Imperial College London

Department of Computing

Wiki-Health: From Quantified Self to Self-Understanding

Yang Li

Submitted in fulfilment of the requirements for the degree of Doctor of Philosophy in Computing of Imperial College London and the Diploma of Imperial College London

December 2014

Declaration

The material presented in this thesis entitled — "Wiki-Health: From Quantified Self to Self-Understanding" is the author's own work, except specifically acknowledged in the text.

Yang Li, December 2014

Abstract

Today, healthcare providers are experiencing explosive growth in data, and medical imaging represents a significant portion of that data. Meanwhile, the pervasive use of mobile phones and the rising adoption of sensing devices, enabling people to collect data independently at any time or place is leading to a torrent of sensor data. The scale and richness of the sensor data currently being collected and analysed is rapidly growing. The key challenges that we will be facing are how to effectively manage and make use of this abundance of easily-generated and diverse health data.

This thesis investigates the challenges posed by the explosive growth of available healthcare data and proposes a number of potential solutions to the problem. As a result, a big data service platform, named Wiki-Health, is presented to provide a unified solution for collecting, storing, tagging, retrieving, searching and analysing personal health sensor data. Additionally, it allows users to reuse and remix data, along with analysis results and analysis models, to make health-related knowledge discovery more available to individual users on a massive scale.

To tackle the challenge of efficiently managing the high volume and diversity of big data, Wiki-Health introduces a hybrid data storage approach capable of storing structured, semi-structured and unstructured sensor data and sensor metadata separately. A multi-tier cloud storage system—CACSS has been developed and serves as a component for the Wiki-Health platform, allowing it to manage the storage of unstructured data and semi-structured data, such as medical imaging files. CACSS has enabled comprehensive features such as global data de-duplication, performance-awareness and data caching services. The design of such a hybrid approach allows Wiki-Health to potentially handle heterogeneous formats of sensor data.

To evaluate the proposed approach, we have developed an ECG-based health monitoring service and a virtual sensing service on top of the Wiki-Health platform. The two services demonstrate the feasibility and potential of using the Wiki-Health framework to enable better utilisation and comprehension of the vast amounts of sensor data available from different sources, and both show significant potential for real-world applications.

Graphical Abstract

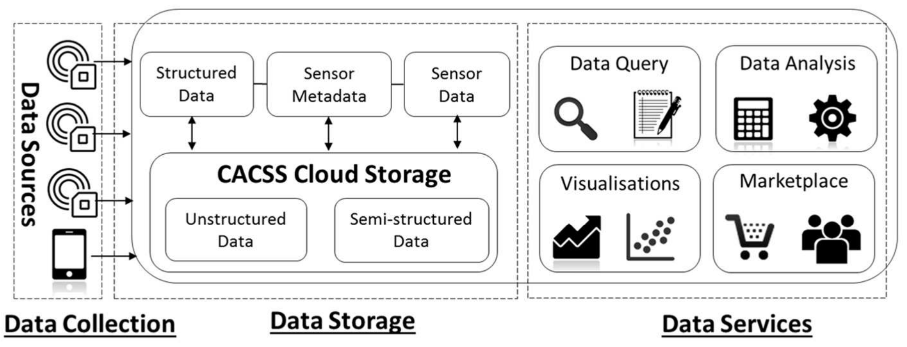
The Challenge

Explosion of Health Data and Sensor Devices

DOC
PDF
JPG
Unstructured Data
Time-Series Sensor Data
Wireless Sensor Devices
Models and Knowledge

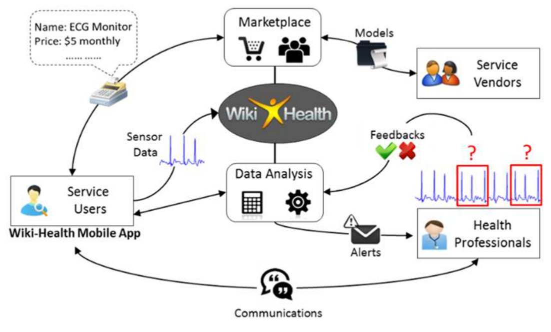
The Solution

Wiki-Health Platform

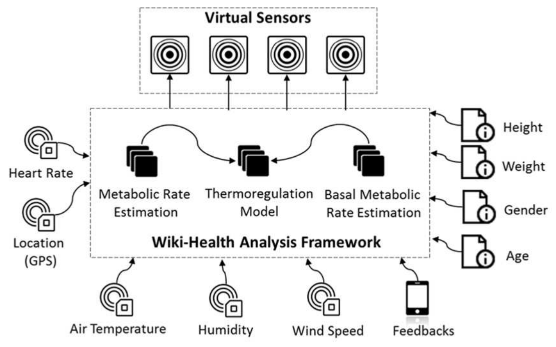


The Evidence

Health Monitoring as a Service



Virtual Sensing as a Service



Acknowledgements

I would like to thank my supervisor Professor Yike Guo for his immense knowledge, tremendous enthusiasm, ispiring guidance, endless encouragement, invaluably constuctive criticism and advice to both my personal and academic development throughout the years of my PhD study.

I would also like to thank Dr. Li Guo and Dr. Chao Wu for their patience, support, helpful suggestions and all the time they have contributed.

Many thanks to all my friends and colleagues at Imperial College London, Prof. Moustafa Ghanem, Dr. Sijin He, Dr. Rui Han, Dr. Ioannis Pandis, Dr. Dilshan Silva, Dr. David Birch, Dr. Kai Sun, Mr. Orestis Tsinalis, Ms. Shulin Yan, Mr.Florian Guitton, Ms. Xian Yang, Mr. Lei Nie, Mr. Akara Supratak, Mr. Shicai Wang, and Ms. Diana O'Malley. They have provided me much help and brought much happiness to my life at Imperial.

Finally, and most importantly, I woud like to thank my parents, my wife Ting Zhang, and all the rest of my family. Their generous love and endless support helped me through the ups and downs of my PhD work.

Dedication

"The rise of Google, the rise of Facebook, the rise of Apple, I think are proof that there is a place for computer science as something that solves problems that people face every day."

Eric Emerson Schmidt

Contents

List of Figures

List of Tables

1. Introduction

1.1. Motivation and Objectives

Healthcare providers are constantly generating large volumes of data. According to industry estimates [1], worldwide healthcare data is expected to grow from 500 petabytes in 2012 to 25,000 petabytes by 2020. Such immense data ranges from complex and high-resolution medical imaging data like X-rays and CT scans to the simple numerical data of systolic and diastolic pressure measurement figures from blood pressure devices. The means by which health care information can be gathered are also evolving. The growing global popularity of smartphones and tablets has led to the development of new ways of gathering information, both manually and automatically, by means of an array of embedded sensors. Professional wearable biosensors can connect to smart phones and track a significant number of physiological parameters. This combination of technologies has provided a more efficient and convenient way to collect a wide array of personal health information such as blood pressure levels, oxygen saturation levels, blood glucose measurements and pulse rates, as well as electrocardiogram (ECG), electroencephalogram (EEG) and electrocardiography (EKG) data. These different data formats used in the collection of healthcare data can be generally classified into three types: unstructured, structured and semi-structured. Examples of unstructured data include medical images, videos, plain text files and PDFs. Most types of structured sensor data generated from healthcare devices such as blood pressure and ECG, can be represented in the form of time-series.

With the introduction of increasingly sophisticated new technologies, the scale and richness of mobile sensor data being collected and analysed is rapidly growing. However, there are still many challenges to face. For example, it is still difficult for users to manage or utilise the data they collect for different purposes. From the provider's perspective, such massive growth in the volume of big health sensor data creates both data manageability and collaboration challenges. Traditional sensor management systems such as TinyDB [2], Aurora [3], and COUGAR [4] offer limited support for collaboration restricted data processing capacities; they are no longer able to cope with the demands created by the pervasive use of mobile devices and the rising adoption of sensing devices. The emergence of cloud computing is seen as a remedy to these issues.

Cloud computing has been widely discussed in the past few years, as it shows great potential to shape the development, maintenance and distribution of both computing hardware and software resources. Within this computing paradigm, the actual provision of resources is only a concern at run-time for specific application requirements, as is also the case for necessary software resources, as they can also be accessed in an on-demand and pay-per-use fashion. Cloud storage takes sharing hardware one step further: unlike local storage, cloud storage relieves end users of the task of upgrading their storage devices constantly. Cloud storage services enable inexpensive, secure, fast, reliable, and highly scalable data storage solutions that can be provided and easily accessed over the Internet. Many enterprises and personal users—especially but not exclusively those with low budgets and limited IT resources are now outsourcing storage to cloud storage service providers in an attempt to leverage the manifold benefits associated with cloud services. Leading cloud storage vendors, including Amazon [5] and Google [6], provide clients with highly available, low cost, and pay-as-you-go cloud storage services with no upfront cost. Companies in a variety of sectors now outsource at least a portion of their storage infrastructure to the cloud storage service providers. Even so, many enterprises and scientists are still unable to shift into the cloud environment due to privacy, data protection and vendor lock-in issues. An Amazon S3 storage service outage in 2008 left many businesses reliant on the service offline for several hours and resulted in the permanent loss of customer data [7, 8]—an incident that led many to question the S3's "secret" architecture.

Implementing cloud computing technologies appropriately can aid healthcare providers in improving the quality of medical services and the efficiency of operations, sharing information, improving collaboration, and managing expenditures. For example HealthVault, is a web-based platform developed by Microsoft to store and manage health and fitness information, also provides storage services for unstructured data such as medical images. Other commercial sensor network platforms such as Xively [9] and ThingSpeak [10] have taken off in recent years. They provide online scalable sensor data management platforms that allow users to connect sensor devices and applications through a web-based application interface (API). However, none of these solutions yet provide sufficient support for running computationally intensive analysis algorithms on their platforms.

Being able to monitor users' bio-signals over the long term shows a great potential to allow us to understand their typical lifestyles and behaviours. More importantly, it allows for the tracking and discovery of any change-signals that could serve as warning signs of potential health issues. Extensive research has been undertaken on various biology and physiological models. However, many of these models require inputs and parameters that still cannot be measured directly by sensors. Fusing and integrating data generated from different sensors requires considerable domain knowledge and experience. As more people are becoming interested in monitoring their health and researching their own data, there is a clear need for an infrastructure on which scientists, developers, and professionals can publish their developed data analysis models as utilities, thereby enabling other users to access those services and utilise their collected sensor data without any expert knowledge.

The principle aim of the thesis is to address the aforementioned challenges by presenting the design and implementation of a platform named "Wiki-Health", which will take advantage of cloud computing for personal health sensor data management. Wiki-Health is not only designed to solve the problems of managing and storing the increased volume, velocity and variety of data, but also to provide support tools for users to create healthcare-related applications and analysis models. As a result, the platform will allow them to effectively utilise all of the data generated from different data sources while reducing the complexity of dealing with its diversity. The functionalities and associated tools of Wiki-Health make it feasible for researchers to use as a test bed for undertaking health-related experiments and delivering novel analysis algorithms.

1.2. Challenges

In this section, we explore and summarise the key challenges to designing and developing a big data service platform for managing health sensor data. While designing efficient, scalable and secure mechanisms for indexing, searching, storing and handling heterogeneous data is a vital and necessary goal, implementing a system capable of handling this kind of workload presents a variety of obstacles, including:

I. Challenges in terms of managing the explosive data growth emerging from modern technologies such as medical imaging, pervasive use of sensor devices and personal whole genome sequencing. There are many large-scale storage systems and sensor data management systems in existence today. However, most of these systems have not been designed to store and hold heterogeneous data types from different sources. Thus, many problems exist with respect to large-scale storage and management solutions: i) these systems can suffer from different levels of bottlenecking at high concurrent access rates depending on the type of the data is stored and accessed [11]; ii) they can lose the data in a hardware failure or a disaster scenario if data is not sufficiently well replicated and iii) they are vulnerable to very common events, such as end user error, in which data may be unintentionally deleted or overwritten. This demands that large-scale data management systems be equipped with sophisticated replication, versioning and recovery mechanisms to restore data.

II. Challenges in storing, managing, and linking user-defined, algorithm-related, and other information that would add further dimensions to the stored sensor data and help identify the correct event, stimulus or cause associated with a corresponding sequence of sensor data.

III. Obstacles between the users who collect the data and health professionals who provide data analysis services. There are many scientists and health professionals who have developed many algorithms and models designed to help patients monitor their disease progression, and detect and diagnose potential health issues through the analysis of different types of data. However, most of the time, only model designers or experts understand how to use these models, how to choose what types of data they need and how to interpret what the results mean. In another aspect, these model developers used to focus on a limited number, sometimes a small group of patients, and use the data from these patients to design algorithms and models. This can potentially be a biased approach that the models may not work properly on patients that were not presented in the originally designing group. Today, with modern technologies, we can now gather about our bodies through different sensor devices and data sources from anywhere. Other challenges have also emerged in terms of how we can utilise the knowledge (models) that we currently have to make sense of the information that we can now gather to provide meaningful results to the users in an easy and convenient way. Another aspect of this question is how we can enable an ecosystem to ensure that both parties, model developers and model users (data owners), are motivated and can benefit both by contributing the models, using the models and providing collected data and analysed results.

IV. Absence of sufficient support for individual self-monitoring and self-understanding. People are now becoming very interested in tracking many day-to-day health parameters and sharing them with their families, friends and other people with common needs or condition. This has posed a number of challenges in terms of providing the support for managing the lifecycles and processes of dealing with health sensor data such as data collection, data query and search, data sharing and data analysis in an efficient and secure way.

V. Problems inherent to data collection and handling specific to the healthcare field. For example, health sensor devices such as Electrocardiogram (ECG) monitors employ a number of data channels and are capable of generating huge quantities of data. Consequently, visual inspection and interpretation of data created on such massive scale is extremely difficult. Monitoring and analysing human vital signs such as core body temperature, blood pressure and heart rates can enable us to trigger alerts and reduce the risks for people under extreme physical and health conditions. However, placing a physical sensor at the ideal location to acquire the necessary sensing information and analysing the incoming sensor data to obtain real-time results and feedback is still challenging; Analysing large medical imaging data such as CT and MRI scans requires a great deal of processing power and may require large amounts of data to be moved around in a distributed environment, This raises the question of how we can scale these analysis tasks and utilise the available computational resources such as CPU, RAM and bandwidth to maximise the efficiency.

In summary, these issues illustrate the types of challenges we would need to tackle in transforming and integrating models into services for end users, synchronising and combining sensor data collected from various types of devices, and performing real-time analysis on large volumes of data.

1.3. Contributions

The main contributions of this thesis are the followings:

- We present a health sensor data management platform named Wiki-Health, which allows users not only to collect, store, annotate, update and share sensor data, but also to create and remix a variety of data analysis results and models.
- We present a detailed design for the proposed hybrid data storage approach of Wiki-Health which takes advantages of cloud storage and features of both relational and non-relational databases. The ability to manage unstructured, semi-structured, structured sensor data, and sensor metadata separately has enabled Wiki-Health to efficiently handle any type and format of sensor data.
- We present a multi-tier cloud storage system (CACSS) for managing unstructured and semi-structured data for healthcare applications as one of the system component of Wiki-Health. CACSS is based on the generic principles of scalability, performance, data durability and reliability.
- We present a health monitoring model and service on top of Wiki-Health, which is able to detect personal ECG beat abnormalities and update the model based on incoming feedback. The proposed Adaptive Learning Approach (ALA) within the analysis model aims to reduce the training time necessary to update the model, while showing improved performance over existing methods.
- We present a virtual sensing service on top of Wiki-Health, which utilises the concept of virtual sensors and functionalities of Wiki-Health to compute and simulate human body temperatures using indirect sensor measurements. The proposed approach demonstrate the feasibility and potential of using the Wiki-Health framework to better utilise and analyse the vast amounts of sensor data from different sources.

1.4. Thesis Structure

- Chapter 1: introduction to the problem and an overview of the thesis.
- Chapter 2: background information on health data acquisition modalities and challenges, existing technologies for big data management, current data analysis tools, and a survey of current applications for enabling self-understanding and proactive health management.
- Chapter 3: discusses how cloud storage can be used as a part of the solution for big health data management and presents a CACSS cloud system for managing unstructured health data such as medical images.
- Chapter 4: describes the design and implementation of the Wiki-Health platform, which incorporates CACSS cloud storage system to provide a unified solution for managing health sensor data and an ecosystem to enable analysis models as services.
- Chapter 5: illustrates the analysis model and key components for enabling long-term ECG-based health monitoring as a service in the Wiki-Health platform for medical use.
- Chapter 6: presents the design rationale of a virtual sensing service application deployed in the Wiki-Health platform to demonstrate the feasibility of integrating human physiological models with health sensor data for health-related knowledge discovery.
- Chapter 7: conclusions are drawn and some directions for future work are discussed.

1.5. Statement of Originality

I, Yang Li declare that this thesis is entirely own work by myself, except where acknowledge in the text and references are given in the bibliography.

1.6. Publications

The research described in this thesis has been published as follows. I am the first author of the publications marked with *.

CACSS Cloud Storage System Architecture

“*CACSS: Towards a Generic Cloud Storage Service”. In the proceeding of 2nd International Conference on Cloud Computing and Services Science (CLOSER 2012), Porto, Portugal, 2012.

“*An Efficient and Performance-Aware Big Data Storage System” *. In the book of Cloud Computing and Services Science by Springer, 2013.

"*Enabling Performance as a Service for a Cloud Storage System"*. In the proceeding of 7th IEEE International Conference on Cloud Computing (IEEE Cloud 2014), Alaska, USA, 2014.

Wiki-Health Platform Architecture

“*Building a Cloud-based Platform for Personal Health Sensor Data Management”. In the proceeding of IEEE BHI 2014 Conference Theme of “Translating key health challenges with advances in biomedical informatics”, Valencia, Spain, 2014.

“*WikiHealth-A Big Data Platform for Health Sensor Data Management”. In Cloud Computing Applications for Quality Health Care Delivery, Advances in Medical Technologies and Clinical Practice (AMTCP) Book Series, 2014.

Wiki-Health Health Data Analysis Applications

“*Enabling Health Monitoring as a Service in the Cloud”. In the proceeding of 7th IEEE/ACM International Conference on Utility and Cloud Computing, London, United Kingdom, 2014.

“*Wiki-Health: A Platform for Enabling Virtual Sensing as a Service”. In the journal of Computer Methods and Programs in Biomedicine, Elsevier, submitted.

2. Background

We are living in a community where we are constantly capturing information about ourselves and around the environments we inhabit, in both active and passive ways. Everything we do and everything that happens in our universe today will leave digital trace and data, such as photos, videos, emails, texts, social network feeds, sensor data and log files. Technological advances have enabled us to digitise new types of information; capture higher dimension and more accurate data, and easily collect data about ourselves independently. More importantly, we are keen on understanding ourselves.

In this chapter, we first present a background survey of the current health data acquisition modalities and challenges in terms of explosive data growth emerged from modern technologies such as medical imaging, pervasive use of sensor devices and personal whole genome sequencing. Secondly, we explore the existing technologies as solutions for these challenges of managing and understanding big health data. Lastly, we review the overall landscape of health information impacted by new technologies, such as wearable and wireless sensors, cloud computing and Internet of Things on healthcare.

2.1. Big Health Data Acquisition

2.1.1. Medical Imaging Technologies

Medical imaging technologies are used to create visual representations of body parts, tissues or organs. Medical imaging provides useful information for clinical diagnose, analysis, treatment and disease monitoring for a wide array of conditions. Various medical imaging technologies have been developed to gather diagnostic information for a range of different health issues, such as X-rays, mammograms, Computed Tomography (CT) scans, Magnetic

Resonance Imaging (MRI) scans, ultrasounds, and Positron Emission Tomography (PET) scans. Medical imaging is a significant diagnostic cornerstone of modern healthcare and will continue to play a vital role on many levels of healthcare systems.

X-ray

X-rays are the oldest and most widely available form of medical imaging. X-rays are created by sending beams of ionising radiation through the body and recording the resulting image on a film or sensor plate. Different amounts of radiations are absorbed by different body tissues and structures, depending on the density of the X-rays pass through. X-ray images are frequently used to evaluate damage or disease in bones, lungs, and cavities. The original photographic films used to create X-rays are being replaced by modern X-ray sensors which allow X-ray images to be easily recorded and stored digitally.

Computed Tomography (CT)

CT (formerly known as computerised axial tomography or CAT) is a more recent imaging tool that collects combines multiple X-ray projections taken from different angles to create more detailed images of the inside of the body than with standard X-ray images. The CT scanner consists of an X-ray tube that rotates around the patient being scanned, collecting a multitude of images of the targeted region with each pass. CT images provide doctors with precise, three-dimensional views of certain parts of the body and are useful for diagnosing cancers.

Magnetic Resonance Imaging (MRI) and Functional MRI (fMRI)

Unlike X-ray or CT scans, MRI is a radiation-free medical imaging technology that produces cross-sectional images of the human body. MRI scanners use strong magnetic fields and radio waves to create detailed images of organs and other structures inside the body. MRI generates more highly detailed images of soft tissue than are possible with X-ray or CT scans, making this technology useful to evaluate problems such as abnormal tissues, ligament and tendon injuries, and brain tumours. fMRI is a procedure based on MRI technology for measuring brain activity through changes in blood oxygenation and flow.

Ultrasound

Ultrasound scans use high-frequency sound waves to build up images of organs and structures inside the body. The images are produced when sound waves are transmitted into the body and reflected back to a scanner that measures them. Ultrasound scanning is completely painless and there are no serious side-effects associated with it, since it does not use radiation. It has been widely applied to detect heart abnormalities (echocardiogram), examine organs such as the liver, kidney and bladder, and is routinely used to monitor pregnancies because it allows physicians to safely observe fetal growth and development.

