case, in the next Chapter.

4.6 Goals of the scenario

After the analysis we did on the Cloudlet scenario with the Hilout detection of Outliers, in the above sections, we can now define the goals of the simulations we ran and their results are shown in the next chapter. The goals are:

- see how the number of the data vectors affects the number of Outliers that are detected
- see how the number of the dimensions affects the number of Outliers that are detected
- see how the deviation of the mean in the Gaussian distribution affects the number of Outliers that are detected
- see how the lambda parameter in the Exponential distribution affects the number of Outliers that are detected
- see how the PCA dimension reduction affects the number of Outliers that are detected

Of course these goals are not independent with each other. In the different simulation cases we followed all these factors are combined.

4.7 Why we adopt this scenario

The answer to the question: "Why we adopt this scenario?" is a combination of two parameters:

1. The first parameter is the Hilout algorithm. Hilout algorithm has been designed to efficiently detect the top n Outliers, in our case 1 per time a data-vector wants to enter into the Cloudlet, of a large and high-dimensional data set. This is a

feature that really fits our use cases because we take into account not only 2-d and 3-d set of dimensions but a multi-dimensional set, until 10 maximum, for Outliers' detection.

2. The second parameter is the centralized nature of the Cloudlet repository. The advantages of a centralized system are:

    - **Data Integrity:** The single greatest benefit of centralizing and management of data is data integrity. One of the cardinal rules of database design is that no redundancy is allowed. That is, no piece of data should ever be repeated within the database. This aids in the maintaining of data as accurate and as consistent as possible and enhances data reliability.
    - **Cost effectiveness:** More cost effective than other types of database systems. By controlling data in a central repository, redundancy and its associated costs are eliminated and productivity is increased.
    - **Increased Efficiency:** With a central repository, all data is integrated and maintained centrally so that manual data processing is eliminated and the resources devoted to multiple data management can be redirected to other business needs.
    - **Enhanced data Quality:** Having parallel databases and transferring data among them can result in data loss or poor-quality data too. Integrating all your data in a central repository improves data quality and consistency to make better assessments.
    - **Changeability:** Data kept in the same location is easier to be changed, re-organized, mirrored, or analyzed.
    - **Accessibility:** All the information can be accessed at the same time from the same location. Updates to any given set of data are immediately received by every end-user.

In our case we care about the data integrity and cost effectiveness because the proper application of Hilout bases a lot on the consistency and not redundancy of data that are kept in the repository. We care about the increased efficiency because manual data processing is not needed at all and this is eliminated by the existence of this feature. Enhanced data quality is crucial to the application of the algorithm. Changeability is a very useful feature because Hilout applies changes very often in the repository. Last but not least accessibility provides immediate notification of the end-users regarding any update on the repository which is really useful especially in the case a user is detected as outlier.

5. CloudSim framework and Hilout's experimental results

In this chapter we give some basic information regarding CloudSim framework and we provide the experimental results of the different cases that we ran and the analysis of these results.

5.1 CloudSim framework

5.1.1 ClouSim framework's features

CloudSim framework is a programming tool designed to normalize and accelerate the process of conducting experimental studies using Cloud Computing environments. Conducting experimental studies using real Cloud infrastructure can be excellent time-consuming due to their size and complexity as well as high cost of access to these infrastructures.

The primary objective of CloudSim is to provide a generalized, and extensible simulation framework that enables seamless modeling, simulation, and experimentation of emerging Cloud Computing infrastructures and application services. By using CloudSim, researchers and industry-based developers can focus on specific system design issues that they want to investigate, without getting concerned about the low level details related to Cloud-based infrastructures and services.

CloudSim follows multi-layered design and consists of many architectural components.