Positron Emission Tomography (PET)

PET scans use radioactive materials (radiotracers) to produce a three-dimensional image of functional processes in the body. As the radiotracer move through the body, it will generate a stream of positrons. The PET scanner is designed to detect gamma waves released by these positrons and build up images that reveal the size, shape, position and functions of organs. PET scans are often used to diagnose conditions that affect the brain and nervous system, to detect range of different cancers and to determine how far cancers have spread.

PET-CT and PET-MRI

PET-CT is a medical imaging technique that combines PET and CT scan into one scan to produce more precise anatomic localisation than functional imaging alone. The PET and CT data are fused (co-registered). Most recently, PET-MRI has been developed. Similar to the PET-CT technology, PET-MRI is a hybrid imaging technique that incorporates both MRI and PET imaging. Both types of multimodality imaging approaches have shown great promise for clinical applications.

2.1.2. Medical Imaging File Formats

Medical imaging files are used to store various kinds of information needed for reconstructing the numerical values to positions of the space and support data compatibility between different imaging systems, including pixel data, pixel depth, metadata and photometric interpretation [12]. There are two types of medical image file formats: one is focused on persisting more information of the patient for clinical domain such as Dicom [13], the other one is designed to help the process of data analysis for research domain, such as Analyze [14], Minc [15], and Nifti [16]. For non-diagnostic images JPEG format has also been widely used. Almost all of these medical imaging files belong to the type of unstructured data.

2.1.3. Era of Big Data Challenge for Medical Imaging

Over the past decade, the amount of medical imaging data generated within the healthcare sector worldwide has undergone rapid growth. A big data revolution is under way in health care [17]. Today, the average storage space required for a single study is approximately 100MB per procedure in the United States. The aforementioned medical imaging modalities such as CT, MRI, 3D/4D ultrasound scans can generate hundreds to thousands of images and occupy more than 200MB storage space per study. More advanced imaging technologies such as PET-CT and PET-MRI are able to generate more than 30 images per second. For an exam requiring just a few minutes to complete, these types of scans can generate more than 1GB of data. Storage requirements are estimated to be tripling in size every four to five years[18, 19]. Many factors have influenced this exponential increase in medical image storage, such as improved medical imaging technologies, an aging population, the dramatic increase in file sharing across different endpoint devices and the use of collaboration tools, which also generate large amounts of data duplication.

As shown in Table 2.1 [20], there were more than 40 million medical imaging examinations performed in the England's National Health Service (NHS) in 2012-2013. Figure 2.1 [20] shows the number of different medical imaging modalities from 1995-96 to 2012-13. Over the last 10-year period, the number of MRI and CT scans has increased by 211%, and 167% respectively, while the total number of examinations performed has increased by approximately 39%, representing an average growth of 3.3% per year. According to industry estimates [21], worldwide healthcare data is expected to grow from 500 petabytes in 2012 to 25,000 petabytes by 2020. Most of these medical images consist of unstructured data and are rarely tagged that are useful for data mining and analysis tasks. The enormous volume of unstructured data generated by the millions of medical images produced each year is increasingly challenging our ability to store, manage, share, and analyse such images.

Modality	*2011-12*	*2012-13*	*Percentage growth in last year*	*Average growth per year since 2002-03*
X-Rays	22,485	22,640	0.7%	1.5%
Ultrasound	9,054	9,324	3.0%	5.2%
CT	4,381	4,726	7.9%	10.3%
MRI	2,299	2,447	6.5%	12.0%
Fluoroscopy	1,347	1,315	-2.4%	0.2%
Radio-isotopes	614	599	-2.4%	0.8%
Total	40,181	41,052	2.2%	3.3%

Table 2.1: Number of tests (000s) by imaging modality 2011-12 and 2012-13.

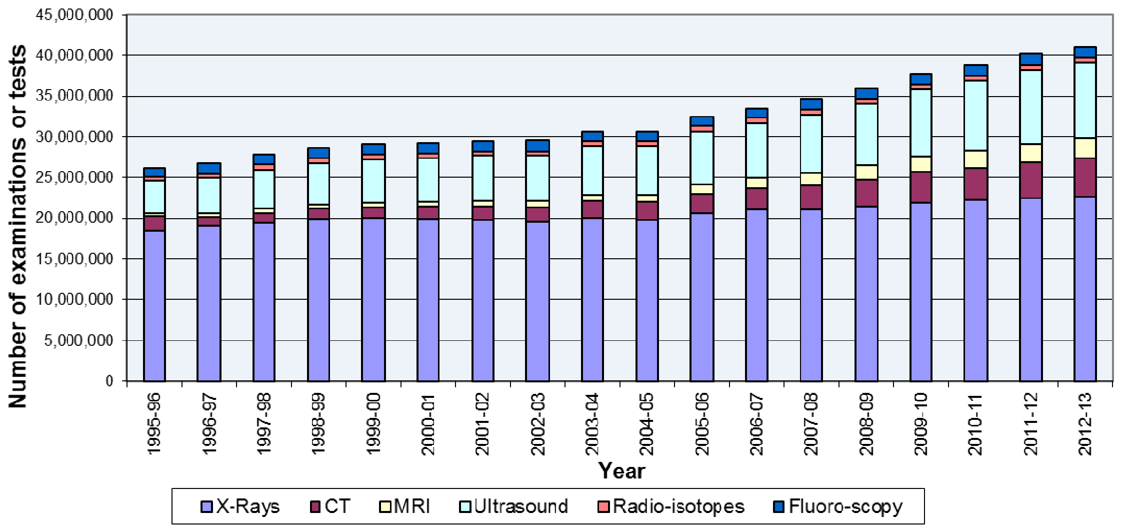


Figure 2.1 [20]: Total number of imaging and radiodiagnostic examinations or tests, by imaging modality, England, 1995-96 to 2012-13

While medical scans may be vital in diagnosing many conditions, medical imaging tests such as CT scans also lead to radiation exposure which may increase the risk of developing cancers. In addition, patients sometimes undergo repeat scans for various reasons, such as incompatibility of imaging software, or limited communications and data sharing capabilities between different health organisations. In the case of scans involving radiation, improving the ability collect and track information on radiation exposure for individual patients as well as improving the data collaboration between health organisations could potentially reduce the number of unnecessary repeated scans.

2.1.4. Time Series based Clinical Measurements

Electrocardiogram (ECG)

ECG is the recording of the electrical activity of the heart. It is a non-invasive procedure and one of the most widely used physiological signals for medical diagnoses.

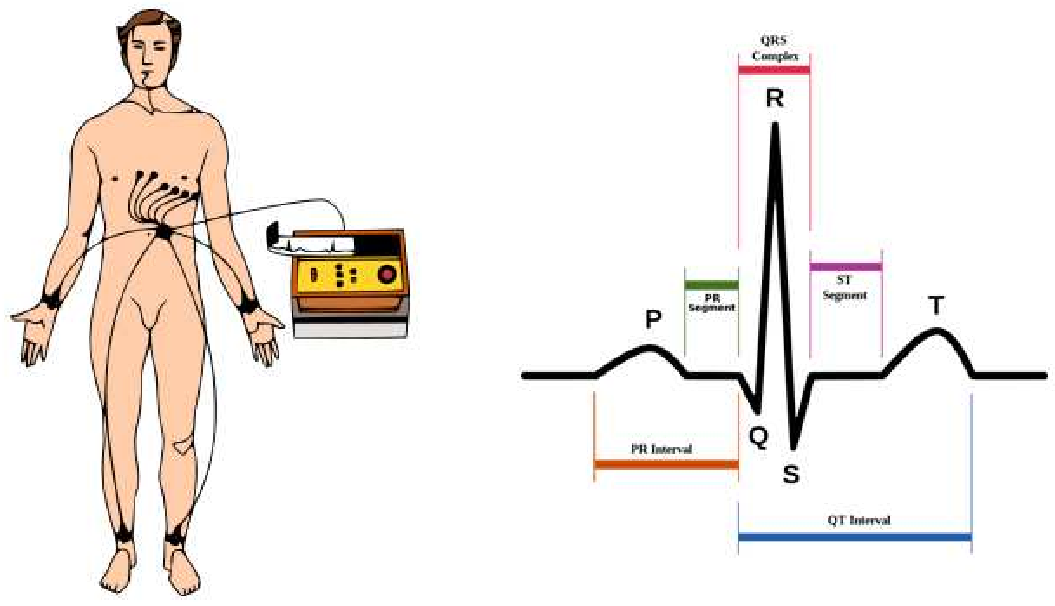


Figure 2.2: Image on the left shows a patient connected to a 12-lead ECG. Image on the right shows the schematic representation of normal ECG.

Traditionally, a standard ECG screening records the electrical activity of your heart for only a few minutes and the procedure is carried out in a hospital (as shown in Figure 2.2). When symptoms persist and a definitive diagnosis cannot be determined using a standard ECG, a Holter monitor—a small, battery-operated wearable device that tracks the patient's heart rhythms—will be used to record 24 to 48 hours of ECG signals. Patients then return the monitors to their doctors. Efforts are being made to enable remote monitoring of such ECG-monitoring systems. For example, the AliveCor device [22, 23] is one of a growing number of health sensors that not only allow users to record and view medical grade single-channel ECG signals on a smartphone/tablet, but also to upload and share their recordings directly with their physicians or cardiologists through the internet.

The QRS complex (as shown in the Figure 2.2) is one of the most key information components extracted from a patient's overall ECG signals to help doctors in diagnosing and tracking the change. The QRS complex is a specific sequence of deflections seen on a printout of typical ECG signals. It represents the depolarization of the right and left ventricles of the human heart.

Normally the heart beats in a regular and rhythmic way producing a P wave, a QRS complex and a T wave. These waves are useful in helping doctors and health professionals diagnose cardiovascular problems, identify abnormal levels of specific minerals in the blood, and observe the effects of drugs used to regulate the heart.

Electroencephalogram (EEG)

EEG is the recording of electric activity of the brain at the time the test is carried out, providing a way of measuring the state of the brain and of specific brain functions. EEG data can be useful for many applications ranging from clinical use to research, and even the development of games. The EEG's clinical uses include diagnosing neurological disorders such as seizures, and evaluating problems associated with coma, tumours and memory loss. In the research field, EEG has been widely used in psychophysiological research, neuroscience and cognitive science. An example of one current research direction is the application with the functional magnetic resonance imaging (fMRI) technique in which synchronously acquired EEGs are used to characterise brain activities across time and study associated haemodynamic (flow of blood within the organs and tissues of the body) changes [24-26]. EEG can also be used to develop brain training games for ADHD patients to improve their ability to focus [27, 28].

Core Body Temperature

Human core body temperature is one of the vital signs to measure the body's most basic functions. Too high or too low core body temperature can cause conditions such as hyperthermia and hypothermia. Being able to monitor human core body temperature can provide a way of reducing the risk for many occupations such as construction builders, fire fighters, transport drivers, athletes, refrigerated warehouse workers, and military crews. Many available medical thermometers accurately measure the temperature at different external positions of the body. However, measuring the most accurate internal core body temperature is only possible by an invasive method, such as by placing a temperature probe into the oesophagus, pulmonary artery or urinary bladder [29-31]. In this context, a number of studies have been performed from different aspects, ranging from modelling human thermoregulation [32, 33] to designing and evaluating non-invasive thermometers [34-36].

2.1.5. Wireless Sensors

The rapidly growing popularity of smartphones and tablets globally has created many opportunities for growth within the health care and wellbeing sector. These devices provide new ways to gather information, both manually and automatically, over wide areas. Many current smart phones come with a number of embedded sensors such as microphones, cameras, gyroscopes, accelerometers, compasses, proximity sensors, GPS and ambient light detectors. Professional wearable biosensors can easily connect to smart phones and can track a significant number of physiological parameters. Existing wearable fitness sensors such as the Fitbit trackers, [37], Zephyr heart rate monitors, [38], Nike+ FuelBand, [39] and Withings blood pressure monitor [40] have already been on the market for a while, and allow users to automatically collect data on their walking steps, activity levels, weight, food intake, blood pressure and heart rate. The newer generation of professional wearable medical sensors can easily connect to the smart phones and transfer the sensing results they collect directly to others. This has provided a more efficient and convenient ways to collect personal health information like oxygen saturation, blood glucose level, ECG [22, 23, 41, 42], EEG[43, 44] and electrocardiography (EKG). A future in which we are all equipped with devices and sensors that passively collect and interpret our health and activity data is not too far off.

Table 2.2 summarises the categories of current sensors and wearable devices and example applications. More information regarding the diseases and treatments are discussed in the later section.

Category	*Example Applications*
Safety Monitoring	Fall, stroke, seizure and heart attack detections, glucose monitoring
Treatment Efficacy Tracking	Sleep apnoea, multiple sclerosis and asthma treatments assessments
Early Detection of Abnormalities	Cardiovascular disease discovery, early detection of dementia and hypertension.
Home Rehabilitation	Attention deficit hyperactivity disorder (ADHD), and sleep apnoea treatments.
Fitness and Wellbeing Monitoring	Weight control, monitor chronic conditions, record sleep patterns, and track activities.

Table 2.2: Categories of wireless sensor devices and example applications

2.1.6. Personal Whole Genome Sequencing

Deoxyribonucleic acid (DNA) is the molecule that encodes the genetic information used in development and functioning of all known living organisms. The information in DNA is stored as a code made up of four bases: adenosine (A), cytosine (C), guanine (G), and thymidine (T). The entire complement of genetic information for the human genome is encoded in DNA sequences consisting of about 3 billion base pairs in total. The sequencing of the human genome was one of the greatest scientific achievements of the twentieth century [45, 46]. Because of our understanding of the interrelation between genes and disease, genome information has shown great potential for doctors and scientists to be able to accurately diagnose patients with unusual diseases, predict how patients might respond to treatments, and improve our understanding of the evolution of bacteria and viruses responsible for human diseases[47-50]. As the costs of whole genome sequencing continue to decline—a personal whole genome sequencing can now be undertaken for a few thousand dollars, there is a future potential for large amounts of genomics data to be generated and which will therefore need to be stored, transferred, and analysed.

The 1000 Genomes Project [51], launched in 2008, is the first project aimed at sequencing the genomes of a large number of people. The goal of the project was to gather genetic data from 26 populations around the world. At present, the project has collected 200 terabytes of genomic data including DNA sequenced from more than 1,700 people. The data has been made public and is currently available at Amazon. For the first time, scientists from all over the world are able to process and analyse the entire DNA sequencing data in one place [52, 53].

2.2. Quantified Self Movement

Measuring things to track progress towards a goal is a typical process used in many large organisations. Companies calculate their profits and turnovers; hospitals measure their waiting times; governments tot up unemployment and trade figures. Although some individuals track their weight or caloric intake in an effort to lose weight, most people do not routinely track or measure their activities, moods, sleeping patterns, body temperatures, or food intake. Most recently the era of cloud computing, smartphones, wearable sensors, social networks and big data have made the self-tracking, sharing and understanding of our physiological signals and lifestyles much easier than before, and these advances are encouraging people to begin to adopt the Quantified Self approach[54, 55]. People are becoming more active in discovering and tracking many day-to-day health parameters such as diet, health supplements, exercise, alcohol consumption, and conditions in their living environments that are correlated to their sleep quality, mood, and body functions [54, 55].

2.3. Technologies for Big Data Management

Quickly evolving modern technologies such as cloud computing, Internet of Things and intelligent data analysis have created great opportunities for better living. We envision the role these technological innovations will play in the health care sector as driving a shift in focus from offering better health care services only to people with problems, to helping everyone proactively achieve a healthier lifestyle. The advent of the cloud era has yielded new ways of storing, accessing and managing data. In this section, we discuss sensor networks, cloud computing, cloud storage, and other related technologies and services.

2.3.1. Sensor Networks

Sensor networks [56-59] provide the infrastructure through which we obtain data about the physical environment, social systems and health data by means of sensing devices. These types of networks are being widely used in areas such as healthcare and fitness. There are many existing sensor network systems such as Aurora [3], COUGAR [4] and TinyDB [2] that focus on storing and querying the sensor data. The Discovery Net system [60] provides an example in which different users can develop their own data collection workflows, which specify how their sensor data can be processed before storing it in a centralized data warehouse. The Discover Net also enables them to develop analysis workflows for integrating their data with data collected from other data sources. Users of the system could thus share the same data and also derive new views and analysis results through sharing. In a similar fashion, the CitySense [61] project deployed a system allowing the general public to provide feedback on pollutants using mobile devices.

2.3.2. Distributed File System

Distributed File System or network file system is a file system that unites the file systems of individual machines in network. Files are stored in different machines but are accessible via a computer network.

File Metadata

A file system's metadata is typically referred to as its file attributes (e.g., size, owner, created time etc). File metadata helps users and administrators understand the type of files stored in a system, where they are located, who they should belong to, and how they are accessed. Some studies have shown that over 60% of all file system calls are to access file metadata [62]. Therefore, the efficient management of file metadata is crucial for the overall file system performance. Traditionally, metadata and file data are managed and stored by the same file system, most of the time on the same device. In some modern distributed file systems, to address issues of scalability and performance, metadata is stored and managed separately by one or more metadata servers [63]. However, many of these systems still suffer from different levels of bottlenecks at high concurrent access rates [11].

Lustre

Lustre [64] is a type of distributed file system, is highly scalable and can support petabyte scale of storage and high performance I/O. Lustre was designed and developed by Cluster File System, Inc and later acquired by Sun Microsystems. Many supercomputers in the world use Lustre file systems. In Lustre clusters there are three essential components: the Clients, the Object Storage servers, and the Meta-Data server. This kind of design eliminates the bottle necks that often occurred in traditional block-based file systems and improves horizontal scalability of the data store. However, the single metadata server still provides quite limited metadata services [65]. These types of servers also do not support live data replication, making recovery really difficult.

Google File System

GFS [66] consists of a single master node and multiple chunk servers. Files are divided into fixed-size chunks (64 megabytes) and each chunk is replicated on multiple chunk servers. The master node stores metadata including: the file and chunk namespaces, the mapping from files to chunks, and the location of each chunk’s replicas. All the metadata is kept in the master node's memory.

Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) [67] is a distributed, reliable, scalable and open source file system, written in Java. It was inspired by the design of Google's MapReduce and Google File System. HDFS is designed more for batch processing than for interactive use by users. It enables applications to work with thousands of nodes and petabytes of data.

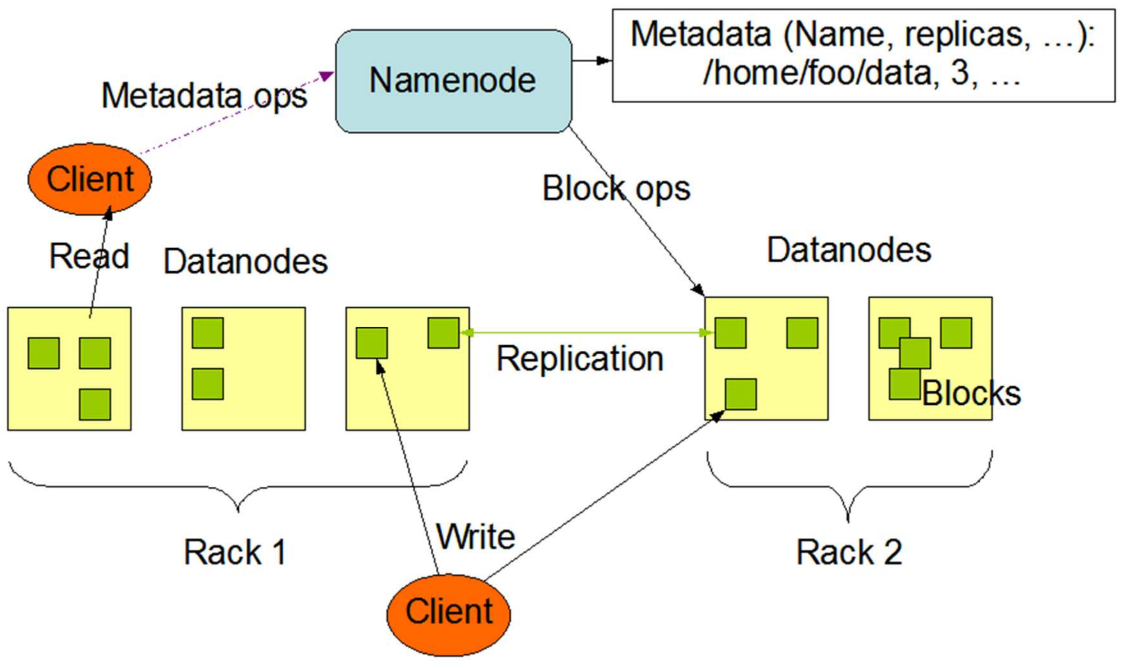


Figure 2.3: The architecture of Hadoop and blocks allocations on different DataNodes.

As shown in Figure 2.3, an HDFS cluster consists of a single node, known as NameNode, which manages the file system namespace and regulates client access to files. The rest nodes are DataNodes, store data as blocks within files. HDFS exposes a file system namespace and allows data to be stored in HDFS files. An HDFS file consists of a number of blocks with block sizes that can be configured. Each block is replicated specified number of times, and these replicas of blocks are stored on different DataNodes to improve data access throughputs and ensure data consistence in the event of failures. HDFS stores the file metadata in the memory of the NameNode server, providing limited metadata storage space. The NameNode is used to store all the file metadata in the memory, track the location of block replicas on the DataNodes, and execute basic file system namespace operations. Figure 2.3 shows how blocks are replicated on different DataNodes. The DataNodes are responsible for performing block creation, deletion and replication according to instructions from NameNode and serving read and write requests from the clients.

Hadoop has been used widely in a variety of companies and organizations and has become a *de-facto* standard for big data processing. For example, Yahoo! Search Webmap [68] runs on a 10,000 core Linux-based Hadoop cluster with over 5 Petabytes of disk storage space and produces data that is used in every Yahoo! Web search query.