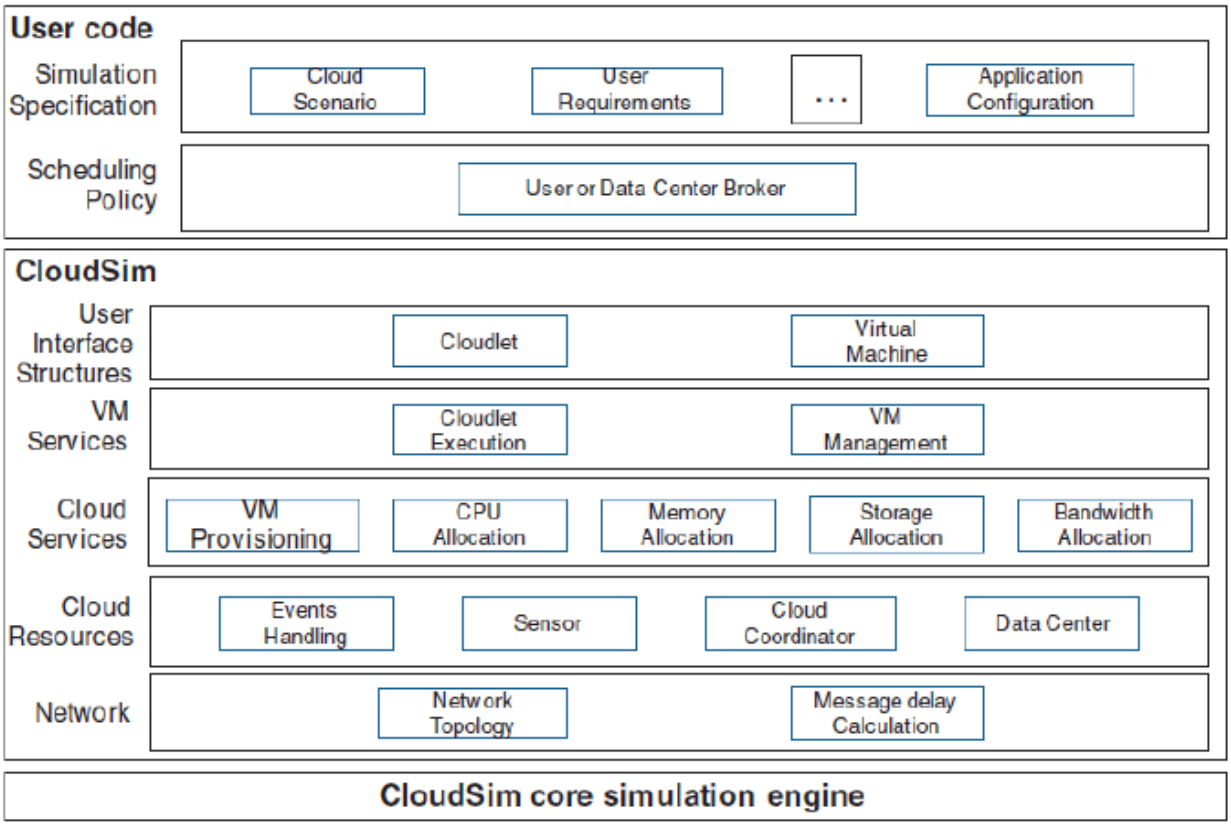

Figure 11: ClouSim Architecture

The CloudSim simulation level provides support for modeling and simulating Cloud-based virtualization environments including special management interfaces for virtual machines, memory, storage, and bandwidth. Fundamental issues such as virtual machine feed, implementation execution management, and dynamic system status

monitoring are implemented in this layer. In addition, a Cloud service provider who wants to study the effectiveness of the various policies for sharing virtual machines in hosting systems should focus on this level. This level also reveals the functions in which a Cloud application developer can be expanded to look at complex workload profiles combined with application performance.

CloudSim's top layer level is the user code that defines and customizes the basic entities for hosting systems such as the number of machines, specifications, applications (the number of tasks and their requirements), the number of users and types of implementation, and the policies of the intermediate system.

5.1.2 Presentation of major CloudSim classes

1. **BwProvisioner:** This is an abstract class that implements the system bandwidth allocation policy for virtual machines. The primary role of this class is to allocate the range of the network to all competing virtual machines to each hosting system, having as a minimum the requirements of the virtual machine and as a limit the available range of the hosting system.
2. **CloudCoordinator:** This abstract class has every data center created in the Cloud system. It is responsible for the periodic monitoring and control of resources that bind and release data centers
3. **Cloudlet:** This class models Cloud-based application services. CloudSim manages the complexity of an application in the form of Computing resource requirements. Each application has pre-set requirements in order to perform indefinitely throughout its existence.
4. **CloudletScheduler:** This abstract class uses different policies to share the computational power each Cloudlet requires on each virtual machine. The main policies implemented are shared space and time sharing.
5. **Datacenter:** This class implements the core of hardware at the infrastructure level as it is currently offered by Cloud service providers. It contains a set of computerized hosting systems built on the available hardware in each data center such as memory, cores, storage, and storage units.
6. **Datacenter Broker / Cloud Broker:** It is the class that acts as the intermediary for communicating between the requirements of the SaaS model and Cloud service providers. These requirements relate to the quality of the service but also to the SLA. The intermediate system therefore assigns the SaaS model to a Cloud service provider and binds this provider with the necessary infrastructure to ensure compliance with QoS and SLA. The difference between the intermediate system and the CloudCoordinator is that the first represents the client of the system while the second one works on behalf of the data center.
7. **Host:** This class models a physical resource such as server-oriented processing or storage. It contains important information such as the amount of memory, the number and type of processing unit kernels, and the policy of allocating resources to virtual machines.
8. **Vm:** This class models a virtual machine that is handled and hosted by a Cloud-based hosting system. Each virtual machine consists of features such as memory, processor, capacity, and prediction policy as defined by CloudScheduler.

9. **VmAllocationPolicy:** This abstract class represents the policy used to commit virtual machines to their respective hosting systems. The main function that it performs is to commit virtual machines to hosting systems that have sufficient memory, capacity and computational power for the operation of each virtual machine.
10. **VmScheduler:** This abstract class is implemented by the hosting systems to select the binding policy of processing kernels in virtual machines. Policies that are used are shared space, time-sharing, and matching policies for specific applications with particular kernels.
11. **CloudSim:** This is the main class that is responsible for managing the queues with the events to be executed and the step-by-step execution of the simulation. Each event created at runtime enters the queue with future events to execute. Then the events scheduled to run in the next steps of the run go out of the future list and are placed in the list of events to be executed. Each time the events that are going to run come out of the last list and dynamic Cloud simulation processes are performed such as disabling resources, increasing customer requirements, creating new clients, and extreme scenarios such as service failure resulting in resumption of simulation.

5.1.3 CloudSim Configuration for our experiments

In our experiments we have only one configuration: a datacenter with one host and run one Cloudlet on it. We take a simple case because our focus is not on the CloudSim framework's different configurations but on the application of Hilout algorithm on a Cloudlet.

5.2 Experiments and results

In our experiments we try to see how some factors affects the number of Outliers that are detected by the version of the Hilout algorithm we apply and we described its logic in Chapter 4. These factors are:

- Number of data vectors, 4 configurations: 50, 100, 500 ,1000
- Number of dimensions, 10 configurations: 2,3,4,5,6,7,8,9,10
- Deviation of mean for Gaussian distribution, 3 configurations: 5,25,50
- Lambda parameter for Exponential distribution, 2 configurations: 0.2,5
- PCA dimension reduction

We tried to examine the 5 above factors with 2 main cases:

- Case without PCA dimension reduction
- Case with PCA dimension reduction

For both of the cases the experiments we ran are summarized in the below table:

Table 4: Experiments per case

Vectors	Dimensions	Deviation	Lambda
50,100,500,1000	2 - 10	5	0.2

50,100,500,1000	2 - 10	5	5
50,100,500,1000	2 - 10	25	0.2
50,100,500,1000	2 - 10	25	5
50,100,500,1000	2 - 10	50	0.2
50,100,500,1000	2 - 10	50	5

5.2.1 Results for case without PCA

In the charts that are shown below we can see the average number of Outliers that was detected for each case. For example: for vectors:100, dimensions:3, deviation of the mean in the Gaussian distribution: 25, lambda parameter of the Exponential distribution: 5, the average number of Outliers that are detected is 1. Average in this case is considered the number of Outliers that appears most frequently. If we consider that we ran the above example 5 times, then the 3 of them 1 outlier is detected.

The display of the charts follows for each of the 4 configurations that are shown in the **Table 4** except 2 configurations: 1) deviation:5, lambda:0.2,2) deviation:5, lambda:5. These 2 configurations detect 0 Outliers on average, therefore there is no point to display them in a chart.

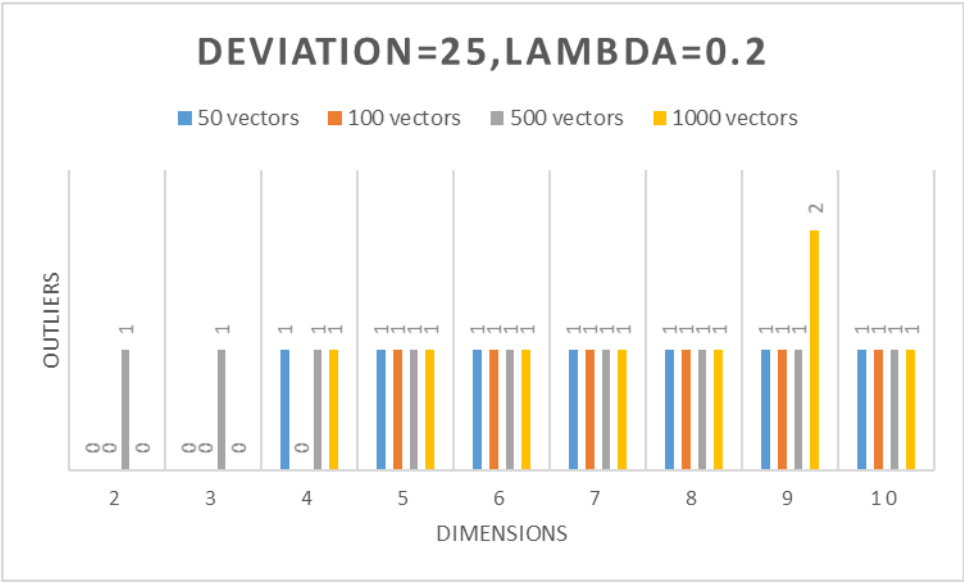

Figure 12: deviation:25,lambda:0.2 configuration

Deviation=25, lambda=0.2: We see that until 4 dimensions 0 Outliers are detected. From 5 to 10 dimensions the Outliers' number on average is stabilized to 1, except in the case of 9 Outliers for 1000 vectors where 2 Outliers are detected on average.