2.3.3. NoSQL

In the past, relational databases were widely used for nearly everything. However, the growth of database volumes has intensified dramatically over the past decade. Typical relational databases have shown poor performance on data-intensive applications. In particular it is difficult to scale and to make transactions in a distributed system. NoSQL is a database movement that promotes non-relational data stores that do not require fixed table schemas, and are normally able to scale horizontally. Several NoSQL systems employ a distributed architecture, which offers data redundant and failure tolerant features.

BigTable

BigTable [69] is a compressed, high performance distributed storage system for managing structured data. It is not a relational database and can be scaled into petabyte-scale across thousands of commodity servers. BigTable maps two arbitrary string values (row key and column key) and timestamp into associated arbitrary byte array.

Cassandra

Apache Cassandra [70] is an open source, scalable and high-availability database management system with no single point of failure. It was initially developed by Facebook to enhance their inbox search feature. Cassandra is essentially a hybrid between a key-value and a column-oriented database. In Cassandra, each key can be mapped to many columns for storing data values. These columns are grouped together into sets called column families.

HBase

HBase [71, 72] is an open source, non-relational, scalable, and distributed database inspired after Google's BigTable and is written in Java. It provides a fault-tolerant way of storing a large quantity of sparse data. It is built for low latency operations, thus offer high performance in random data access. In addition, since HBase is commonly deployed on top of Hadoop, it offers effective MapReduce integration for distributed computation over the stored data. As a result, many enterprises and scientists adopt HBase for building real-time applications. For example, the social networking service–Twitter[73] uses HBase as a time-series database for storing performance data and create periodic MapReduce jobs over stored user data to power their people search feature.

MongoDB

MongoDB [74] is an open source, document-oriented database that provides high performance, availability and easy scalability. All data in MongoDB is treated in JSON/BSON format, therefore users have to insert, update, or fetch the document as a whole.

Redis

Redis [75] is an open source, in-memory, persistent, key-value store. It is designed for rapidly changing data with a foreseeable database size since the whole dataset is held in RAM. By default, Redis syncs data to the disk for persistence at least every 2 seconds. In the case of system failure, only a few seconds of data would be lost.

2.3.4. Crowdsourcing and Collaboration

Wiki-based Approach

In a general sense, a wiki is a web application that allows people to add, modify, or delete content in collaboration with others. Wiki differs from many other content management systems. It adopts little implicit structure and allows structure to emerge according to the needs of the users without allocating a defined owner or leader at the outset. The Wiki-based approach has enabled quick access to information and the rapid production of content. The past few years have seen an explosion in the number of organisations that successfully adopt the Wiki-based approach, ranging from very general like Wikipedia [76], to more specialised ones like WikiProteins [77], WikiPathways [78], and WikiGenes [79].

Feedback-based Approach

Websites such as *Stack Overflow* and *Stack Exchange* enable users to post their questions online and allow other users to provide answers and comments to the questions. Users can also rate the quality of posted information: “voting up” a question, answer, or comment indicates to the rest of the community that a post is interesting, well-researched, and useful, while “voting down” a post indicates that the post contains wrong information, is poorly researched, or contains misleading information. Receiving “up” votes increases the rating of a particular user’s profile, which presents a kind of encouragement mechanism for producing high-quality information.

Crowdsourcing and collaboration for sensors

The work in [80] presents a SmartPhoto framework to rate the quality of crowd-sourced photos based on the geographical and geometrical information including the smartphone's orientation, position and all related parameters of the built-in camera. The ShopProfiler [81] is an automatic profiling system with associated models to utilise crowd-sourced sensor readings from mobile phones to refine floor plans and characterise shops in terms of location, category and name with little human intervention. Artikis et al. [82] proposed a system for heterogeneous stream processing and crowdsourcing supporting intelligent urban traffic management. The system is able to detected complex events related to the urban traffic using the data collected from different sensors. The work in [83] introduces an approach of adopting a large number of GPS trajectories from multiple users to discover the correlation between users and locations, and mine interesting locations and classical travel sequences.

Social Media for Health

People have always been searching for the best knowledge to empower themselves to lead a healthier life. Social media has changed the nature of interactions among people and organizations. According to PwC's consumer survey of 1,060 U.S. adults, about one-third of consumers are using the social space as a natural forum for health discussions [84]. We can now share our lives and thoughts with a specific social community or the general public through social media and enjoy connection with and access to a wide variety of sources for personal fulfilment. Despite the often insurmountable issues of privacy and data protection facing the health and wellbeing industry, the benefits of embracing social media have begun to emerge. Today more and more patients are participating actively in all aspects of personal health information. At Sarasota Memorial Hospital [85] in Florida patients "tweet" their doctors when they have questions about their care. At Chicago's Rush University Medical Centre physicians keep connected with patients through Facebook and Twitter so that they are notified of the progress of their recovery [86]. During a real-time brain surgery in March 2009, doctors at Detroit's Henry Ford Hospital answered questions via "tweets," broadcasting to more than 6000 followers [87]. Other healthcare connector sites include PatientsLikeMe.com, which incorporates patient education with online peer-to-peer communication, using information sharing about conditions, symptoms, and treatments to link patients together to exchange information and insights. Doximity [88] and Sermo [89] are web and mobile based social networking platforms where physicians can share insights about medicine and specific cases. People are now becoming very interested in using data-sharing social platforms for healthcare to communicate with other people with common needs or conditions and to learn more about themselves.

2.3.5. Cloud Computing

Cloud computing has been widely discussed in the past few years as it shows great potential to shape the development, maintenance and distribution of both computing hardware and software resources. Within this computing paradigm, the actual provision of resources is only a concern at run time for specific application requirements, and this is also the case for software resources as they can also be used in an on demand and pay-per-use fashion.

After leading cloud provider Amazon launched their Spot Instances service, the path to commodity computing has become clearer. Through this first step, the idea of utility computing and cloud commoditization is becoming more of a reality. As cloud computing turns into a commodity, financial products and their derivatives - such as forward contracts, future contracts and options - could be brought into the market. Cloud computing users can then be the end-users who consume cloud computing resources, and can also act as investors who can make profit by trading the financial products of cloud computing as well. Furthermore, various economic pricing models from the real world market can be adopted by Cloud Computing to provide more opportunities and better services to end-users.

2.3.6. Conceptual Layers of Cloud Computing

Traditional service hosting environment provides limited flexibility on acquiring and releasing the resources, mainly offer dedicated servers in a monthly or yearly contract basis. Compared to the traditional infrastructures, the architecture of cloud computing is more modular. Each layer is loosely coupled with each other, allowing each layer to evolve separately. The architectural modularity allows cloud computing to support a wide range of application requirements while reducing management and maintenance costs. In general, cloud computing can be divided into four service layers:

- Infrastructure as a Service (IaaS): Fundamental hardware and network resources are provided as a service to the consumer. The consumer is allowed to deploy operating systems and applications on the infrastructure. e.g., Amazon's Elastic Compute Cloud (EC2), Amazon Simple Cloud Storage Service.
- Platform as a Service (PaaS): The consumer deploys consumer-created or acquired applications created using programming languages and tools supported by the provider onto the cloud infrastructure. The consumer does not manage or control the underlying cloud infrastructure including the network, servers, operating systems, and storage, He does, however, have control over the deployed applications, and possibly, the application hosting environment configurations. e.g., Google App Engine.
- Software as a Service (SaaS): The provider delivers an application based on a single set of common code and data definitions, which are consumed in a one-to-many model by all contracted customers, at any time, on a pay-for-use basis, or as a subscription based on usage metrics. The application is owned, delivered and managed remotely by one or more providers. e.g., Customer Relationship Management application on Salesforce.com, Gmail and Google Docs.

2.3.7. Virtual Machines

A virtual machine (VM) is an environment which functions like a physical computer in its ability to run an operating system and applications, but which does not physically exist. VMs are able to provide their own virtual hardware resources such as CPUs, memory, hard disks and network interfaces. Using virtualisation technologies, a physical machine (host) can be used to create and host multiple isolated virtual machines (guests). Applications running on guest operating systems are not aware that they are running on a VM and are able to function as they would on a physical machine. VMs can be easily copied and migrated between hosts for load-balance, maintenance, backups and disaster recovery purposes.

2.3.8. Cloud Storage

For many IT professionals, researchers and computer owners, finding enough storage space to hold data is a real challenge. Some people invest in ever larger external hard drives, storing files locally in order to deal with their data growth. However, protecting against the many unexpected things that can happen to computers or hard drives is not a trivial task. Similar issues exist for a number of large and geographically diverse enterprises; they store and process their rapidly growing data in several data centres and may adopt multiple storage systems. Finding and managing files in this increasingly large sea of data is extremely difficult. Although modern distributed computing systems provide ways of accessing large amounts of computing power and storage, it is not always easy for non-experts to use. There is also a lack of support in user defined file metadata in these systems. This has led most users to store a lot of data that strongly relates to the file content in relational databases. This separation of the file data and semantic metadata can create issues of scalability, interoperability and furthermore may result in low performance.

Unlike local storage, cloud storage relieves end users of the task of upgrading their storage devices constantly. Cloud storage services enable inexpensive, secure, fast, reliable and highly scalable data storage solutions over the Internet. Leading cloud storage vendors, such as Amazon S3 [5] and Google Cloud Storage[6] , provide clients with highly available, low cost and pay as you go based cloud storage services with no upfront cost. A variety of companies have outsourced at least some of their storage infrastructure to Amazon AWS, including SmugMug[90], ElephantDrive[91], Jungle Disk[92] and 37signals[90]. Recently, Amazon announced that it currently holds more than a trillion objects as of June 2012, the service has so far been growing exponentially [93]. Even so, many enterprises and scientists are still unable to shift into the cloud environment due to privacy, data protection and vendor lock-in issues. An Amazon S3 storage service outage in 2008 left many businesses that rely on the service offline for several hours and resulted in the permanent loss of customer data, [94, 95], an incident that led many to question the S3's "secret" architecture. In addition, the high read/write bandwidths that are demanded by I/O intensive operations, which occur in many different scenarios, cannot be satisfied by current internet connections [96, 97].

In Chapter 3, a multi-tier and scalable cloud storage system named CACSS is introduced as a key component of the Wiki-Health platform. Table 2.3 shows a comparison of the support features available on Amazon S3, Google Cloud Storage and CACSS. Many features are shared, with CACSS having the additional capability of metadata and user defined data searching.

	Amazon S3	**Google Cloud Storage**	**CACSS**
Bucket region support	Yes	Yes	Yes
Security control	ACL; access key with signature; policy	ACL; OAuth access	ACL; access key with signature; policy
Large file upload	Multi part upload	Resumable upload	Multi part upload
Data De-duplication	Unknown	Unknown	Yes
Object immutable	Yes	Yes	Yes
Host static website	Yes	Yes (launched in 2013)	Yes
Versioning	Yes	Yes (launched in 2013)	Yes
Access logs	Yes	Yes	Yes
Random read support	Yes	Yes	Yes
Random write support	No	No	No
Search support	Object key only	Object key only	Object key, system metadata and user-defined data
Performance as a Service	Not yet available	Not yet available	Yes and can be adjusted according to provider's

			goal. Support both object data and metadata caching
Encryption as a Service	Yes, Server-Side Encryption (launched in OCT 2011) and Client-Side Encryption (launched in JUN 2014)	Yes, Server-Side Encryption	Yes, Client-Side Encryption (Implemented in Feb 2013)
Pricing	Storage space, network usage and number of requests	Storage space, network usage and number of requests	All the accesses are logged. Different pricing models can be applied.
SLA	99.9% uptime and 99.999999999% durability	99.9% uptime	N/A

Table 2.3. Comparison of features between CACSS, Amazon S3 and Google Cloud Storage systems.

2.3.9. *De-facto* Standard of Cloud Storage Services

Amazon S3 is the leading *de-facto* standard of bucket-object oriented storage services. Successive cloud storage vendors, such as Rackspace [98] and Google [6] all adopt s3's style of bucket-object oriented interface. This style hides all of the complexities of using distributed file systems, and has proven to be a highly successful model [99] [100]—more than 2 trillion objects are stored in Amazon S3 and the service is able to handle more than 1.1 million requests per second. As Figure 2.4 illustrates, the bucket-object interface simply allows users to use the storage service from a higher level: an object contains file content and file metadata, and it is associated with a client assigned key; a bucket, a basic container for holding objects, plus a key uniquely identify an object. The volume of data and number of objects that a user can store is unlimited. Users are able to perform a number of operations using the interface such as encrypting their data, controlling access permissions, setting-up bucket policies, viewing access logs, storing multiple versions of the objects and hosting static websites.

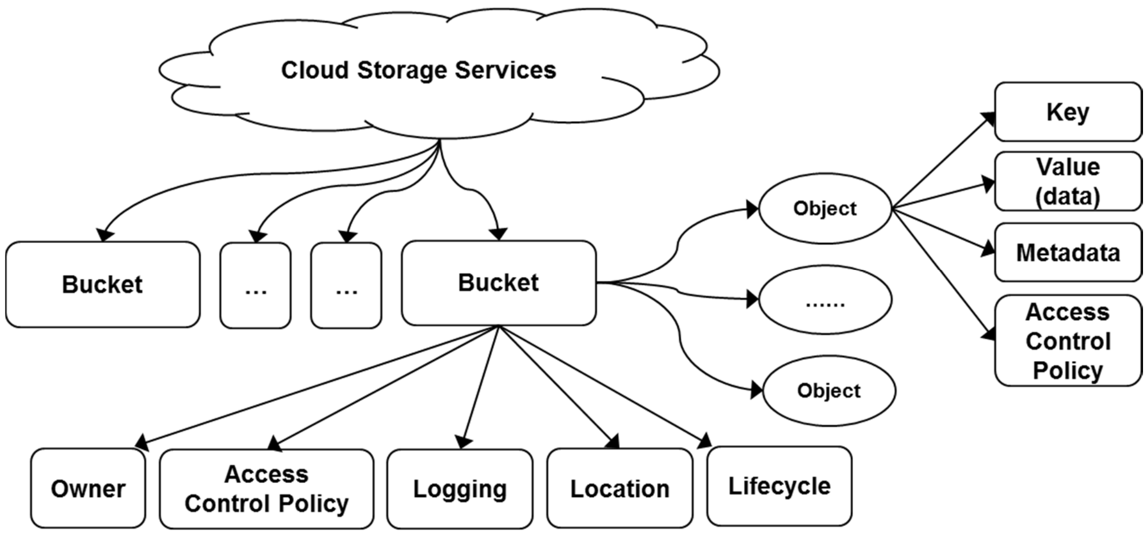


Figure 2.4: *De-facto* Bucket-Object interface for Cloud Storage Services.

2.3.10. Key Characteristics of Cloud-based Services

Pricing Models

Many major cloud computing providers (e.g. Amazon AWS and Microsoft Azure) employ pay-per-use and pay-on-demand based pricing models, which charge users according to their acquired resource and usage. Resource metrics including the number of CPUs, size of the memory and disk, the number of I/O requests, and network traffic are used to meter and track the usage. Some providers (e.g. GoGrid [101]) also offer pay-as-you-go based pricing model. This type of model is suitable for customers who have a strict budget, ensuring that they will never go beyond their consumption capability. Subscription is another pricing model used by many providers, which offers a certain service to the users for a fixed price for relatively longer periods of time. Spot-price based models also exist in the cloud computing paradigm, which utilise provider's available resources by offering them to the users through auction-style bidding. The price of these resources fluctuates with market demand but is usually significantly lower than the pay-per-use or subscription based price. Spot-price based services might not be suited for real-time or mission-critical applications, but offer a cost-effective way of carrying out computationally intensive analysis tasks.

High Scalability and Elasticity

Cloud computing offers consumers the ability to scale the resource capacity up or down at the speed of full automation. Elasticity is associated not only with scale but also with pricing models that enable scaling in both directions. As a result, users are able to easily expand their applications to large scales by acquiring more resources in order to handle rapid increases in demands and to reduce resources when the demand decreases.

Everything-as-a-Service (XaaS)

Through the cloud, nearly any hardware, software, digital resource, and business process can be offered as a service. Instead of investing a large amount of money in the capital cost of purchasing physical hardware and software, and installing packaged software applications on their computers, people and businesses will be able to access a wide range of cloud-based services on-demand over the Internet. A service-level agreement (SLA) is an agreement between two or more parties, where one is the customer and the others are service providers. Because of the service-driven and shared infrastructure environment of the cloud computing, users may not know where their data resides or understand the underlying architecture of provider's systems. SLA becomes an important protocol to guarantee many aspects such as the availability and performance of the services. SLA assurance is therefore a crucial object for every cloud provider.

2.3.11. Cloud-based Sensor Data Management

Implementing cloud computing technologies appropriately can aid service providers in improving the efficiency of operations, sharing information, improving collaboration, and managing expenditures. For example, Sensor-Cloud [102] enables an infrastructure on which users can share different physical sensors with others through the implementation of virtual sensors. Commercial sensor network platforms such as Xively [9] and ThingSpeak [10] have taken off in recent years. They provide an online scalable sensor data management platform that allows users to connect sensor devices and applications through a web-based application programming interface (API). In the area of data management for personal health data, HealthVault and WebMD are two leading platforms providing users with the capability to store, manage and share their health and fitness information. Another example is the Wiki-Health platform which is presented in this research. Different from a system like Xively or ThingSpeak, Wiki-Health is able to handle any type and format of sensor data and offers support for running analysis algorithms as services within the platform. A summary comparison of features among aforementioned systems is depicted in Table 2.4.

Time	before 2006	2007 - Now	2007 - Now	Now
Type	Data management systems for sensor networks	Web-based data management for Internet of Things	Collaborative data management for personal health data	Collaborative data management for personal health data
Examples	Aurora, COUGAR, and TinyDB	Xively, CitySense and ThingSpeak	HealthVault and WebMD	Wiki-Health
Challenges Addressed	Distributed systems for querying and aggregating sensor readings	Platform scalability and data sharing features	Platform scalability and data sharing, features	Solving Big Data challenges, ecosystem to enable collaboration and data analysis
Multi-Tenancy	No	Yes	Yes	Yes
Data Storage and Access	Support only time-series	Support only time-series	Support unstructured	Support both unstructured,

	sensor data	sensor data	data and time-series sensor data	semi-structured and time-series sensor data
Collaboration	Limited	Support user-defined tags and locations information of data streams	Marketplace for developers to publish apps and connect devices	Marketplace to enable analysis models as services. User-defined data
Data Analysis Capability	No	No	Limited, through 3rd party apps	Yes, through the analysis framework within the platform.

Table 2.4. Comparison between different data management systems

2.4. Data Analysis and Applications for Health

2.4.1. Data Analysis Tools

Scientists and researchers adopt various tools for data analysis and frameworks to design algorithms, analyse their datasets and build computational applications. Big data frameworks, like MapReduce have received a lot of attention recently for their ability to handle petabytes of data. The truth is that many data analysts and scientists are still dealing with relatively smaller amount of data, i.e. megabytes or gigabytes of data. This section will briefly cover some of the common data analysis tools.

MapReduce Framework

MapReduce [103, 104] is a programming model for processing large amounts of data with a parallel, distributed algorithm on a large number of computers (nodes). The model usually consists of two functions: a map function and a reduce function. The map function involving the master node takes the input, splits it into smaller pieces, and distributes them to worker nodes. A worker node may split and distribute its input again in turn. As a result, the original problem is converted into a multi-level tree structure of smaller problems. The worker node processes the smaller problem, then passes the result back to its master node. The reducer function activates the master node to collect the answers from worker nodes and combines them into the final answer to the original problem as requested.

Although MapReduce has proven to be a effective model for large-scale parallel programming, there are a few limitations [105, 106]: In order to carry out computational jobs on MapReduce clusters, users have to write their data analysis programs following a fixed acyclic dataflow – both map and reduce functions have to be stateless, not dependent on any data generated in the same MapReduce job. Therefore, changing existing iterative algorithms into MapReduce is very challenging since some algorithms that require multiple inputs are not well supported, no result can be obtained until the entire mapping process has finished, and most MapReduce systems are still not efficient in real-time based analytics.

Meanwhile, numerous efforts are under way to increase the usability of MapReduce for researchers, including Apache Mahout, a framework for executing machine learning algorithms on top of MapReduce, Spark, HaLoop [107] and M2M [108] .

MATLAB/Octave

MATLAB/Octave [109] [110] is a numeric computing environment and programming language used extensively by researches in many domains such as bioinformatics, physics, and chemistry. MATLAB/Octave runtime environment allows executable MATLAB programs to be deployed as real-time data analysis applications.

R Programming

R [111] is a free software programming language and software environment for statistical computing and graphics based on an implementation of S language [112-114]. It is now becoming one of the most widely used data analysis tools for statisticians and data analysts, mainly due to its object-oriented programming facilities, extensibility, data handling and modelling ability.

Julia

Julia [115] is one of the latest fast-growing programming languages that aims to address the requirements of high-performance numerical and scientific computing while keep a familiar syntax to users of other technical computing environments. Users are able to create distributed/parallel computations without imposing any particular style of parallelism.

2.4.2. Towards Self-Understanding

The advancement of the Internet of Things and wireless sensors have empowered users with an easier and more convenient way of digitalising our bodies by allowing the collection of our own personal health signals. This newfound capability also comes with a challenge – how can we utilise and make sense of the information we can now gather about our bodies? We envision that the approach of Self-Understanding will be a natural progression of Quantified-Self.

This section will discuss several common medical conditions that show potential for improved monitoring, management and treatment with the aid of modern sensor technologies. These examples have been chosen on our best knowledge of the available and effective bio-sensors currently on the market as well as solutions presently in development in various research communities.