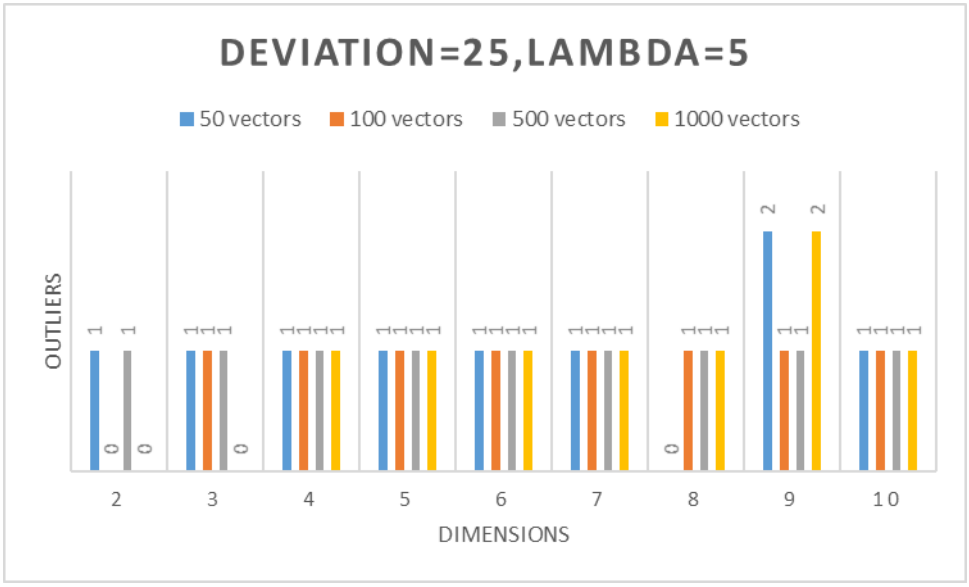

Figure 13: deviation:25,lambda:5 configuration

Deviation=25, lambda=5: The value of lambda has been increased in this case and we can see that even from 2 dimensions' case the Outliers' number is stabilized to 1 more or less. In the case of 9 dimensions for 50 and 1000 vectors 2 Outliers are detected.

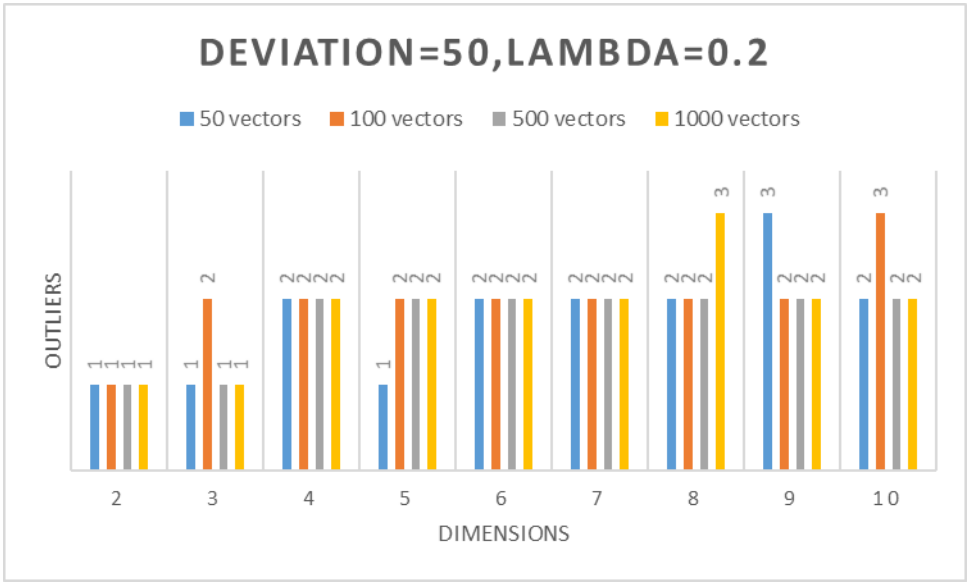

Figure 14: deviation:50,lambda:0.2 configuration

Deviation=50, lambda=0.2: In this case the deviation has been increased from 25 to 50 and it is clear that because the range of the data values is increased also the number of the Outliers has been increased from 1 to 2 on average from the 4 dimensions' case. For 8,9,10 dimensions we see also the number 3 appears.

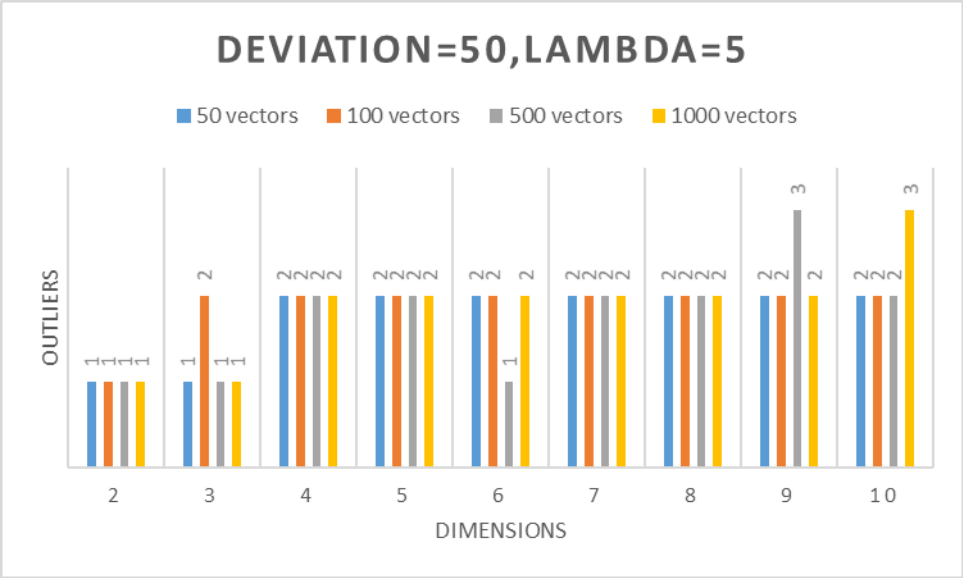

Figure 15: deviation:50,lambda:5 configuration

Deviation=50, lambda=5: In this case the lambda has been increased and we can see that the Outliers' number is similar with the case of lambda=0.2. From 4 dimensions until 10 the number is stabilized to 2. For 8,9,10 dimensions we see also the number 3 appears.

It's really important to mention that the Outliers are detected within the first 10 data vectors that arrive and this make sense because as long as the time window W is getting bigger and bigger the comparison with the timestamps of the stored vectors won't never mark an upcoming vector as an outlier. The factor that affects the number of the Outliers of each case is the distance between their dimensions and it seems that in this case the deviation's value is really crucial, the lambda's value seems to affect less. The number of vectors doesn't seem to affect the number of Outliers so much. Interesting detail is that the maximum number of Outliers we detected in all the cases was 4. This means that the temporal approach of Hilout actually minimizes the Outliers.

In the next section we see how the PCA affects the Outliers' detection.

5.2.2 Results for case with PCA

The display of the charts follows for each of the 4 configurations that are shown in the Table 4 except 2 configurations: 1) deviation:5, lambda:0.2,2) deviation:5, lambda:5. These 2 configurations detect 0 Outliers on average, therefore there is no point to display them in a chart, as for the case without PCA.

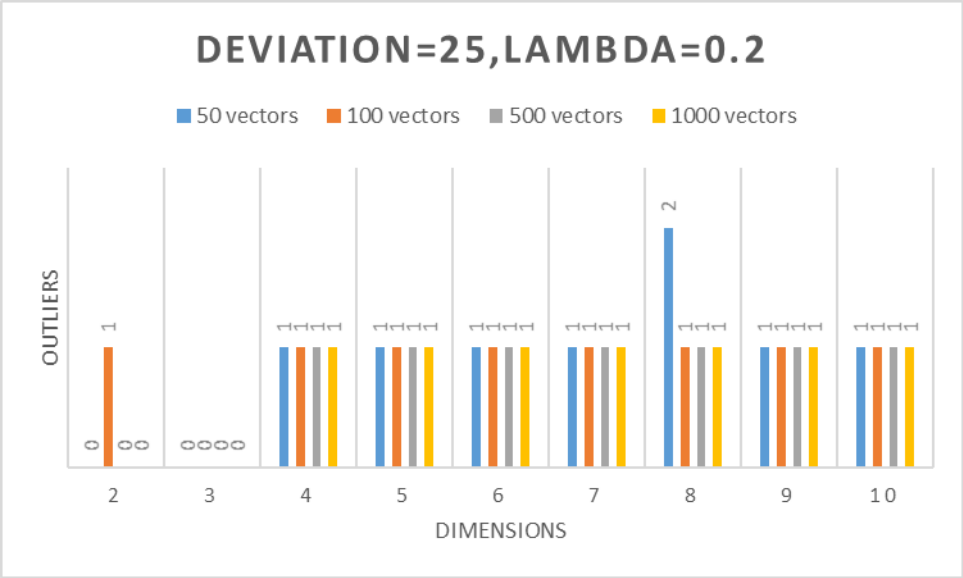

Figure 16: deviation:25,lambda:0.2 configuration

Deviation=25, lambda=0.2: For 2 and 3 dimensions 0 Outliers are detected on average, except the case of 100 vectors for 2 dimensions where 1 outlier is detected. From 4 to 10 dimensions the Outliers'' number is stabilized to 1, except for 50 vectors and 8 dimensions where 2 Outliers are detected.