Cardiovascular disease

Cardiovascular disease (CVD) is the leading cause of death worldwide. In 2011, nearly

160,000 people died in the UK from CVD. The estimated CVD-related healthcare cost to the UK's National Health Service (NHS) in total was £15.7 billion, representing 21% of the overall NHS expenditure in 2004 [116]. Tremendous pressure has been placed on maintaining the quality of healthcare while reducing costs. ECG is one of the most widely used physiological signals to monitor health. It helps doctors to diagnose various cardiac diseases and identify abnormal levels of certain minerals in the blood. Traditional tests only measure ECG signals over short intervals, a practice which doesn't always give the doctor enough information about the heart condition of the patient. With the evolution of technologies such as wearable health devices and smartphones, patients are able to self-monitor their ECG and other bio-signals at any time and any location. Being able to monitor a user's bio-signals over a longer timeframe allows us to better understand their typical lifestyle and behaviours. More importantly, it also allows for the discovery of any change-signals that could lead to potential health issues. A considerable amount of research has already been undertaken with the aim of developing efficient wearable CVD-detection systems [58, 117, 118].

Asthma and COPD

Asthma and Chronic Obstructive Pulmonary Disease (COPD) are both chronic respiratory diseases affecting the lungs which often have similar symptoms, such as causing airways to swell and narrow, making it difficult for those with these conditions to breathe. Patients with asthma and COPD must carry inhalers to use when they are experiencing symptoms. The time, frequency and use patterns of inhalers reflect levels of asthma control or COPD status in the patients who use them. Propeller (previously known as Asthmapolis program) [119] is a new type of wireless sensors that can attach to metered-use inhalers and connect with a smartphone via Bluetooth to allow patients to track the time and location of each inhaler use and provide personalized feedback and education on ways to understand and improve asthma control or COPD. The inhaler usage information can also be useful for alerting physicians and other asthma patients to particular locations associated with asthma attacks.

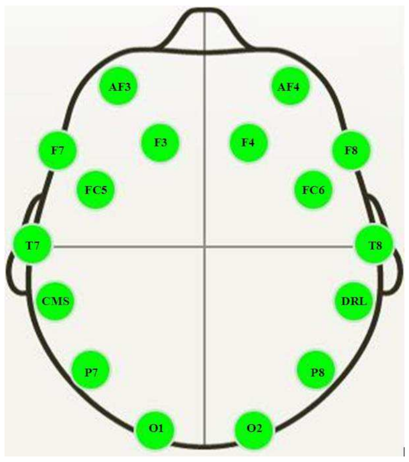
Sleep apnoea

Sleep apnoea is a sleep disorder in which suffers have pauses for 10 seconds or more in breathing or shallow breaths during sleep—commonly known as obstructive sleep apnoea (OSA). Sleep apnoea is most commonly a chronic condition that disrupts sleep cycles and can also lead to high blood pressure in the pulmonary artery, heart arrhythmias, and heart failure [120]. Continuous positive airway pressure therapy (CPAP) uses a machine to the OSA sufferer breathe more easily during the sleep. It is the most effective and most widely used nonsurgical treatment for OSA. The condition of OSA is often difficult to detect during routine health checks. The patients normally need to spend the night at a clinic sleep laboratory to be assessed and diagnosed by monitoring their respiratory rates, heart rates, oxygen levels, brain waves, muscle tone, body movement, and snoring throughout the night. This involves large expenditures in terms of resources for the health service, in addition to the inconvenience of overnight stay in the hospital environment and interruption of daily life activities necessary for the patient being assessed. A few studies [121-123] have recently been proposed to solve this problem by introducing the use of various wireless sensor systems to remotely monitor the patient's sleep and vital signs in the home environment. These approaches provide a more convenient, reliable, and inexpensive way of assessing the OSA status of patients before, during and after treatments.

Attention deficit hyperactivity disorder (ADHD)

ADHD is a condition that makes a person inattentive, impulsive and hyperactive. People with ADHD may also have additional problems, such as sleep, mood, and anxiety disorders. The symptoms of ADHD are more noticeable at an early age, i.e. between the ages of 6 and 12, and tend to improve with age. However, many adults who are diagnosed with the condition at a young age will continue to experience problems [124-126]. To the current level of medical knowledge of this condition, there is no cure for ADHD. Stimulant medications are commonly prescribed to manage ADHD symptoms. However, the reported side effects of these stimulants include restlessness, depression, headaches, and rapid heart rate [127, 128]. Apart from these short-term side effects, the long-term effects of these stimulants on adults are not fully known. Clinical EEG devices have been used in most studies but the disadvantage of using them are that they are expensive, intrusive and difficult to operate. Recently, a number of new low-cost, wearable, and easy-to-use wireless EEG devices are appearing on the market [43, 44].

In the paper [129], a prototype system of a video table tennis game for brain training is presented. As shown in Figure 2.5, the system uses an EEG device to monitor the alertness level of the user while playing the game. The purpose of the brain training game is to improve the user's ability to while helping them enter a stable and relaxed brain state at the same time. Typically, the difficulty level of the game should increase if the patient is not alert and decrease if patient is alert. Through this continuous feedback, the difficulty will be adjusted to a degree that allows the user to keep focus on playing game, but not become tense or agitated. This kind of brain training games can potentially be used for ADHD treatment. Apart from this, a considerable number of studies [28, 130-134] have also demonstrated the feasibility of adopting EEG-based technology to develop similar types of neurofeedback-based training and games, as non-drug alternatives for treating ADHD patients. These approaches show great potential for building next generation health care applications.



(a)

(b)

Figure 2.5: (a) Scalp locations of 14 sensors and 2 reference points of Emotiv EEG Device. (b) A subject wearing with an Emotiv EEG device playing the brain training game.

Multiple Sclerosis

Multiple sclerosis (MS) is a disease that affects nerves in the brain and spinal cord, causing symptoms such as fatigue and loss of balance, muscle control and vision. It is estimated that over 6000 people were diagnosed with MS in 2010 and there are more than 120,000 people with MS in the UK. MS is most commonly diagnosed in people between ages 40-50 years [135]. The cause of MS is still unclear and there is currently no cure for the condition, but there are treatments available which help to relieve the symptoms. Current clinical assessments requires patients to visit the hospital to take walking tests, such as a timed 25-foot walk to assess the level of disability and its progression over times in people with MS [136]. Walking speed, balance and gait reflect major dimensions of disability for indexing disease progression rehabilitation in MS. However, these dimensions also vary throughout the day. Being able to monitor longer periods of activity in MS patients outside clinical environments could enhance the ability to assess MS and monitor changes in patients over time. Recent studies [137, 138] have demonstrated the feasibility and potential for adopting low-cost and non-invasive wireless sensors for monitoring MS patients in order to capture more reliable measurements and improve self-management.

Dementia

Dementia is the medical term referring to a group of illnesses associated with the ongoing decline of the brain and its abilities that interfere with daily life. It is estimated that about 800,000 people in the UK have dementia [139]. The risk of developing dementia increases with age, and the condition most commonly develops in people over the age of 65. Signs of dementia are typically include memory loss, reduced concentration, reduced ability to think quickly, overall decline in languages, understanding, and judgement skill, as well as losses in motivation and interest in socialising and etc. Common types of dementia include Alzheimer's disease, vascular dementia, dementia with Lewy bodies and frontotemporal dementia. Most types of dementia are progressive and cannot be cured. Non-drug treatments such as mental training and physical activity can help people with dementia cope with the symptoms and rehabilitate their memories.

A recent strategy for dealing with the decline in function experienced by dementia sufferers is lifelogging, in which the principle idea is one of continuously capturing data about your life—ideally every moment—typically using images, videos and sounds, to create a complete digital record of your life. In 2013 and 2014, lifelogging reached a milestone with the launch of automatic and wearable cameras (as shown in Figure 2.6) such as Google Glass [140], Narrative Clip [141], and Autographer [142], which often have the capacity to capture photos or video. These devices generate a large number of time-series photos and videos throughout the day. One of the emerging challenges is how to efficiently manage, index, search, and use these collected photos, videos and context information for better healthcare applications. Some recent studies [143-145] have shown promise of adopting these lifelogging devices can potentially help dementia and amnesia sufferers to improve their memory abilities and recall important events. For example, the use of lifelogging can help dementia sufferers recall who they saw last week or what the doctor said to them in order to help them remember.

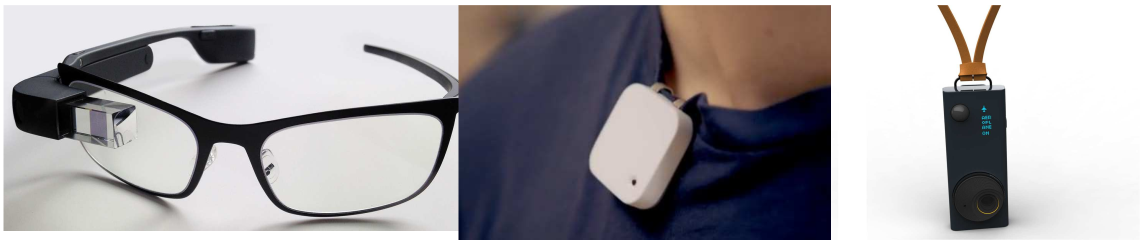


Figure 2.6: Wearable camera devices. From left to right: Google Glass, Narrative Clip and Autographer.

The ageing population

More than 10 million people in the UK are over 65 years old, and an estimated 19 million people will be over 65 by 2050. The growing number of elderly people is also impacting on the NHS [146]. In 2007-2008, the average spending of the NHS on services for retired households was £5200, nearly doubled the figure for non-retired households. A large portion of this spending arises from more than 500,000 elderly patients who end up in accident and emergency departments after a fall each year [147, 148]. Smart homes equipped with intelligent technologies can provide a way of preserving the ability of the elderly to safely remain in their own homes as long as possible. To this end, considerable research has been performed by scientists on different levels aimed at developing the direction of smart home-based health monitoring [149-152].

2.4.3. Combining Sensors and Genomics

A heart attack is a serious medical emergency in which the supply of blood to the heart is suddenly blocked, a condition that is normally produced by a blood clot. Lack of blood to the heart can seriously damage the heart muscle. Although there are ways to dissolve the blood clot that caused the heart attack by using medication or surgery, transferring the patient to the hospital and performing the treatment in time is still very challenging. No effective strategies to prevent heart attacks have been developed beyond population medicine approaches which mainly include lifestyle modifications: eat a low-fat, high fibre diet, moderate alcohol consumption, avoidance of smoking, take regular exercise and lose excess weight. Several associated variants [153-157] in human genomic information that are associated with the increased risk of heart attack have already been discovered and are independent of known risk factors. They can be combined in a generic test to determine the risk of heart attack [158]. Implanting nanosensors [159, 160] in the individuals whose genome sequence had already put them at risk of a heart attack have great potential in clinical use for these patients in the future. The ability to generate whole-genome sequences represents a major technological advance in the evaluation of heart attack risk. As our understanding of gene–gene and gene–environment interactions grow, the combination of wireless sensors and personalized genomic information will have a significant impacted in health care sector.

2.4.4. Collaboration Issues

In many domains, including healthcare, the use of traditional data collaboration tools such as email can lead to the creation of large amounts of duplicate data being stored in different places. For example, when a doctor sends a file to other doctors and nurses, each of those people will then save the file to their own hard disk. This kind of scenario occurs frequently and causes unneeded duplicate data to be stored unintentionally. In situations like this, there is a definite need to design collaborative tools to enable increased storage efficiency and reduce costs, such as implementing certain levels of data de-duplication and compression mechanisms.

2.4.5. Key Elements for Enabling Self-Understanding

As discussed in the previous sections, with the current revolutions in smartphone and other technologies, an increasingly powerful new set of tools is being developed that can potentially help us to reduce our use of doctors, speed up the pace of care and give more insight and power to patients. We still cannot replace the doctors with digital avatars, but the relationship between patients and doctors is transforming as a result of technological innovations. Not only are patients currently collecting, storing, displaying, and sharing blood pressure measurements and ECG signals obtained from the smartphone devices, but many are also beginning to adopt various analysis apps and tools in order to have their data analysed and interpreted before seeing a doctor. Although many bio-parameters can currently be measured or monitored directly by an individual such as blood pressure, oxygen saturation, glucose level, ECG, EEG, and body temperature, there are still many tests—for example X-rays, as well CT, PET-CT, MRI and ultrasound scans—that can only be performed by professionals because the technologies required are sophisticated to use, expensive to purchase and maintain, and can be hazardous to use without specific training. The key elements of a platform capable of helping individuals to understand their own health are as follows:

- Support for analysis tools that can provide different levels of interpretation of results to the patients, depending on their knowledge and background.
- Support for effective feedback and communication mechanisms between patients and doctors so that analysis results can be validated by doctors.
- Capability for measuring data believability, such as capturing the provenance of the data and software tools.
- Support for a community platform where patients can share their results, experiences, and treatment outcomes in order to learn from others and connect with people who have similar conditions.
- Creation of an easy-to-use interface/portal with sufficient guidance on how to ensure that the accuracy of measurements is as high as possible.

2.5. Summary

This chapter presented the background on data acquisition of health and medical information using different modalities, as well as the potential challenges of managing and making use of this explosion of big data. Various technologies exist that have the capacity to help us tackle these challenges, including sensor networks, distributed file systems, cloud computing, cloud storage, existing cloud-based sensor data management platforms, and data analysis tools. Several common diseases, the means by which they can potentially be monitored, managed or treated using modern technologies have also been discussed. The next chapter presents the architectural design of a cloud storage system named CACSS for managing unstructured data within the Wiki-Health framework.

3. CACSS Cloud Storage System for Unstructured and Semi-Structured Data Management

3.1. Introduction

Today, healthcare providers are experiencing explosive growth in digitised data. Medical imaging such as X-rays, computed tomography (CT), magnetic resonance imaging (MRI) and ultrasound represent a significant portion of that data. Most of these medical images are unstructured data. The increasing size and complexity of this data make their storage difficult to handle. Traditional storage systems face many challenges in managing this volume of data, and the emergence of cloud storage services is seen as a remedy. For healthcare vendors, adapting cloud storage services for managing unstructured data can potentially help them improve the quality of medical services, efficiency of operations, security of information sharing, and management expenditures as well as enhancing collaboration.

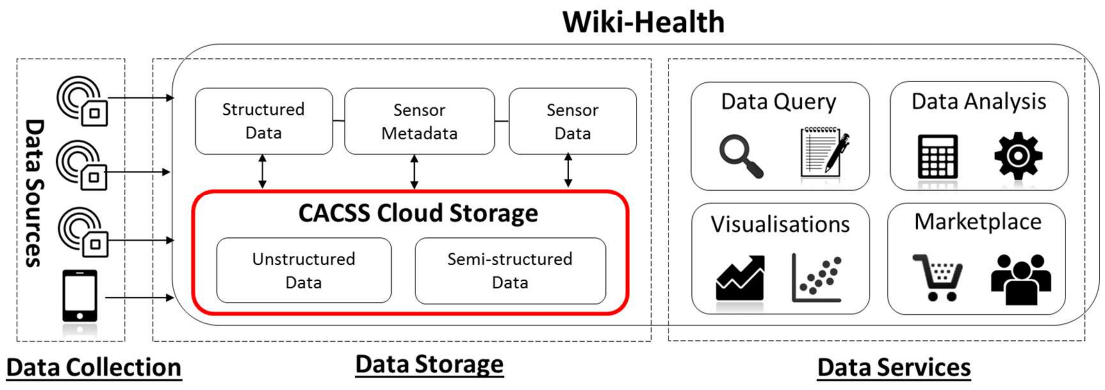


Figure 3.1: Overview of the Wiki-Health platform with a focus on the CACSS Cloud Storage System for managing unstructured and semi-structured data.

As shown in Figure 3.1, the CACSS cloud storage system serves as the fundamental storage infrastructure for managing unstructured data within the Wiki-Health framework. This chapter presents an in-depth explanation and analysis of the key features of cloud storage services for managing unstructured data and semi-structured data, and of how such services should be constructed and provided. This is achieved through the demonstration of design rationales and the implementation details of a real cloud storage system (CACSS).

3.2. Problem Analysis

3.2.1. Cloud Storage for Healthcare

Service providers in the healthcare industry are constantly faced with the challenges of relentless data growth. Industry estimates forecast that worldwide healthcare data will increase to 25,000 petabytes in 2020, rising from just 500 petabytes in 2012 [21]. High-resolution medical imaging and numerical data are primarily expected to account for the expanding data requirements. Overall, data can be broadly categorised as unstructured, structured, and semi-structured. Medical images, videos, plain text files, and PDFs are the most common examples of unstructured data. For structured data drawn from medical devices, time series is the most common form of representation. Size and complexity are the greatest difficulties associated with the storage of medical data. Advancing technology, emerging regulations, and decreasing budgets are presenting further challenges to healthcare providers. In response, providers are attempting to increase the extent of their collaborations, data usage, and proactivity for the purpose of becoming more patient-centred. Unfortunately, however, many traditional storage solutions cannot continue to meet the growing requirements [161].

Cloud storage is fast emerging as the technology of choice for many enterprises and scientists because it offers inexpensive, secure, fast, reliable, and highly scalable data storage solutions over the internet. For the most part, cloud computing and storage solutions are viewed as the solutions for many issues and challenges in healthcare. Implementing cloud technologies appropriately can help healthcare providers improve the quality of medical services and the efficiency of operations, share information, enhance collaborations, and manage expenditures. A selected summary of generic features for designing and implementing a cloud storage system and its key concerns for usage by healthcare applications include:

- Security and privacy control: for the health care industry, mandates such as the Health Insurance Portability and Accountability Act (HIPAA) [162], Health Information Technology for Economic and Clinical Health Act (HITECH) [163], European Union Data Protection Directive (EUDPD) [164], and UK Data Protection Act [165] have been issued by governments across the world to ensure health data security and privacy. Guidelines, such as digital imaging and communication in medicine (DICOM) [13] standards, have also been proposed to deal with security issues. In general, for any cloud storage system, various security models need to be implemented to ensure that documents and files are accessible at the correct time, location, and by the right person, providing adequate and accurate security control over the data without compromising performance.
- Computational: many applications in healthcare and science are becoming increasingly demanding in terms of their computational and data requirements. Some applications store terabytes of data and carry out intensive I/O operations. Examples include medical image processing and bioinformatics analysis applications. Improving the overall performance of such applications often demands a cloud storage system that can bring computational power closer to the data. Amazon Elastic MapReduce [166] adapts a hosted Hadoop framework running on the infrastructure of Amazon EC2 to provide an on-demand service to setup computational tasks and process data that are stored on Amazon S3.

- Metadata operations: metadata is the information that describes data files. Common metadata attributes include the time of an event, author’s name, GPS location and captions. Many scientists record information about their experimental configuration, including temperature, humidity, and other data attributes; for many it has become an integral part of storage. Accurate identification and adequate support for metadata queries of these metadata would bring additional values to the stored files and ensure that analyses and computations are carried out correctly and efficiently. The support of user-defined metadata, tags, and structures provide the capability of classifying and grouping the data in a user’s preferred approach. Unfortunately most storage systems are not capable of efficiently searching file metadata, especially those that are user-defined on a large scale. Leading cloud storage providers such as Amazon S3, Cloud Files [98] and Google Cloud Storage [6] offer cloud storage services for object data combined with a key-value style of user-defined metadata. These metadata can be retrieved and used by various user applications. Some research has already been done on the topic of metadata indexing and searching services [167, 168]. However, none of these cloud storage providers yet support any object search or query based on the object metadata.
- High scalability and performance: traditionally, metadata and file data are managed and stored by the same file system and most of the time on the same device. In some modern distributed file systems, to improve scalability and performance, metadata is stored and managed separately by one or more metadata servers [63]. However, many of these still suffer from bottlenecks at high concurrent access rates [169]. The metadata of a petabyte scale file system could contain in excess of billions of records, consuming terabytes of space or more. Therefore, the efficient metadata management of cloud storage systems is crucial for their overall storage system performance.
- Data durability: a far more common event than hardware failure or a disaster scenario is end user error, by which data is unintentionally deleted or overwritten. This demands that

cloud storage systems have sophisticated replication, versioning and recovery mechanisms to restore data.

- Support of various pricing models: the traditional pricing model for software has been a one-time payment for unlimited use. Cloud pricing models, such as pay as you go and pay monthly are very similar to the usage charges of utility companies. To accommodate this, it is necessary for cloud storage systems to have an efficient monitoring framework to track all kinds of resource usage, including network data transfer, I/O requests, amounts of data stored (file content and file metadata), and resources consumed for various computations.
- Interoperability: no specific standards that enable interoperability between cloud storage vendors currently exist, and this has created problems in the moving of data from one cloud to another. Similar issues arise in converting existing applications that are built on traditional file systems to the cloud. The ideal cloud storage system must offer a level of abstraction, portability and ease of use that enables the use of storage services with minimal support and development overhead.
- Storage efficiency: data redundancy is common and exists in many enterprises [170, 171]. For example, if a company director sends out a 15MB document to 150 employees in the company for a meeting, and each of those people saves the document to their hard drive, the document will take up more than a collective 2GB of storage on the hard disk. This kind of scenario constantly happens in everyday businesses, requiring so much unneeded data to be stored unintentionally. According to the International Data Corporation (IDC), more than 80% of corporations are exploring de-duplication technologies to reduce costs and increase storage efficiency [172]. One of the main reasons for this dramatic increase is file sharing across different endpoint devices and collaboration tools creating large amounts of data duplication. Implementing certain levels of data de-duplication and compression mechanisms in cloud storage systems can effectively eliminate unneeded

data and improve the storage efficiency.

3.2.2. Performance Matters

CACSS has been deployed on top of the IC-Cloud [173, 174] infrastructure and used by internal and external users for managing their daily work data as well as storing and processing large biological datasets.

With regular periodical analysis over the historical data logs, through monitoring the data access patterns and analysing the data stored in our system, it was discovered that while some data were accessed intensively over a period of time, other data were seldom accessed. When a large quantity of these kinds of “peak-time” scenarios happen simultaneously, the system suffers from network/disk I/O and CPU bottlenecks, and therefore reduces the overall service performance. Notably, these problems have been intensively reported and studied over many years by both academic and commercial organisations. Large-scale websites such as Facebook [175], Youtube [176], Wikipedia [177] and Reddit [178] have all used mem-cache [179] (a distributed caching system) as the central component in their platforms for tackling this problem. In 2008, Facebook reportedly used over 800 mem-cache servers delivering over 28TB of memory to serve over 200,000 UDP requests per second. Caching frequently accessed data has become an effective way to improve data access and processing performance.

Consequently, a decision was made to deploy our own “mem-cache” servers to deal with such “peak-access” problems. After a few weeks of operation, it was discovered that although the data access performance has improved, the overall system I/O (Disk I/O, network I/O) was almost doubled. This was largely due to the frequent data exchange between our storage servers and newly introduced mem-cache servers, as all objects (data) that were requested would be placed on the cache server first by default and were then moved back to the normal storage server. Therefore, an efficient mechanism was needed to place the most “desirable” objects into the caches at the optimum time, only removing them when other objects were considered to have higher access priorities through data access pattern analysis. Such a mechanism can efficiently use the available caching space; thus improving the overall performance of data access while also reducing disk I/O and CPU bottlenecks. Moreover, by offering users additional and superior performance as a separated service and charging them accordingly, and by usage, service providers can generate greater revenue.

Cloud storage systems are different from traditional single-user file systems in terms of economic models, architectures, and user behaviours. While considerable research has been undertaken on data caching in traditional file storage systems, there is still a lack of insight and evaluation for enabling caching as a service in cloud computing and big data environment.

3.3. Related Work

3.3.1. Cloud Storage Systems

Besides Amazon S3 and Google Cloud Storage, there have been quite a few efforts in cloud storage services, including the following:

Walrus [180] is a storage service included with Eucalyptus that is interface-compatible with Amazon S3. The open source version of Walrus does not support data replication services. It also does not fully address how file metadata is managed and stored.

The Openstack [181] project has an object storage component called Swift, which is an open source storage system for redundant and scalable object storage. However, it does not support object versioning at present. The metadata of each file is stored in the file’s extended attributes in the underlying file system. This could potentially create performance issues with a large number of metadata accesses.

pWalrus [182] is a storage service layer that integrates parallel file systems into cloud storage and enables data to be accessed through an S3 interface. pWalrus stores most object metadata information as the file's attributes. Access control lists, object content hashes (MD5) and other object metadata are kept in .walrus files. If a huge number of objects are stored under the same bucket, pWalrus may be inefficient in searching files based on certain metadata criteria; this factor can cause bottlenecks in metadata access.

Cumulus [183] is an open source cloud storage system that implements the S3 interface. It adapts existing storage implementations to provide efficient data access interfaces that are compatible with S3. However, details of metadata storage model and object data versioning features are not fully addressed.

3.3.2. Database Technologies

Object-oriented Database

An object-oriented database is a database management system in which information is represented in the form of objects as used in object-oriented programming. Object databases store objects rather than data and usually consist of the following:

- Attributes - Attributes are data that define the characteristics of an object. This data may be simple, consisting of integers or strings, or it may act as a reference to another complex object.
- Methods - Methods define the behaviour of an object and are what was formally called procedures or functions.

Example of object-oriented databases are Objectivity/DB [184], VelocityDB [185], EyeDB[186] and ObjectStore [187].

Object-oriented Database vs NoSQL

In the section 2.3.3, the concept of NoSQL databases is discussed. The data in NoSQL databases are often stored in a de-normalized state (by copying the same data into multiple documents or tables in order to simplify/optimize query processing), whereas the data in object-oriented databases (OODB) are often stored normalized in object at one place and are linked (relation) to other objects. Both of these types of databases have their own unique characteristics. For example, object-oriented databases eliminate the need for an inconvenient mapping layer between the database model and the application data model. They can be efficient when used to build applications with sophisticated object-to-object relationships. On the other hand, NoSQL databases such as HBase and MongoDB are not aware of the concept of object or the relations between objects. The application data model usually needs to be mapped into a Key-Value model (in which data is stored in unstructured records consisting of a key and the values associated with the corresponding record) to be stored in NoSQL. With ordered Key-Value, these databases can achieve high performance with respect to random data access and aggregation capabilities.

3.3.3. Data De-duplication

Data de-duplication is a data compression technique for eliminating duplicate copies of redundant data. The de-duplication technology has been widely applied in disk-based secondary storage systems to improve cost-effectiveness via space efficiency. It is most effective in storage systems where many copies of very similar or identical data are stored. Many studies on block-level and file-level data de-duplication have been carried out. One of the challenges facing large-scale de-duplication enabled storage systems are duplicate-lookup creates bottlenecks due to metadata and actual file data are stored separately and the large size of data index, which limits the de-duplication throughput and performance[188-194].

3.3.4. Techniques for Improving Performance

Caching data in faster devices such as RAMs and placing the data closer to the end users are both effective methods for improving system and application performance in conventional storage systems. Data caches have been an integral part of database systems for years, including MySQL and Oracle [195]. However, most of the existing cache solutions were read-only, which limited their usage to a small segment of applications until recently. Bi-directional updates are normally applied for updateable caches. Those updates, which happen in cache, are propagated to the target database and any updates that happen directly on the target database would cache automatically. Typically, the updates on cache table are propagated to the target database in two modes. Synchronous mode makes sure that after the database operation completes the updates are subsequently applied to the target database as well. In the case of asynchronous mode, the updates are delayed before being applied to the target database.

Beyond the reality that large-scale websites such as Facebook, YouTube, Wikipedia and Reddit have used mem-caches within their platform to reduce response time, data caching has been intensively studied in the academic field along with several commercial products. In the domain of content delivery network (CDN), Applegate et al. [196] proposed a content placement technique based on solving the mathematical models to decide on the placement of on-demand videos at the video hub offices (VHO) of a large-scale IPTV system. Their goals are to distribute videos among VHOs to minimise the total network bandwidth consumption while serving all requests and satisfying all disk and link capacity constraints. Valancius et al. [197] proposed a content placement strategy to calculate the number of video copies placed at ISP-controlled home gateways to reduce network energy consumption. Zhou and Xu [198] studied the video replication and placement problem in distributed video-on-demand clusters. They presented a replication and placement framework to minimise the load imbalance among servers subject to storage space and network bandwidth constraints.

At a more generalised level for data accessing and caching provision, Zhang et al [199] proposed a data migration model that takes application specific I/O profiles, application workload characteristics and workload deadline into consideration. It integrates solid state disks (SSD) into the multi-tier file system and migrates data between SSDs and hard drives (HDD) to improve performance. They look at the problem from the user's perspective, aiming to improve data performance for each of the users' applications. Eshel et al [200] introduced a clustered file system cache for parallel data-intensive applications that need to access and update the cache from multiple nodes while data and metadata is pulled into and pushed out of the cache in parallel. Data is cached and updated using pNFS, which performs parallel I/O between clients and servers, eliminating the single-server bottleneck of vanilla client-server file access protocols. Wang et al [201] addressed the problem of I/O performance fluctuation in cloud storage providers and proposed a middleware, titled CloudMW, to improve the stability and performance between the cloud storage provider and the cloud computing layer. Zhao and Raicu [202] proposed a file-level caching layer to improve the performance of distributed file systems, such as Hadoop's HDFS [67]. The most fascinating aspect of this work is that users are allowed to specify their desired catch size. This can be regarded as the beginning step towards the "performance as a service" concept for cloud/high-performance storage systems.

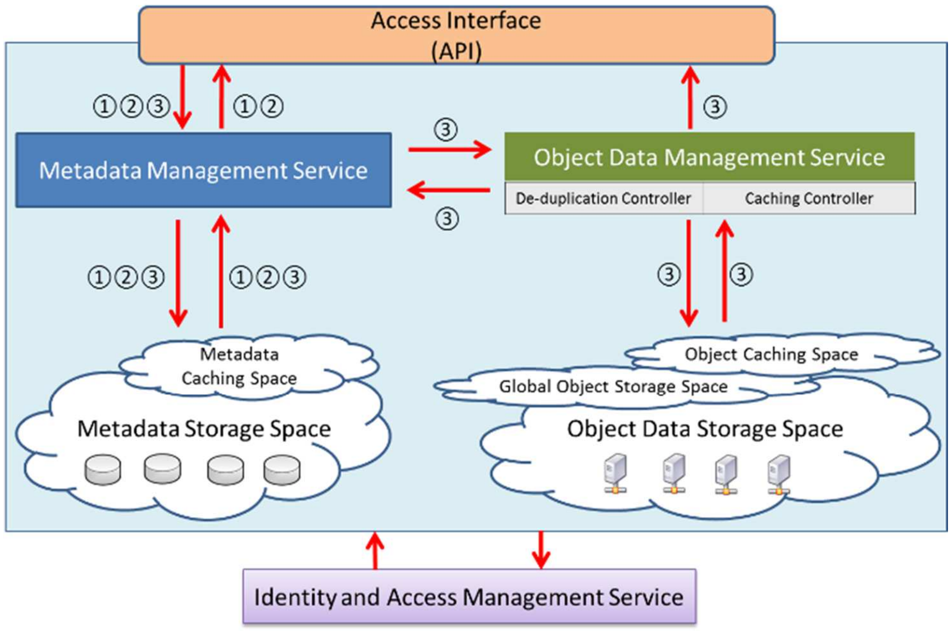
Additionally, there is further non-caching-based work for data performance improvement. McCullough, et al [203] introduced an interface for accessing cloud storage services by adaptively batching multiple requests together to reduce request latency. Instead of directly employing caching devices, this work focuses on application request caching. Under heavy workloads, they group all the same requests before sending them to the back end services, thus resulting in a higher throughput. Tran et al. [204] introduced distributed data overlays and simple policies for driving the migration of data across geo-distributed data centres in order to move data to the closest access point to the users, and therefore reduced the latencies caused by network communication. Dong et al. [205] introduced file merging, file grouping and prefetching schemes for structure-related and logic-related small files capable of improving the storage and access efficiencies of small files on HDFS. Through analysis of file access patterns in HDFS, they discovered the correlations between different small files. Therefore, they are capable of predicating which files have to be prepared after their highly correlated partners were accessed. Tirado et al. [206] proposed an elastic, cost-efficient infrastructure that adapts to workload variations through future workload predication. Using historical data, the system is able to forecast the data workload trend and seasonal patterns and then dynamically scales up and down servers beforehand.

We are also aware that caching in conventional file system has been studied intensively in the past. To the best of our knowledge, existing data caching and placement techniques in conventional file systems concentrate mostly on improving the response time of the service; however, these techniques rarely consider the issues of how to best utilise the available/limited high-cost caching resources for different purposes of cloud computing services. The workload required to enable performance as a service in the cloud computing paradigm, how to extend the existing features of cloud computing and cloud storage economic models, and how to adaptively adjust system configuration according to providers' requirements, is unfortunately absent. The rationale proposed in this thesis is a configurable service approach for cloud service providers to leverage the capability of composing different targeted storage services according to their needs.

3.4. Core System Design

Following earlier discussion on key characteristics of the generic private cloud storage system and related work, it is now time to present a detailed design rationale of the CACSS system. The architecture of CACSS is shown in Figure 3.2, consisting of the following components:

- Access interface: provides a unique entry point to the whole storage system.
- Metadata management service: manages the object metadata and permission controls.
- Metadata storage space: stores all of the object metadata and other related data.
- Metadata caching space: stores data caches of object metadata
- Object data operation management service: handles a wide range of object operation requests.
- De-duplication controller: manages global data de-duplication.
- Caching controller: provides data caching as a service, manages data placement for improving performance
- Object data storage space, global object storage space and object caching space: store all of the object content data in different circumstances.


Access Interface
(API)
①②③
①②
③
Metadata Management Service
③
③
Object Data Management Service
De-duplication Controller
Caching Controller
①②③
①②③
③
③
Metadata
Caching Space
Metadata Storage Space
Object Caching Space
Global Object Storage Space
Object Data Storage Space
Identity and Access Management Service

①: Put Bucket, Get Bucket, Put/Get Bucket acl, Put/Get Bucket website, List Multipart Uploads, Put/Get Bucket logging, Metadata Caching Requests
②: Head Object, Put/Get Object acl, List Parts, Initiate Multipart Upload
③: Put/Post Object, Get Object, Delete Object, Upload Part, About Multipart Upload, Object Data Caching Requests

Figure 3.2: CACSS architecture

3.4.1. Access Interface

CACSS offers a web-based interface for managing storage space and searching for objects. The current implementation supports Amazon's S3 REST API, the prevailing standard commercial storage cloud interface.

3.4.2. Identity and Access Management service (IAM)

IAM is a separated service that provides authorisation and access control of various resources. It offers sub user, group management, and precise permission control of which operations a user can perform and under what conditions such operations can be carried out.

3.4.3. Structure of Metadata Storage

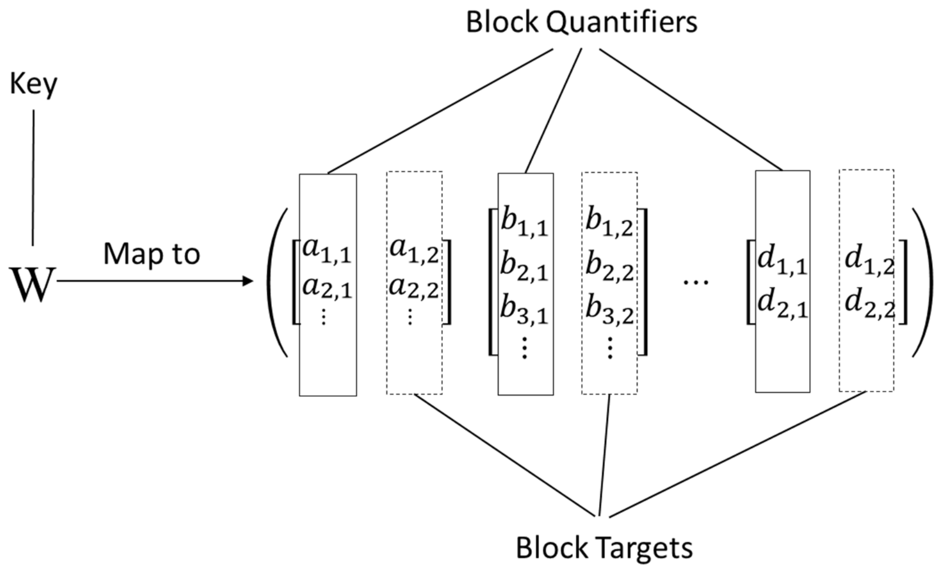
To achieve high performance in metadata access and operation, the CACSS's object metadata and content are completely separate. Each object's metadata—including its system metadata such as size, last date modified, and object format, together with user defined metadata—are stored as a collection of blocks in the CACSS's Metadata Storage Space (MSS). A unique index key is used to map each collection. Each collection consists one or more blocks. MSS keeps all of the collections' data sorted lexicographically by the keys. Each block is akin to a matrix that has exactly two columns and an unlimited number of rows. The values of the elements in the first and second columns are block quantifiers and block targets, respectively. All of the block quantifiers have unique values in each block:

$$\text{Block}=\left[a_{i,j}\right]\ 1\leq i\leq m,\ 1\leq j\leq 2,\ \text{where } m\in\mathbb{Z}^{+}$$

E.g. a key W maps to a collection:

$$\left(\begin{bmatrix} a_{1,1} & a_{1,2} \\ a_{2,1} & a_{2,2} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} b_{1,1} & b_{1,2} \\ b_{2,1} & b_{2,2} \\ b_{3,1} & b_{3,2} \\ \vdots & \vdots \end{bmatrix} \cdots \begin{bmatrix} d_{1,1} & d_{1,2} \\ d_{2,1} & d_{2,2} \end{bmatrix}\right)$$

The corresponding key, block quantifiers and block targets are annotated in the figure below.

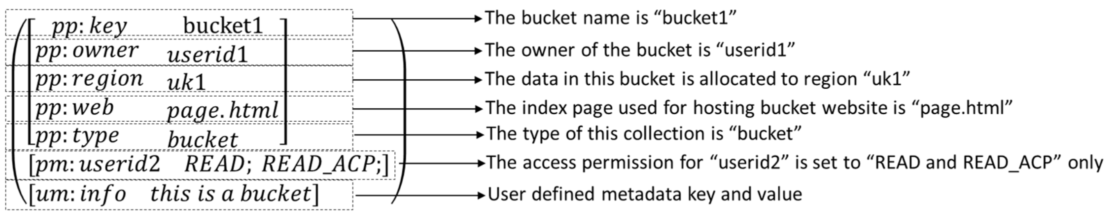

Block Quantifiers
Key
W
Map to
$a_{1,1}$ $a_{1,2}$ $a_{2,1}$ $a_{2,2}$
$b_{1,1}$ $b_{1,2}$ $b_{2,1}$ $b_{2,2}$ $b_{3,1}$ $b_{3,2}$
$d_{1,1}$ $d_{1,2}$ $d_{2,1}$ $d_{2,2}$
Block Targets

Bucket

To reduce interoperability issues, CACSS adopts the de facto industry standard of buckets as basic containers for holding objects.

Unlike some traditional file systems, in which a limited number of files can be stored in a directory, there is no limit to the number of objects that can be stored in a CACSS bucket. CACSS has a global namespace—bucket names are unique and each individual bucket's name is used as the key in MSS. Various block quantifiers and block targets are used to store a variety of information, including properties of a bucket or an object, permissions and access lists for a particular user, and other user defined metadata.

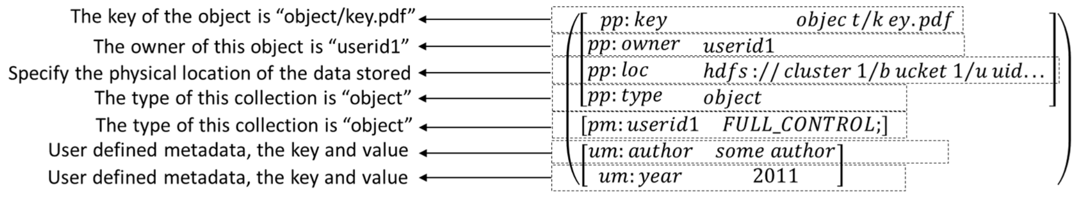
For example, for a bucket named "bucket1", a key "bucket1" should exist, which maps to a collection of data such as:


pp: key bucket1
pp: owner userid1
pp: region uk1
pp: web page.html
pp: type bucket
[pm: userid2 READ; READ_ACP;]
[um: info this is a bucket]
The bucket name is "bucket1"
The owner of the bucket is "userid1"
The data in this bucket is allocated to region "uk1"
The index page used for hosting bucket website is "page.html"
The type of this collection is "bucket"
The access permission for "userid2" is set to "READ and READ_ACP" only
User defined metadata key and value

Objects

The index key of each object is comprised of a string, which has the format of the bucket name together with the assigned object key. As a result of this nomenclature, objects of the same bucket are naturally very close together in MSS; this improves the performance of concurrent metadata access to objects of the same bucket.

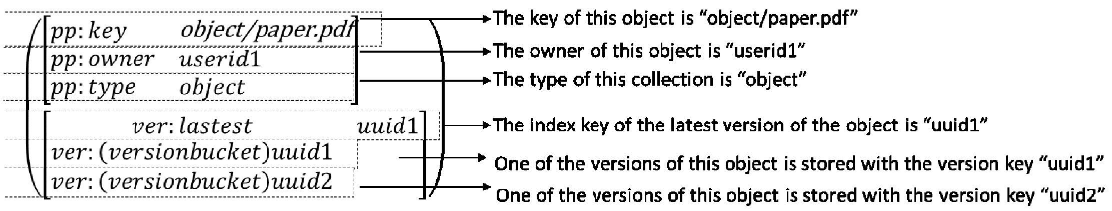
For example, considering an object with the user-assigned key "object/key.pdf" in bucket "bucket1", a key of "bucket1- object/key.pdf" should exist, which maps to the following collection of data:


The key of the object is "object/key.pdf"
The owner of this object is "userid1"
Specify the physical location of the data stored
The type of this collection is "object"
The type of this collection is "object"
User defined metadata, the key and value
User defined metadata, the key and value
pp: key objec t/k ey.pdf
pp: owner userid1
pp: loc hdfs :// cluster 1/b ucket 1/u uid...
pp: type object
[pm: userid1 FULL_CONTROL;]
um: author some author
um: year 2011

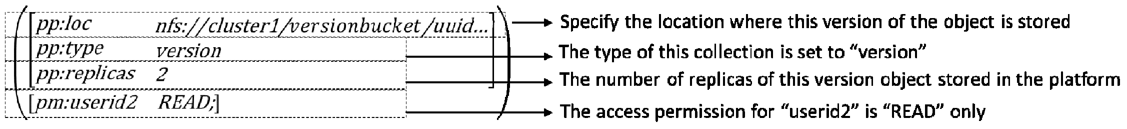
3.4.4. Object Versioning

When the versioning setting is enabled for a bucket, every object key is mapped to a core object record. Each core object record holds a list of version IDs that map to individual versions of that object.

For example, for an object with a predefined key "object/paper.pdf" in bucket "versionbucket", an index key of "versionbucket − object/paper. pdf" should exist, which maps to the collection data shown below:


pp: key object/paper.pdf
pp: owner userid1
pp: type object
ver: lastest uuid1
ver: (versionbucket)uuid1
ver: (versionbucket)uuid2
The key of this object is "object/paper.pdf"
The owner of this object is "userid1"
The type of this collection is "object"
The index key of the latest version of the object is "uuid1"
One of the versions of this object is stored with the version key "uuid1"
One of the versions of this object is stored with the version key "uuid2"

Similarly, the object's version record with row key "versionbucket-object/paper.pdf-uuid1" maps to the collection data:


pp:loc nfs://cluster1/versionbucket/uuid...
pp:type version
pp:replicas 2
pm:userid2 READ;
Specify the location where this version of the object is stored
The type of this collection is set to "version"
The number of replicas of this version object stored in the platform
The access permission for "userid2" is "READ" only

3.4.5. Metadata Management Service

Metadata management service (MMS) manages the way in which an object's metadata is stored. In such a system, a client will consult the CACSS MMS, which is responsible for maintaining the storage system namespace, and they will then receive the information specifying the location of the file contents. This allows for multiple versions of an object to exist.