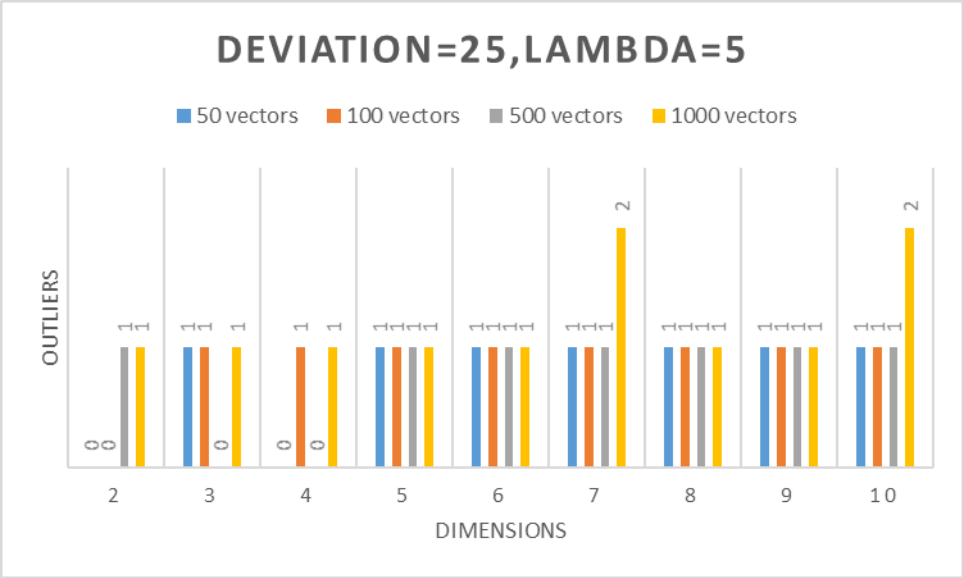

Figure 17: deviation:25,lambda:5 configuration

Deviation=25, lambda=5: The lambda value has been increased and we notice that even from 2 dimensions' Outliers are detected. This time the number of Outliers is stabilized to 1 from 5 dimensions' case. For 7 and 10 dimensions and 1000 vectors 2 Outliers are detected.

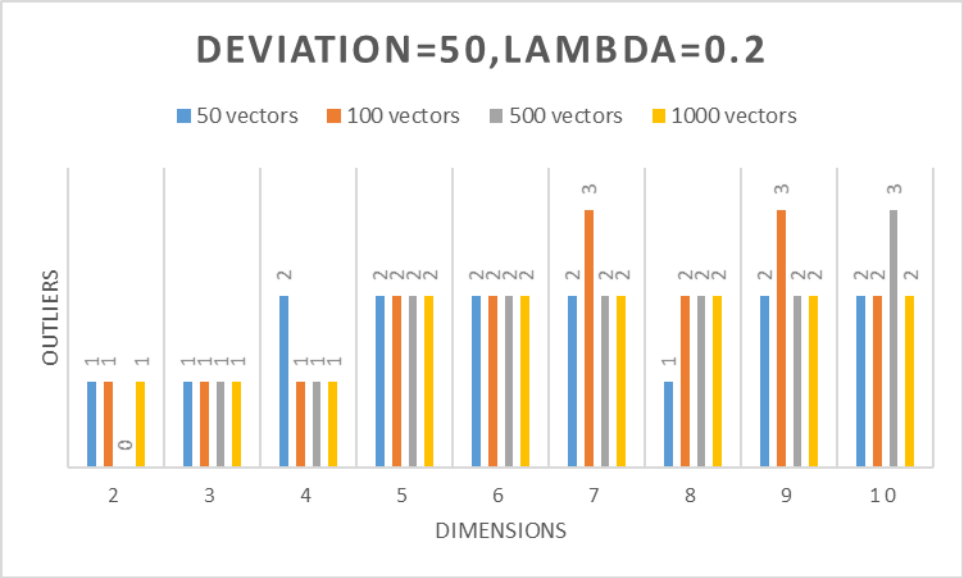

Figure 18: deviation:50,lambda:0.2 configuration

Deviation=50, lambda=0.2: The deviation has been increased from 25 to 50 and we see again the move from 1 outlier to 2 from 5 to 10 dimensions' range. From 2 to 4 dimensions the Outliers' number on average is 1, except the case of 50 vectors for 4 dimensions. For 7 and 8 dimensions, for 100 vectors we see 3 Outliers are detected. Also for 10 dimensions and 500 vectors 3 Outliers are detected.