MMS handles requests as follows. First, it checks if a request contains an access key and a signed secret key. CACSS consults IAM and MSS to verify whether the user has the permission to perform the operation. If they do have permission, the request is authorized to continue. Error information is returned if they don't. If a request does not contain an access key or a signed secret key, MMS is looked up to verify if the request to the bucket or object is set as publicly available to everyone. If it is set as public, then the request continues to the next step. All the requests are logged, both successful and failed. Logging data can be used by both the service provider and storage users for billing, analysis, and diagnostic purposes.

In contrast to traditional storage systems that limit the file metadata which can be stored and accessed, MMS makes metadata more adaptive and comprehensive. Additional data regarding file and user-defined metadata can be added to the metadata storage, and these data can be accessed and adopted on-demand by users or computational works at any time. Searching via metadata is another key feature of CACSS.

3.4.6. Object Data Management

CACCS stores all unstructured data, such as file content, in the Object Data Storage Space (ODSS). ODSS is intentionally designed to provide an adaptive storage infrastructure that can store unlimited amounts of data and that does not depend on underlying storage devices or file systems. It works closely with MMS to maintain the whole namespace of CACSS. Storage service vendors are able to compose one or multiple types of storage devices or systems together to create their own featured cloud storage system based on their expertise and requirements in terms of level of availability, performance, complexity, durability, and reliability. Such implementation could be as simple as NFS [207], or as sophisticated as HDFS [67], PVFS [208], and Lustre [64].

3.4.7. Use Cases and Scenarios

As shown in Figure 3.3, developers are able to build scalable applications by adopting the functionalities of the CACSS system for managing their unstructured and semi-structured data through the web-based API provided. CACSS can be used users and applications use their preferred means of organising buckets, objects, keys and permission controls to the CACSS system. For example, the current Wiki-Health platform uses one account on the CACSS and hosts all its users' unstructured and semi-structured data within the CACSS and under a single bucket called "Bucket-Wiki-Health". The access between Wiki-Health and stored data in the CACSS has to go through the CACSS's security control, while, Wiki-Health uses its own security control for its users and applications.

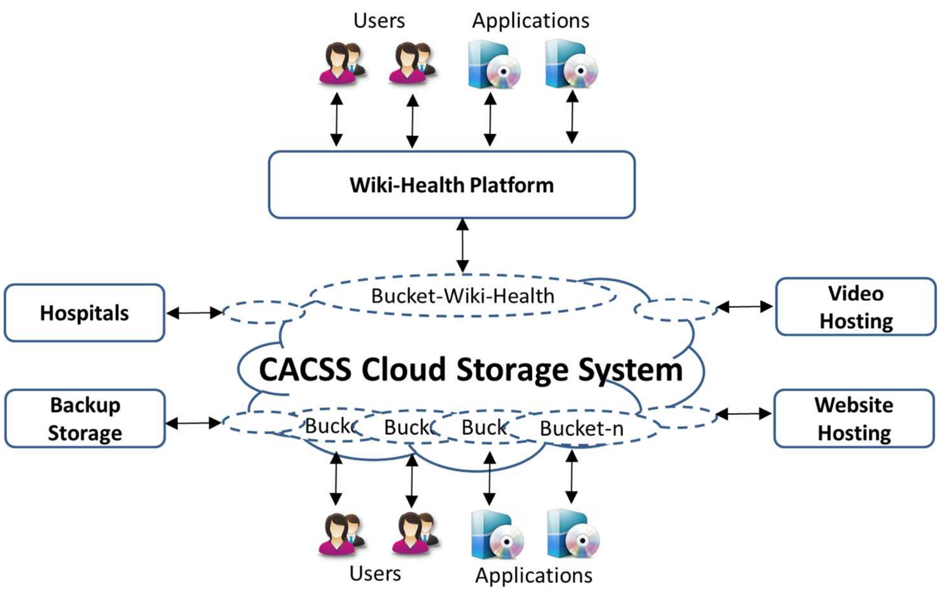


Figure 3.3: Use Cases and Scenarios of CACSS

3.4.8. Data De-duplication

To enable data storage efficiency, CACSS has introduced a De-duplication Controller (DDC) and a Global Object Storage Space (GOSS) into the design. File-level de-duplication is not as efficient as block and byte level; however, it is more lightweight and requires less processing power in generating checksum hashes. Therefore, currently, the DDC is only implemented to use a global file-level de-duplication method, which manages how and where duplicated objects should be stored in the GOSS. If a bucket is configured to enable data de-duplication, all the objects in this bucket will be stored in the GOSS. It would be extremely unlikely to have a collision between two files with different content but the same SHA-256 checksum [209]. Thus, CACSS uses the SHA-256 hash function to calculate the checksum of each incoming storing object, and if the checksum does not exist in the MSS, a new de-duplication object metadata record $ddes, and the hash value as the key will be inserted as the following:

$$\text{e.g. Row key "}\$ddes - D3F4D74F814D55EE80 \ldots\text{" maps to records:}$$
$$\left(\begin{bmatrix} pp: < bucket1 >< user1 > key1 & \\ pp: < bucket2 >< user1 > key2 & \\ pp: < bucket3 >< user2 > key3 & \\ & \\ pp: filesize & 19751324 \end{bmatrix}\right)$$

The physical file content will be saved into the GOSS as a new file. If the checksum already exists in the MSS, the file size and file content are also compared with the existing stored object with the same checksum to ensure they are perfectly identical. If they are identical, the record will be updated to attach this object's bucket name, object key, and the user id (Figure 3.4); if they are not identical or the checksum does not exist, a new record will be created. DDC supports both in-line where not-yet-deduplicated object data is never stored and post-process de-duplications where the deduplication process is performed after object data is stored. Post-process de-duplications is normally used for a bucket in which data de-duplication was not initially configured, but enabled at a later stage.

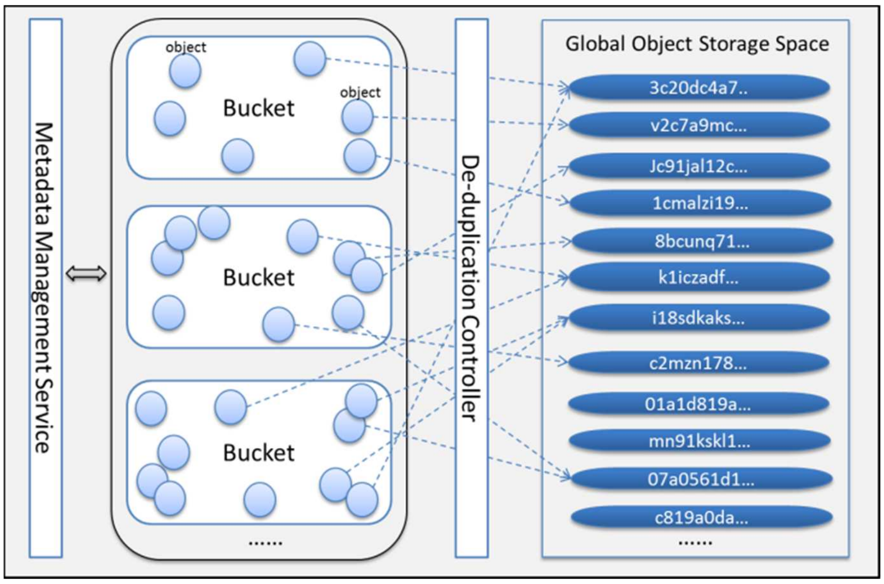


Figure 3.4: Bucket-Object to Global Object Storage Space Mapping

3.4.9. Security and Privacy Control

Security and privacy are the key concerns for any cloud service users. CACSS has enabled the following data protection mechanisms to improve the level of security and privacy.

Access Control List

CACSS supports user authentication and permission operations to control access to data. All buckets and objects are associated with access control lists (ACLs). With ACLs, data owners can grant specific permissions to specific users or groups for an individual bucket or object.

Authentications

CACSS enables user identity authentication through http request headers or query strings. Users are required to compute a signature with predefined access key, secret key and the targeted resources. MMS compares the server computed signature based on incoming request parameters against the signature presented by the requester. If two signatures do not match, MMS responds with an error message.

Data Encryption as a Service

Security and data breaches are a concern for any industry utilising the cloud, but are especially important in the case of healthcare applications. Any business or vendor dealing with personal and sensitive information must ensure data is held securely and that no other parties can access or read the data, even the cloud storage service providers themselves. Many cloud storage vendors such as Amazon have been developing and proposing various encryption mechanisms [210, 211] in recent years. Security-as-a-Service is becoming one of the key features of a complete cloud storage system and is evolving rapidly. CACSS tackles the security problem by introducing an on-demand "Data encryption as a service" (DEaaS) approach that allows users to encrypt the data stored in the CACSS.

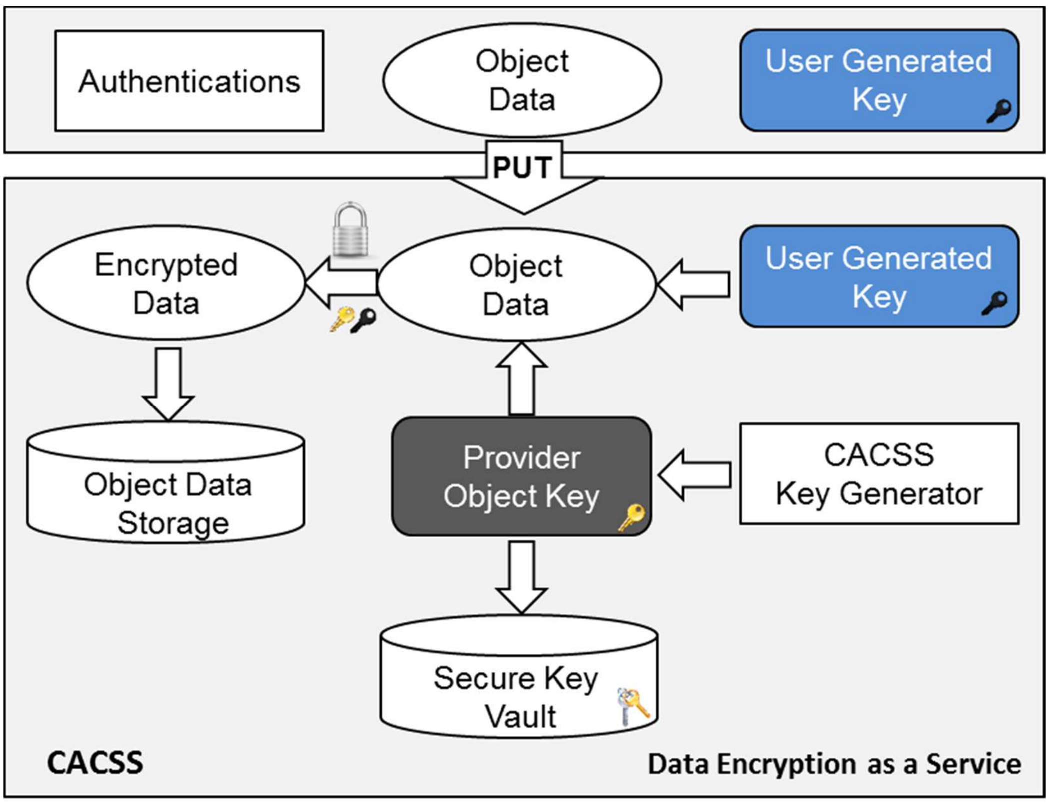


Figure 3.5: The design of data encryption as a service

To place a new object into CACSS and have it stored with encryption, the user first needs to create a personal key and send it with object data and authentication parameters (as shown in Figure 3.5). Then, CACSS generates a unique provider object key for the object. Next, the object data is encrypted using both user's personal key and provider object key and saved in the object data storage space. For added security and protection, the provider object key is stored in the secure key vault, a separate and different host from where the object data and metadata are stored. A user's personal key is never kept in the cloud storage system. CACSS uses an AES algorithm[212] for encrypting object data.

To access the encrypted data, two different types of keys are needed: the personal key defined by user and the provider object key (as shown in Figure 3.6). If both keys are correct, data is decrypted and returned to the user. Without both keys, the data can't be decrypted. The personal key is only used for decryption and never kept in the system for long time periods.

As the decryption is performed in the provider's part, which cannot be directly inspect by the user, this mechanism does require some degree of trust between the user and the cloud storage service provider.

DEaaS is able to protect against data breach by cloud storage service providers and attackers, as well as the scenarios of stolen and lost disks. If extra security is needed, users can always use their own method to encrypt the data before sending and storing it to CACSS.

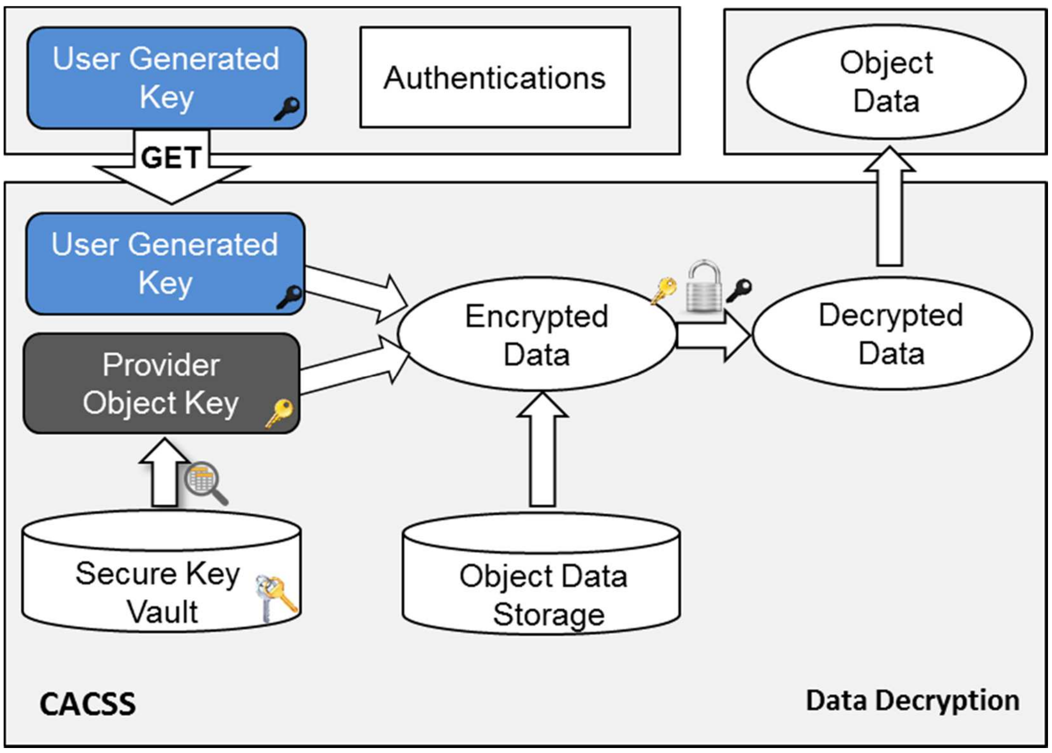


Figure 3.6: The design of data decryption workflow

3.5. Performance as a Service Design

In this section, we present an in-depth analysis of our approaches for managing data caching as part of the cloud storage platform. A thorough demonstration of CACSS can offer full details on what needs to be considered when a caching (high-performance) component is provided as a service and how to integrate a caching algorithm into a cloud storage system that can extend the existing features of cloud computing and cloud storage economic models, and to utilise available resources according to providers' requirements.

3.5.1. Data Caching Management

As explained in the previous sections, by making CACSS public as a service to the internal and external users, a discovery has been made shows the "peak-time" data access behaviours in various scenarios. To understand whether similar data-access behaviours and patterns exist in other cloud storage spaces in general and to gain more insight in other larger-scale websites and applications, an extensive analysis of the Page View statistics of Wikipedia Project has been performed, as collected by Domas Mituzas [213].

From the analysis of Wikipedia traces, we found a variety of interesting access patterns (some are which are presented in Figure 3.7). For example, there are file-access patterns that exhibit a linear increasing trend, while some have a diurnal pattern where the access frequency peaks during the day and also at night. These findings have been useful to us in our investigations of what is really happening in real-world scenarios and have enabled us to consider how we can utilise available resources (limited caching storage spaces such as mem-cache servers) to improve performance without incurring unnecessary costs, as well as demonstrating how we can leverage the existing capability to maximise revenue for providers. To build on these findings, we introduced the Local Data Caching (LDC) model to determine the placement of objects for caching through analysing historical data.

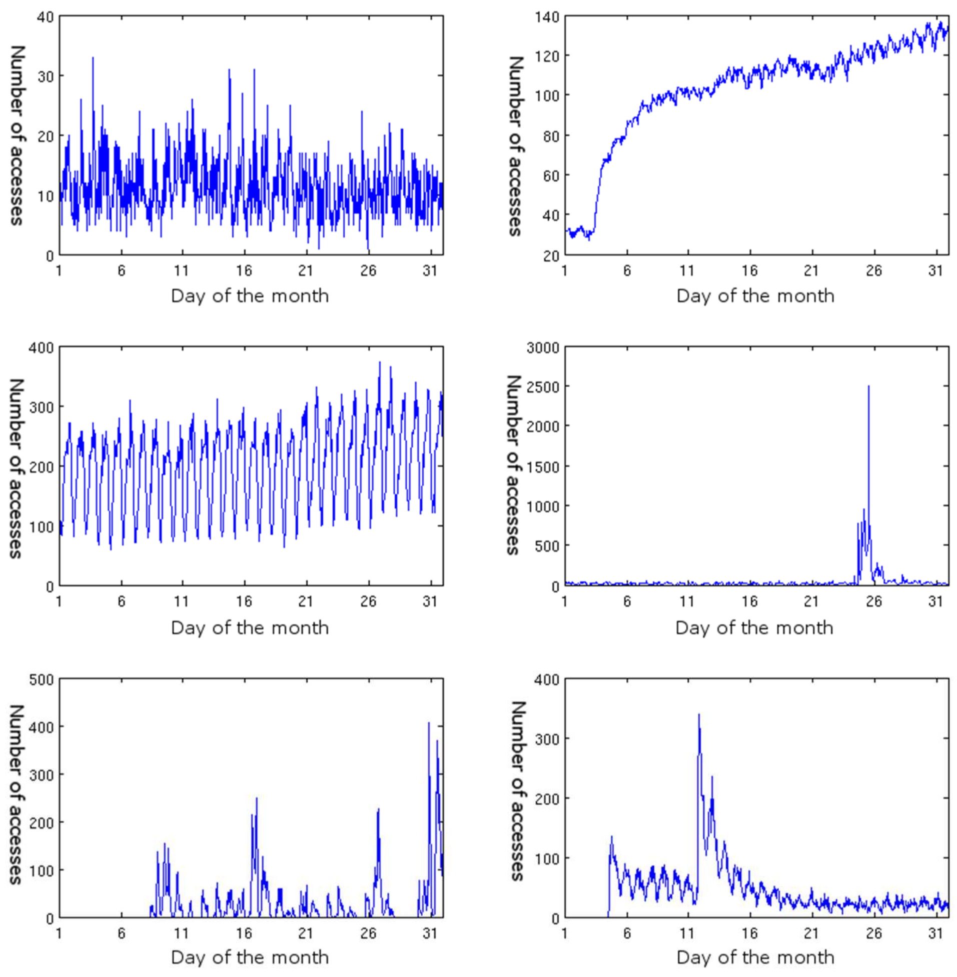


Figure 3.7. Examples of access patterns in Wikipedia traces.

3.5.2. Local Data Caching Model

The objective of the LDC model, in relation to the data caching issue, is to place the correct data in caching spaces for an optimum period of time via maximising an overall score, subject to cache space capacity. This model takes recent data access history, current resource usage, and system status as parameters to predicate the future demand for particular pieces of data. It reconciles the discrepancy of the trade-off between the goals of improving the performance for users and increasing the revenue for service providers through using a score-based approach. The score is calculated for each object based on the potential performance improvement, the expected earnings through that improvement, and the cost of moving the object if the object is to be placed into the cache. The model is defined as follows:

$$Maximize \sum_{m \in \mathrm{M}} S_m x_m \tag{1}$$

where

$$S_m = \alpha f(m) + \beta E(m) - \gamma\, C(m) \tag{2}$$

$$E(m) = V_m P_r + V_m z_m P_t \tag{3}$$

$$C(m) = \tau z_m + \psi \frac{R_m}{V_m} + \theta U_c + \iota U_r + \kappa U_n \tag{4}$$

$$\alpha, \beta, \gamma, \tau, \psi, \theta, \iota, \kappa \ are\ constants$$

subject to

$$\sum z_m x_m \le d \tag{5}$$

$$x_m \in \{0,1\}, \mathrm{m} \in \mathrm{M} \tag{6}$$

Parameters	**Semantics**
S_m	total score of putting object *m* into cache
M	set of objects
T	set of time points
$f(m)$	performance-improvement function for object m ∈ M
z_m	size of object *m* ∈ M
V_m	data access times of object m ∈ M
d	cache space capacity
P_t	data transferring per unit price
P_r	data requests per unit price
U_c	CPU utilisation percentage at present
U_r	ram usage at present
U_n	network utilisation percentage at present
$E(m)$	expected earning of object *m* ∈ M

$C(m)$	transfer cost function for object $m \in \mathrm{M}$
R_m	number of object change times, $m \in \mathrm{M}$
Decision Variable	**Meaning**
x_m	binary variable indicating whether to store video $m \in \mathrm{M}$ in cache space

Table 3.1: Parameters and Semantics of the Local Data Caching Model

Constraint (5) ensures and reflects the limited size of cache space. For each object $m \in \mathrm{M}$, x_m is a binary variable indicating whether we should cache m (i.e., yes, if $x_m = 1$; no, if $x_m = 0$). S is the overall score function used for calculating the score of each object m. $f(m)$ is a list of (in our case) functions for calculating performance improvement after placing object $m \in \mathrm{M}$ from normal storage space to caching space. Any object with a negative calculated score is reset to 0 to ensure the minimum score of any object is 0. Since different storage hardware devices and underlying file systems have varied characteristics (such as performance, cost, durability and stability), how to design the performance improvement function $f(m)$ depends on the targets of service providers. For our work, we have used linear regression model trained by concurrent request/response samples for different sized objects between HDFS and RAMDisk file systems for driving coefficients W_i of functions $f_n(m) = \sum_{\mathrm{i=0}}^{\mathrm{j}} W_i \emptyset_i(m)$ where $f_n(m)$ is the regression function for a particular data size range, for example file size of 1MB-5MB. The value of $f(m)$ is set to zero for any object m with file size larger than 5MB, as the performance improvement is not significant for a large file in our case. Detailed experiment settings are described under Experimental Evaluation section. $E(m)$ is the function to calculate expected earnings from the object-caching service based on the number of requests and the size of data transfer. The formula is derived from the current pricing model of Amazon S3. If an object has to be placed in the cache space, the owner of the object pays for the object's cache hits and data transfer. The provider gains revenue from this service. For our work, we set P_r and P_t at 4×10^4 per data cache hit and 1.2×10^{-5} per MB of data transfer. In this thesis, the earning function we have proposed for $E(m)$ is based on the aspect of maximising the revenue gain from cache hits and data transfer. It was inspired by the content distribution model proposed by the Amazon Cloud Front service [214]. Some providers like CloudFlare [215] charge users based on different data caching frequency levels (e.g. from 30 seconds to 2 hours). In fact, there are many aspects that the earning function $E(m)$ can be applied to, such as maximising of number of concurrent users, peak time or off-peak time usage and etc.

$C(m)$ is the transfer cost function; it takes consideration of object data size, how frequently the object is changed, and current system load, including CPU (U_c), RAM (U_r) and network (U_n) utilisations to balance the added load to the system caused by the data migration process. There are different ways of defining the $C(m)$ function by considering various factors. The design of $C(m)$ depends on the requirement of the provider. For example, the provider can add disk I/O rate and power consumption to the cost function as well.

$\alpha, \beta, \gamma, \tau, \psi, \theta, \iota, \kappa$ are constants in the model. These constants reflect the extent of the impact there is on the score of each factor in the system. In such a way, cloud storage vendors are able to construct different caching mechanisms based on their needs by choosing the right values for the constants. For example, a profit-driven cloud storage vendor can set α to be 0 if they are only interested in maximising the profit. In constrast, for a non-profit private cloud storage service vendor, the only important issue is the performance, so β can be set to 0. Other constants $\gamma, \tau, \psi, \theta, \iota, \kappa$ are used to control the intensity of the physical resources (CPU, RAM and network). For example, setting a higher value of ψ will decrease the chance of caching frequently updated objects; setting a higher τ will decrease the change of caching larger files. The LDC model is capable of solving both object data and object metadata caching problems by changing the parameters and inputs. In addition, all coefficients can be learned given enough samples are collected. The methods for deriving those coefficients are beyond the scope of this thesis.

3.5.3. Caching Controller

Figure.3.8 shows the overview of CACSS's caching controller. CACSS's data caching management service consists of three main components: Caching Controller (CC), Metadata Caching Space (MCC), and Object Caching Space (OCS). MCC and OCS are implemented by storage devices or file systems that could provide larger bandwidth and lower latency than ODSS, so that caching selected object metadata in MCC or object data in OCS can accelerate data access to them. All data access requests to the system are logged. The caching controller follows the Local Cache Control (LCC) algorithm. First of all, it launches access-log-processing jobs using the Map-Reduce Access Log Processing algorithm to Hadoop Cluster and to generates the summary result of object accesses. The result is then used to solve the LDC Model and decide which metadata and object data are to be cached. Finally, it triggers the Data Cache Migration (DCM) algorithm to perform metadata records and object data synchronisation.

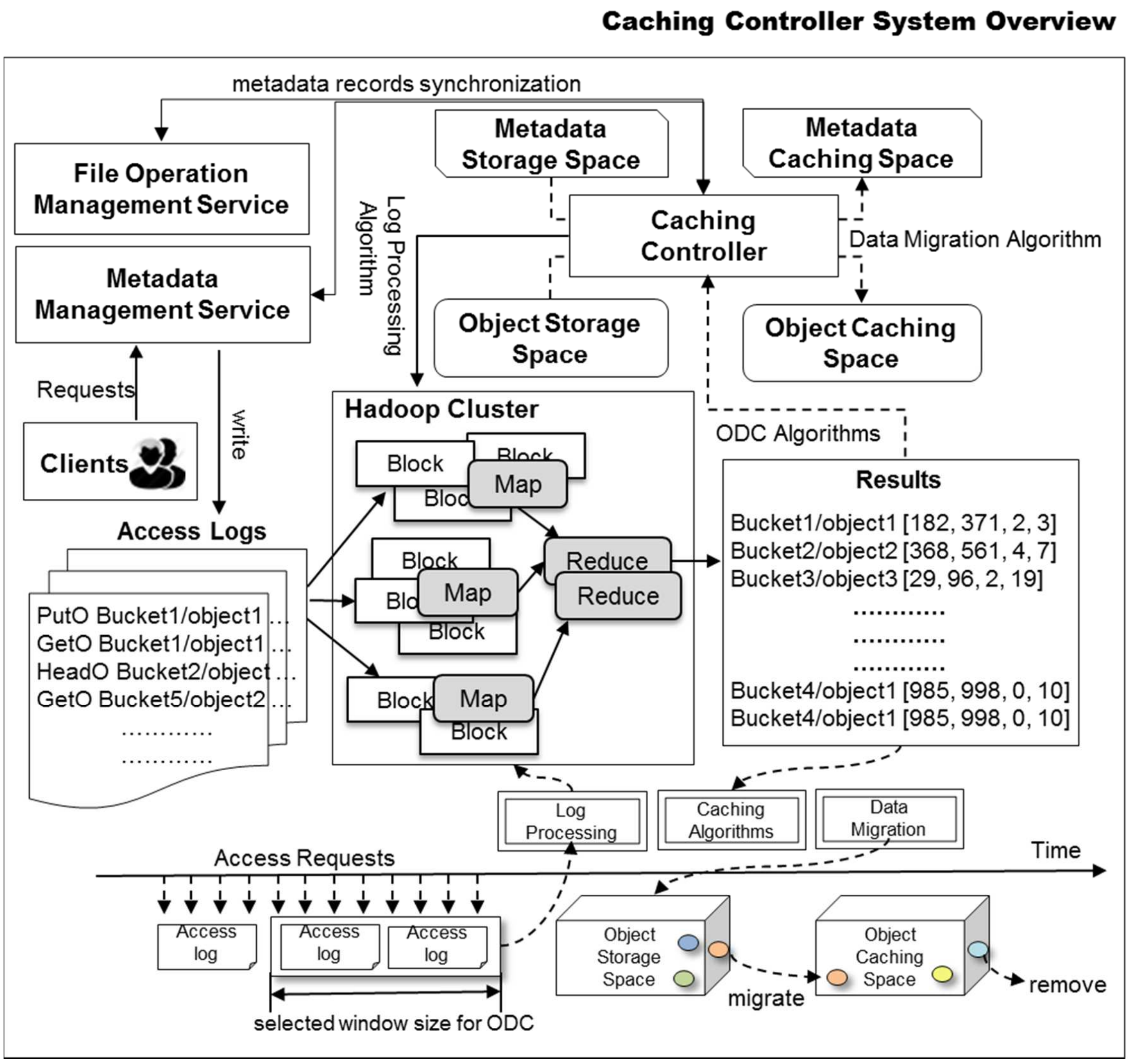


Figure.3.8. Caching Controller System Overview

Local Cache Control (LCC, **Algorithm 1**) defines the main flow of the caching controller. Firstly, it evokes **Algorithm 2** to perform access log analysis with targeted window size. The window size can be chosen based on profiling and experimental results using various window sizes (e.g. 10 minutes, 30 minutes, 1 hour, 12 hours etc.) to determine which size performs the best in terms of cache hits and revenue gain.

Secondly, it then updates the object cache hash table (as shown in Figure 3.9) based on the log analysis results. Next, it creates and solves the LDC instances. Finally, it calls **Algorithm 3** to perform the data migration process.

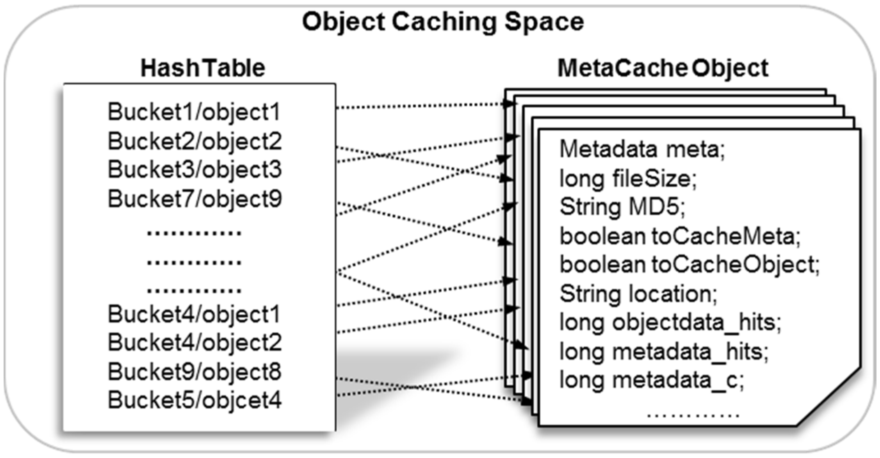


Figure 3.9: Structure of Object Caching HashTable

Algorithm 1: Local Cache Control

Inputs:

C:	cached hash table <key, ObjectCache>
t_s:	start time
t_e:	end time
MAX:	maximum number of objects to cache

```
1.  Begin
2.  obtain processed logs using Algorithm 2 with selected window size between $t_s$ and $t_e$
3.  create new object caching hash table C'
4.  foreach (log in logs)
5.      set key to be combined key of bucket name and object key from log
6.      If C' contains key, update object cache c with latest log otherwise create new object
        cache c from log
7.  end for
8.  update object cache in C with items in C'
9.  remove objects from C between items below threshold
10. if (size(C)> MAX), then
11.     create ranked cache object array list A from C up to MAX
12. else
```

```
13. convert hash table $C$ to object cache array list $A$
14. $C$ =Solve LDC model for object data caching with $A$ as input
15. $C$ =Solve LDC model for object metadata caching with $A$ as input
16. migrate data for $C$ using Algorithm 3
17. End
```

Our system retains tracking records of all the accesses in log files stored in the HDFS cluster. The log file contains traces of the request method type, accessed bucket name, object key, date, time, and other information. **Algorithm 2** can be used to generate summaries and statistics of data access history for various purposes such as for data caching, billing, and invoicing. It leverages the MapReduce framework, providing a scalable way of processing large amount of log files across clusters of machines.

Algorithm 2: Map-Reduce Access Log Processing

Inputs: log files

Map(String key, String value)

```
1. //key: line number (not in use)
2. //value: represents a line of the log file with format: //<ReqMethod
   BucketName/ObjectKey>
   //e.g. "GetObject bucket1/objkey1"
3. set n_obj=0, n_meta=0, n_oc=0, n_mc=0
4. emitKey=parseEmitKey(value) //get bucket name and object key from each line
5. if (is GetObject request)
6.     set n_obj=1, n_meta=1
7. else if (is PutObject request)
8.     set $n\_oc$=1, $n\_mc$=1
9. ……//update request counters
```

```
10. end if
11. set emitValue with all request counters
12. emit(emitKey, emitValue)
```

Reduce(String key, iterator values)

```
1.  //key: emitKey, i.e. BucketName/ObjectKey
2.  //values: list of output values for each emitKey
3.  set n_obj=0, n_meta=0, n_oc=0, n_mc=0
4.  foreach (value in values)
5.      //parse and update request counters
6.      n_obj= n_obj+parse_n_obj(value)
7.      n_meta=n_meta+parse_n_meta(value)
8.      ……
9.  end for
10. set emitValue with all request counters
11. emit(key, emitValue)
```

Algorithm 3 is used to migrate and synchronise the objects that have been as retainable in OCS. It can be called either periodically or on demand.

Algorithm 3: Data Cache Migration

Inputs:

C: cached hash table <key, ObjectCache>

```
1. Begin
2. Initialise latestCacheFiles[]
3. foreach (key in C)
4.     c=C(key)
5.     meta=fetchObjectMetadata(key)
```

```
6.      c.setMetadata(meta)
7.      if (c.toObjCache())
8.          latest_etag= meta.getEtag() //etag represents the file checksum
9.          cached_etag=c.getEtag()
10.         if(cached_etag==latest_etag)
11.             latestCacheFiles.add(c.getCacheFile())
12.         else
13.             file_id=randomUUID()
14.             etag=copyfile(meta.getFile(), new File(file_id))
15.             c.setCachedFile(file_id)
16.             c.setEtag(etag)
17.             latestCacheFiles.add(c.getCachedFile())
18.             updateCachedHashTable(C, c)
19.         end if
20.     end if
21. end for
22. allCacheFiles[]=listAllFiles()
23. foreach (file in allCacheFiles)
24.     if(!latestCacheFiles.contains(file))
25.         delete file
26.     end if
27. end for
28. End
```

3.6. Implementation

After considerable research and experimentation, HBase was chosen as the foundational MSS storage for all object metadata. HBase is highly scalable and delivers fast random data retrieval. Its column-orientated design confers exceptional flexibility in the storing of data.

We chose Hadoop DFS (HDFS) as the foundational storage technology for storing object data in ODSS. Hadoop also supports MapReduce framework [104] and can be used for executing computation tasks within the storage infrastructure. Although there is a single point of failure at the NameNode in HDFS's original design, many research studies have been carried out in order to build a highly available version of HDFS NameNode; for instance: AvatarNode [216]. Every file and block in HDFS is represented as an object in the NameNode's memory, each of which occupies about 150 bytes. Therefore the total memory available on NameNode dictates the limitation of the number of files that can be stored in the HDFS cluster. By separating object metadata and object data, CACSS is able to construct an adaptive storage infrastructure that can store unlimited amounts of data using multiple HDFS clusters, while still exposing a single logical data store to the users (Figure 3.10). The Linux Ram Disk technique is used to utilise server RAMs in Tomcat Clusters serve as the Object Caching Space.

Multi-Region Support

The design and architecture of CACSS are based on the principles of scalability, performance, data durability, and reliability. Scalability is considered in various aspects including the overall capacity of multi-region file metadata and file storage, as well as throughput of the system. Taking another perspective, the implementation of CACSS consists of a region controller and multiple regions (Figure 3.11 and Figure 3.10).

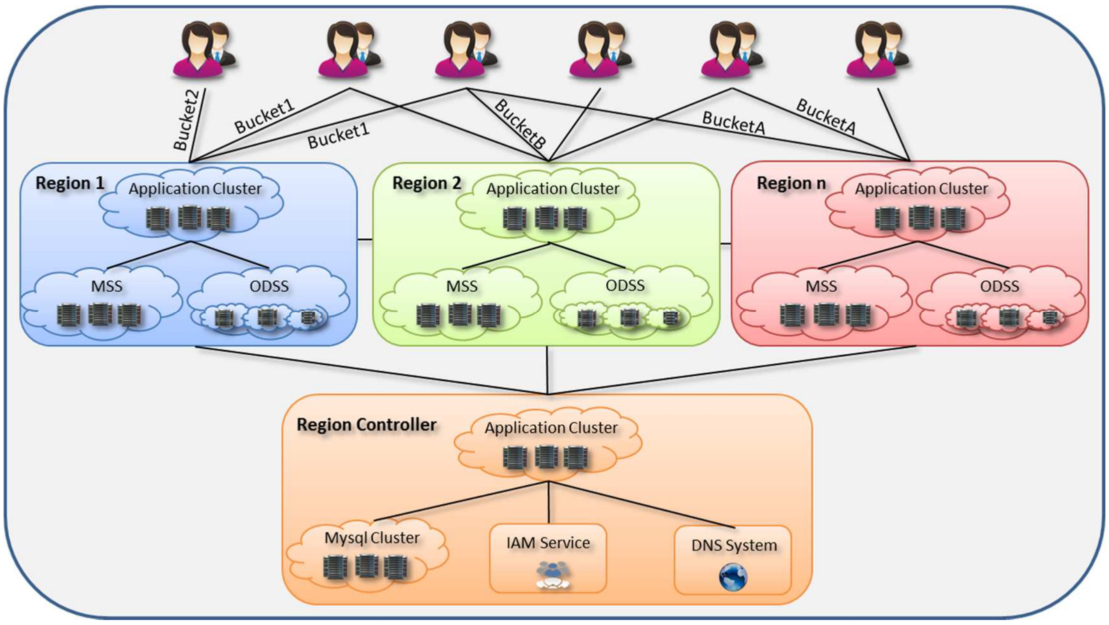


Figure 3.10: Implementation of CACSS

A Tomcat cluster is used as the application server layer in each region. It is easy to achieve high scalability, load balancing, and high availability by using a Tomcat cluster and configuring with other technologies, such as HAProxy and Nginx [217, 218]. For security concerns, the HBase cluster and HDFS clusters are not allocated any public IPs; they can only be accessed through their region's Tomcat cluster and by private IPs.

CACSS uses both header and query string authentication: the user has to create the string, signed by concatenating the request verb with canonicalised headers and the resource that the request is targeting. The region controller has a MySQL cluster for storing various data such as user account information and billing and invoice details.

A bucket can be created in one of the regions. At the same time, a DNS "A" record is also inserted into the DNS server. This mapping ensures that clients will send a hosted-style access request of the bucket and the object to the correct region; examples are shown in Table 3.2. Each region is consistent with a Tomcat cluster, an HBase cluster, and a set of HDFS clusters. The object data is stored in one of the HDFS clusters in the region. The object key and metadata are stored in the region's HBase cluster. It is always important to consider that any access to a bucket or object requires access rights to be checked. In CACSS, each request goes through its region first; if the requested bucket or object is set to be public (to "AllUsers"), there is no need to communicate with the region controller. If it is not set as public, it consults the region controller to perform the permission check before making a response. The region controller, which includes a MySQL cluster, keeps records of all the requests and maintains user accounts and billing information. A Domain Name System (DNS) serves to map the bucket name to its corresponding region's Tomcat cluster IP. The region controller can also connect to the existing Identity and Access Management service (IAM) to provide more sophisticated user and group management.

Bucket name	Region	DNS A record
bucket1	Region1	bucket1-> 66.23.233.59 (Region1 Tomcat cluster)
bucket2	Region2	bucket2-> 155.198.140.14 (Region2 Tomcat cluster)
bucket3	Region4	bucket3->66.23.233.86 (Region4 Tomcat cluster)
……	……	……

Table 3.2: Example of buckets and region mappings using a DNS system.

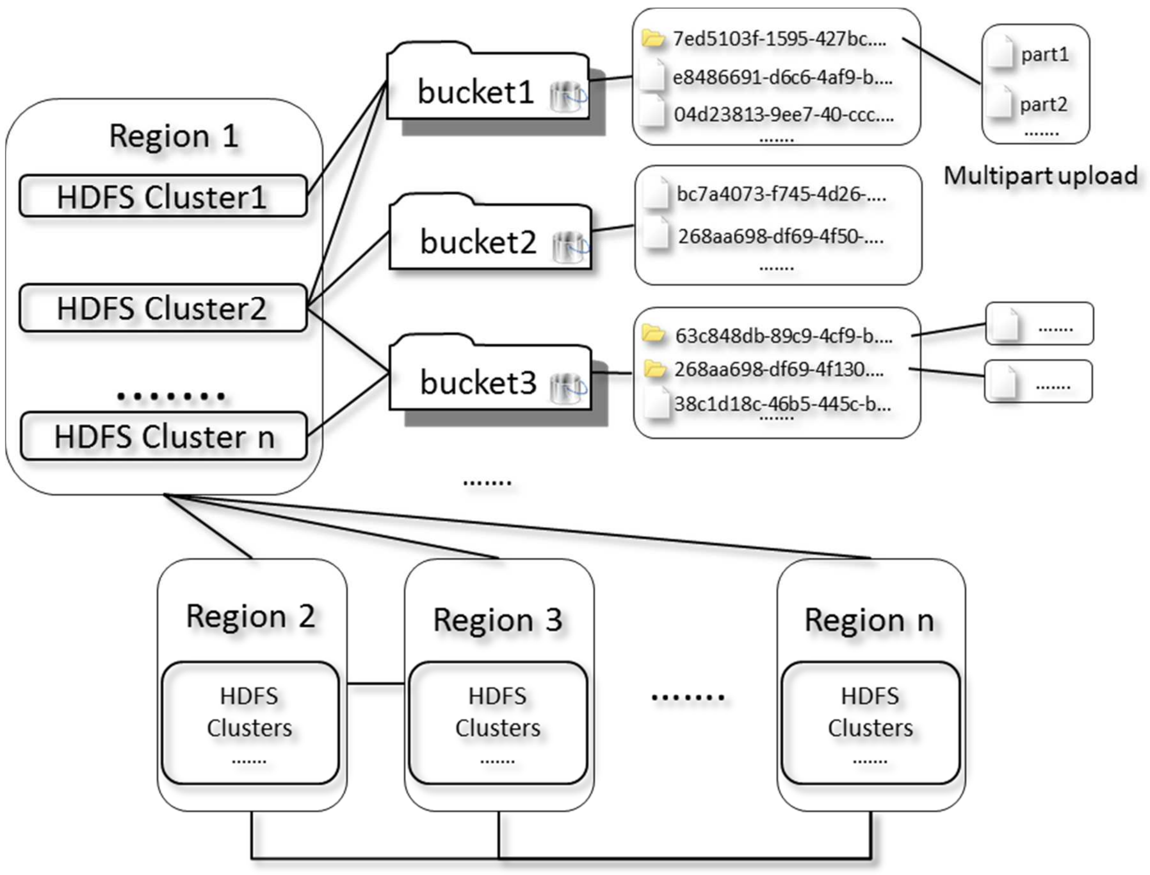


Figure 3.11: Implementation of multi-region HDFS clusters for storing buckets and contents of objects

CACSS also adopts other useful features of HDFS, including no explicit limitation on a single file size and no limitation on the number of files in a directory. In CACSS, most of the objects are stored in a flat structure in HDFS. Each object's file name under HDFS is a generated UUID to ensure uniqueness.