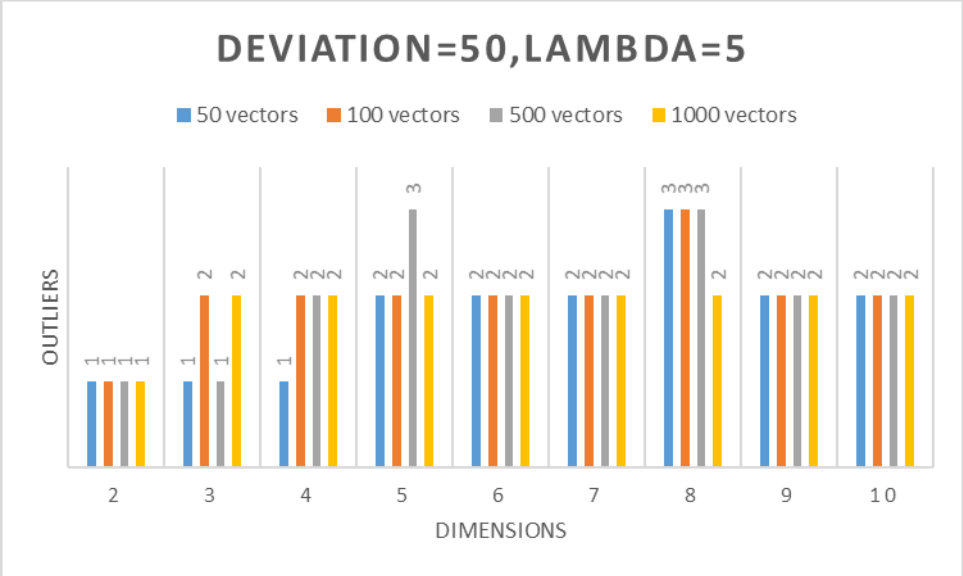

Figure 19: deviation:50,lambda:5 configuration

Deviation=50, lambda=5: In this case, lambda has been increased, 2 Outliers are detected on average from 4 dimensions until 10. For 3 dimensions, for 100 and 1000 vectors, 2 Outliers are detected but for 8 dimensions 3 Outliers are detected on average, except for 1000 vectors.

It's really important to mention that the Outliers are detected within the first 10 data vectors that arrive as in the case without PCA. The number of vectors doesn't seem to affect the number of Outliers so much. Interesting detail is that the maximum number of Outliers we detected in all the cases was 4 as well. In the next section we see the comparison among the results between PCA case and without PCA case.

5.2.3 Comparison between PCA and No PCA cases

In general, we saw that deviation is the most important factor that differentiates the number of Outliers that are detected for all the dimensions. The number of vectors doesn't seem to affect this phenomenon so much. Also the number of dimensions doesn't seem to play an important role. The lambda parameter seems to be affective only on the fact from how many dimensions and afterwards the number of Outliers is stabilized to 1 or 2.

Regarding the PCA factor, before the experiments, we should have expected less Outliers to be detected comparing to the configurations that we ran without PCA. It seems that this is not the case. More or less the results are similar and in fact the peaks of Outliers, 3 Outliers, are detected a bit more for the case of PCA. Still this fact can be interpreted under the next 2 reasons: 1) the PCA runs on the fly along with the Hilout and it is very possible that a lot of times the number of reduced dimensions is not quite smaller than the original one, 2) even if the PCA returns a number much smaller than the number of the actual dimensions each time the range of the values affects the afterwards application of Hilout on the dataset.

5.2.4 Variance per Dimension

In this section we give some indicative results for each dimension for the 6 different configurations of the combination of deviation and lambda and for the 4 different numbers of input data vectors (50,100,500,1000). As average we consider the average Variance for all the dimensions, therefore we have common results for all the dimensions because the range of values that is available every time is for all the dimensions the same. The results are displayed in the tables below:

Table 5: Variance per dimension ,deviation:5,lambda:0.2

Vectors	NoPCA	PCA
50	56.29	53.79
100	85.81	105.2
500	102.45	98.92
1000	107.08	111.02

Table 6: Variance per dimension,deviation:5,lambda:5

Vectors	NoPCA	PCA
50	25.31	24.98
100	23.82	23.92

500	25.71	26.89
1000	25.08	24.96

Table 7: Variance per dimension,deviation;25,lambda:0.2

Vectors	NoPCA	PCA
50	698.92	683.01
100	696.04	685.15
500	760.02	756.88
1000	768.99	762.26

Table 8: Variance per dimension,deviation:25,lambda:5

Vectors	NoPCA	PCA
50	627.71	560.74
100	632.78	644.75
500	700.02	709.12
1000	713.04	699.01

Table 9: Variance per dimension,deviation:50,lambda:0.2

Vectors	NoPCA	PCA
50	2088	2443.45
100	2341	2158.5
500	2224.52	2493.9

1000	2508.02	2359.09

Table 10: Variance per dimension,deviation;50,lambda:5

Vectors	NoPCA	PCA
50	2349.79	1982.15
100	2081.22	2269.1
500	2450.22	2230.89
1000	2368.29	2580.14

In general, we can see that on the contrary with Outliers' numbers, Variance can be differentiated a bit for the number of the vectors. The deviation seems to affect a lot the range of the values each time, we see for deviation=5 values ,on average, lower than 5 but for deviation=25 or 50 the values between 600 and 2600.Lambda doesn't seem to affect the Variance so much except the case of deviation=5 where the values for lambda=5 are lower comparing to the ones with lambda=0.2.PCA and no PCA results are not very different for the same number of vectors and this is normal because PCA doesn't affect the calculation of the Variance for each dimension. We could have excluded the results for PCA but we put them for statistical reasons.

6. Conclusions

In this Thesis we saw in theory the concepts of Cloud Computing, Mobile Computing, the definitions of a Cloud, of a Cloudlet and the concept of cooperative Caching of the nodes in a Cloudlet but practically we implemented the Hilout algorithm on data vectors that try to enter a Cloudlet in a simulation level in order to provide an intelligent scheme that detect Outliers. For this purpose, we used the CloudSim framework.

In our scenario the input data vectors include coordinates from IoT devices, that can be within the range of 2 to 10, and the values of these coordinates-dimensions are given by the Gaussian distribution that every 10 vectors its mean deviates by a factor that is produced by the Exponential distribution. The scheme we applied for the Outliers' detection uses the Hilout Algorithm with a temporal approach. Not only the distance between the stored vectors and the incoming one is taken into account for the decision to let it enter the Cloudlet but also a shifting time window. Each time the algorithm compares the timestamps of the vectors that are considered as neighbors with the timestamps of this window and adds accordingly a weight value that is taken into consideration plus the distances in order to decide if the incoming vector is an outlier or not.

In addition, we saw some theoretical information regarding the Gaussian and Exponential Distribution but also we included in our scenario the case of PCA dimensions' reduction. PCA dimension reduction was used in an alternate implementation of the Hilout in order to diagnose if this feature is going to differentiate the results comparing to the simple case.

Last but not least, we ran our experiments based on two main cases: Hilout applied without PCA, Hilout applied with PCA. The configurations of the dataset were: for 2-10 dimensions, for deviation parameter of the formula of Gaussian: 5,25,50 and for lambda parameter of the formula of Exponential: 0.2,5. The results showed that the most important factor is the deviation parameter and more Outliers are detected on average while it increases. The case with PCA gave more or less similar results with the simple case. In our results we provided also the Variance per dimension in order to provide some statistical conclusions regarding the different configurations.

ABBREVIATIONS - ACRONYMS

IaaS	Infrastructure as Service
PaaS	Platform as Service
SaaS	Software as Service
MaaS	Monitoring as a service
QoS	Quality of Service
MCC	Mobile Cloud Computing
AAA	Authentication,Authorization,Accounting
SOAP	Simple Object Access Protocol
DIC	Data Intensive Computing
IoT	Internet of Things
OCR	Optical Character Recognition
VM	Virtual Machine
PCA	Principal Component Analysis

REFERENCES

[1] Mobile Cloud Computing by Lalit Kumar, Nishant Malik, Gourav Agghi, Ajay Anand
[2] Mobile Cloud Computing: A Tool for Future by Dr. Atul Gonsai and Mr. Rushi Raval
[3] Mobile Cloud Computing - IEEE COMSOC MMTC E-Letter by Dijiang Huang
[4] MOBILE CLOUD COMPUTING AS FUTURE FOR MOBILE APPLICATIONS by C Shravanthi, H S Guruprasad
[5] Mobile Cloud Computing: Implications and Challenges by M. Rajendra Prasad Jayadev Gyani P.R.K. Murti
[6] A Review on Mobile Cloud Computing by S M Shamim, Angona Sarker, Ali Newaz Bahar and Md. Atiqur Rahman
[7] A Mobile Cloud Computing Architecture with Easy Resource Sharing by Debabrata Sarddar and Rajesh Bose
[8] The Case for VM-Based Cloudlets in Mobile Computing by Mahadev Satyanarayanan, Paramvir Bahl, Ramón Cáceres and Nigel Davies
[9] Cloudlets: Bringing the cloud to the mobile user by Tim Verbelen, Pieter Simoens, Filip De Turck and Bart Dhoedt
[10] Supporting Cooperative Caching in Ad Hoc Networks by Liangzhong Yin and Guohong Cao
[11] Cooperative Caching Framework for Mobile Cloud Computing by Preetha Theresa Joy & K. Poulose Jacob
[12] Mobile Cloud Computing: A Comparison of Application Models by Dejan Kovachev, Yiwei Cao and Ralf Klamma
[13] Mobile cloud computing: A survey by Niroshinie Fernando, Seng W. Loke and Wenny Rahayu
[14] Central Controller Framework for Mobile Cloud Computing by Debabrata Sarddar1, Priyajit Sen2 and Manas Kumar Sanyal
[15] Implementation and Evaluation of Mobile-Edge Computing Cooperative Caching – Master Thesis by Morteza Neishaboori for Aalto University School of Science Master's Programme in ICT Innovation
[16] https://en.wikipedia.org/wiki/Dimensionality_reduction
[17] https://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/
[18] DATA MINING Concepts and Techniques (3rd edition) by Jiawei Han, Micheline Kamber, Jian Pei
[19] https://www.itl.nist.gov/div898/handbook/eda/section3/eda3661.htm
[20] http://hyperphysics.phy-astr.gsu.edu/hbase/Math/gaufcn.html
[21] https://www.britannica.com/topic/normal-distribution
[22] https://stats.stackexchange.com/questions/126351/reasons-for-data-to-be-normally-distributed
[23] http://www.statisticshowto.com/exponential-distribution/#poisson
[24] https://en.wikipedia.org/wiki/Quantile_function
[25] https://en.wikipedia.org/wiki/Exponential_distribution
[26] https://www.ck12.org/statistics/Variance-of-a-Data-Set/lesson/Calculating-Variance-PST/
[27] http://www.cloudbus.org/cloudsim/