The implementation of CACSS does not need to rely solely on HDFS. The separation of file metadata entirely from file content enables CACSS to adapt to one or even multiple file systems, such as GPFS or Lustre. It is now deployed as a service under IC-Cloud platform [173], and is expected to work with a variety of distributed file systems through POSIX or their APIs without much effort.

3.7. Experimental Evaluation

The server environment setup for the first two sets of experiments has been summarised in Table 3.3.

Experiments	**Server Type**	**Memory**	**Number of Servers**	**Network**
CACSS Performance	m2.xlarge (EC2)	17.1GB	1	Shared 500Mbps
	m1.large (EC2)	7.5GB	3	Shared 500Mbps
Profiling for Local Data Caching Model	4 CPU cores (IC-Cloud)	8GB	1	Shared 100Mbps
	2 CPU cores (IC-Cloud)	4GB	3	Shared 100Mbps

Table 3.3: Experiment Environment Setup Summary

3.7.1. CACSS Performance

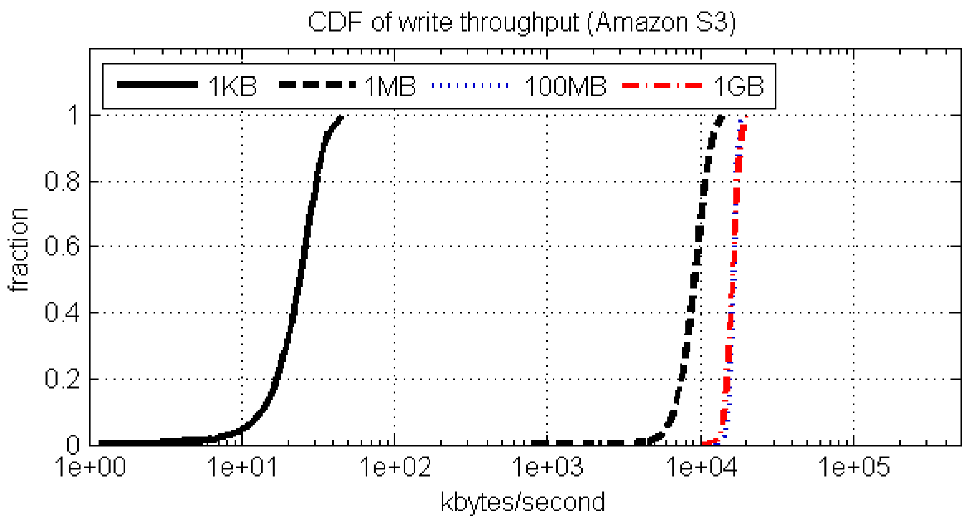
Experiment Setup

The purpose of the first set of experiments was to evaluate the performance of CACSS and compare CACSS with Amazon S3 under similar hardware and network environments. We constructed and configured a CACSS cluster on top of Amazon EC2 instances. We used JetS3t [219], an open source Java S3 library and toolkit, and Siege [85], an http load testing and benchmarking tool for our experiments.

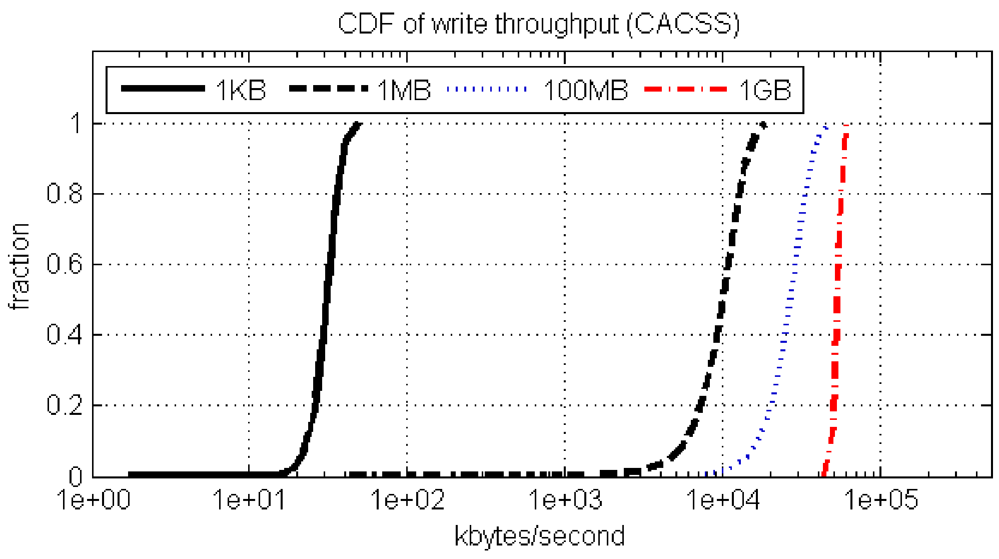
We used one m2.xlarge instance, with 17.1GB of memory and 6.5 EC2 Compute Units, to run MySQL, HDFS NameNode, HBase Hmaster, allocated RAMDisk space, and Tomcat with the CACSS system deployed. Three m1.large instances, each with 7.5GB memory and 4 EC2 Compute units ran HDFS DataNodes and HBase Regionservers. Each of these instances was attached with 100GB of storage space. Another two m1.large instances were configured with the same experiment program, but different access end points. We refer to these two instances as "S3 test node" and "CACSS test node". All the instances we used were rated as "Moderate" network performance by Amazon.

Throughputs Performance Comparisons

Some evaluation of the performance of Amazon EC2 and S3 has been carried out previously by Garfinkel [220]. In this experiment, a similar method was used to evaluate the overall throughput of CACSS and Amazon S3. Figure 3.12 and Figure 3.13 respectively illustrate the write and read throughputs of Amazon EC2 to Amazon S3, and of EC2 to CACSS, based on our experiments. Each graph contains traces of observed bandwidths for transactions of 1KB, 1MB, 100MB and 1GB. Both Amazon S3 and CACSS perform better with larger transaction sizes, because smaller size files would require more overhead transaction. For files larger than 1MB, the average speed of CACSS transaction is higher than Amazon S3; this is likely due to underlying hardware differences between Amazon EC2 and Amazon S3, such as hard drive RPM and RAID levels. The obtained performance results are unlikely to be repeatable due to the sharing of underlying hardware architecture with other users, since workloads on Amazon's physical servers and network may be different next time anyone attempts to test this out [221].

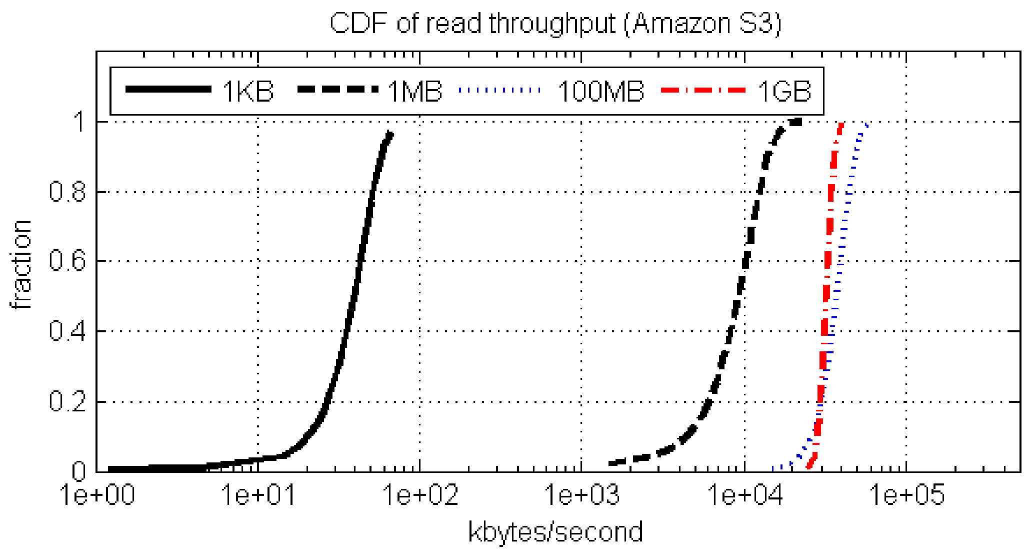


(a)

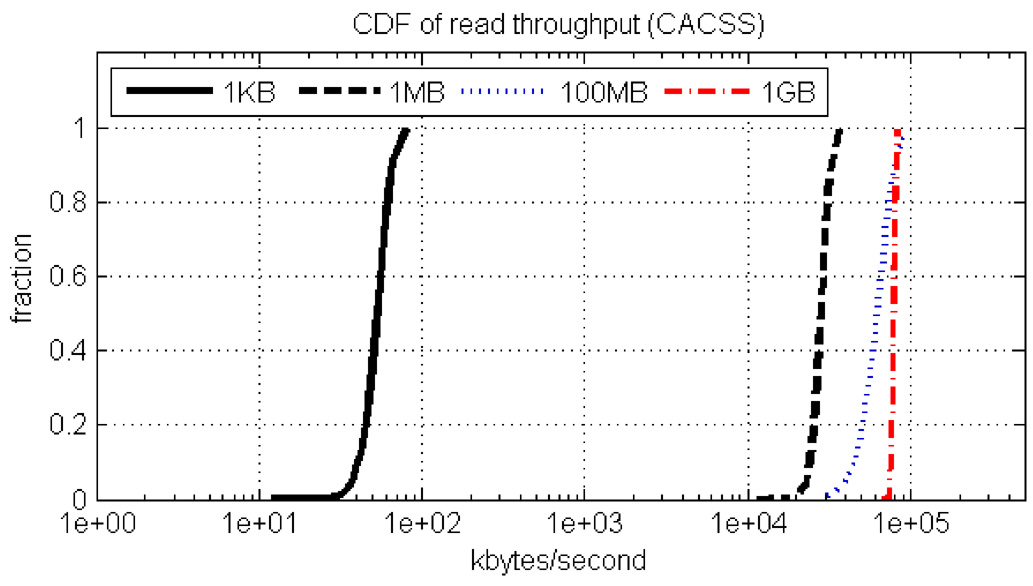


(b)

Figure 3.12: Cumulative Distribution Function (CDF) Plot for Writing Transactions of Various Sizes from EC2 to Amazon S3 and CACSS.



(a)



(b)

Figure 3.13: CDF Plots for Reading Transactions of Various sizes from EC2 to Amazon S3 and CACSS

Performance of metadata management

Object listing and searching is a common and frequent operation in cloud storage systems. Amazon S3's List Objects operation will only support up to 1000 objects to be returned at a time. Thus, we could not evaluate fully its object metadata service performance. However, we were able to run some tests to evaluate CACSS's metadata management. We ran a List All Objects operation after every 1,000 Put Object operations. All of the operations were targeted to the same bucket. Each Put Object operation used an empty file, because for this experiment we were only interested in the performance of the metadata access.

A total number of 500,000 Put Object requests was performed. Figure 3.14 shows a scatter graph of response times for each Put Object, with respect to the total number of objects in the bucket. The result shows an average response time of 0.007875s and a variance of 0.000157s for each Put Object operation. This indicates that the response time is pretty much constant no matter how many objects are stored in the bucket. Figure 3.15 illustrates the response time of each List All Objects operation with respect to the total number of objects contained in the bucket. There are several peaks in the graph marked with a green circle, which are caused by sudden network latency between Amazon EC2 instances during these times as indicated. Otherwise, the overall result shows a linear relation between the response time and the total number of objects.

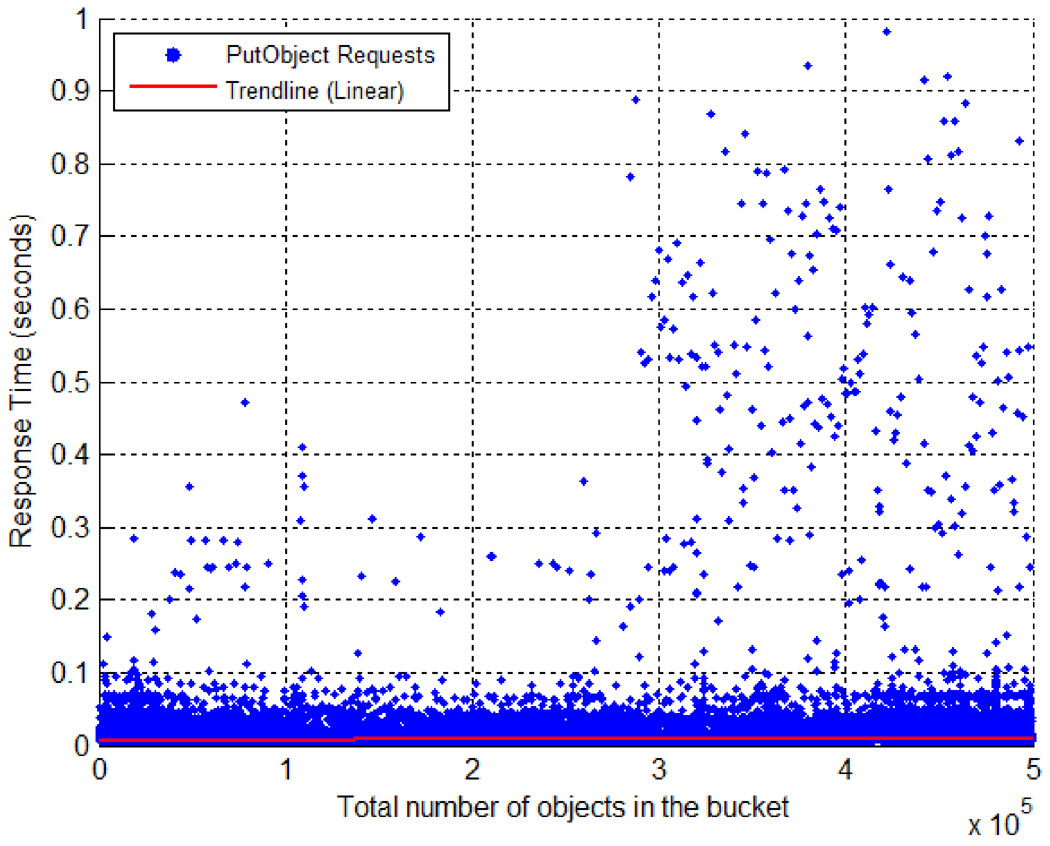


Figure 3.14: Response Times of Put Object Requests Correspond to the Total Number of Objects in the Bucket.

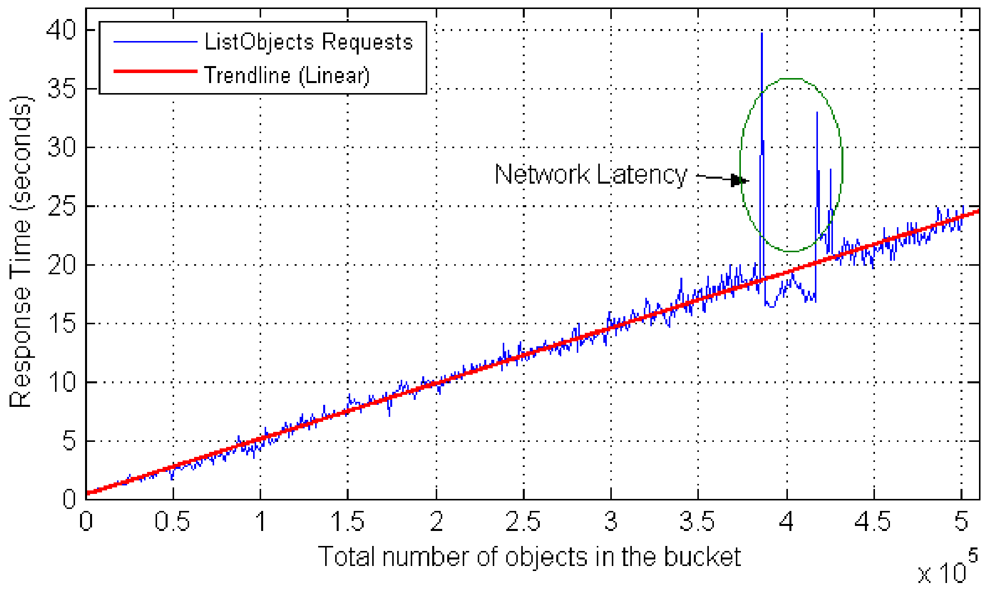


Figure 3.15: Response Times of List All Objects Requests Correspond to the Total Number of Objects in the Bucket.

3.7.2. Profiling for Local Data Caching Model

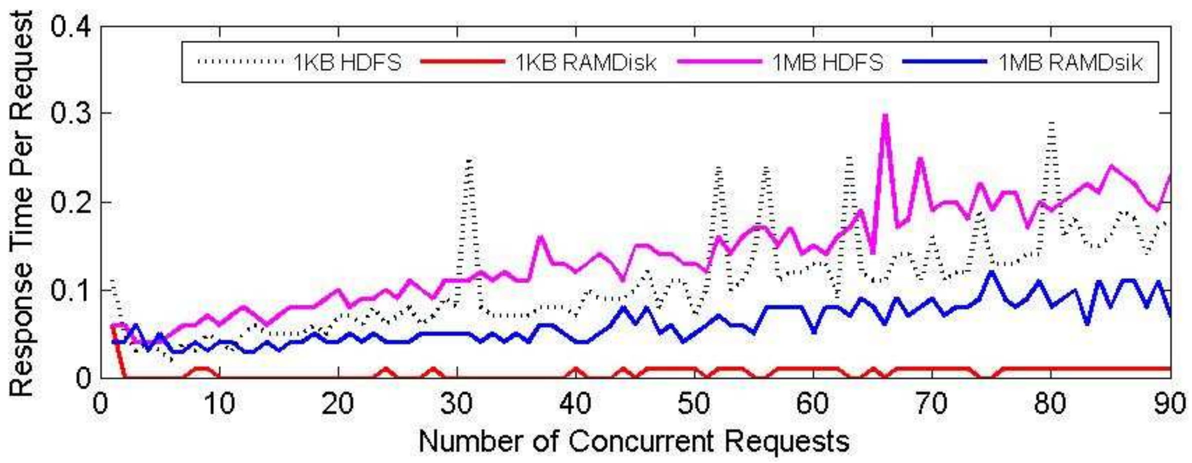
Experiment Setup

A CACSS cluster was constructed and configured on top of IC-Cloud. We used JetS3t [219], an open source Java S3 library and toolkit, and Siege [222], an http load testing and benchmarking tool for our experiments. Four virtual machines (VMs) were used to create a CACSS cluster. One VM with 8GB of memory and four CPU cores was used to run MySQL, HDFS NameNode, HBase Hmaster and Tomcat with the CACSS application and allocated RAM disk space. The other three were each configured with 4GB memory and two CPU cores to run HDFS DataNodes and HBase Regionservers. The network between each virtual machine is 100Mbps.

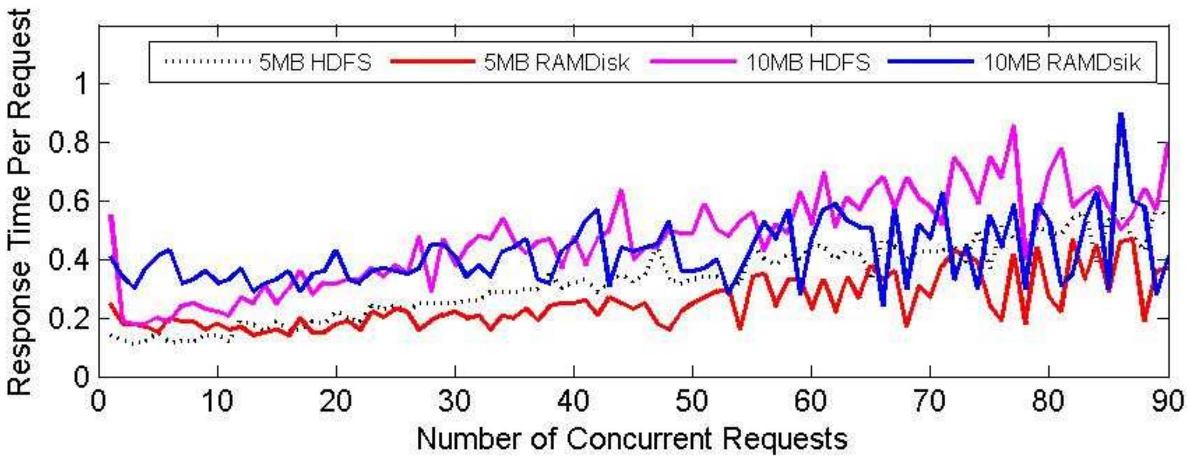
Performance of Concurrent Requests

In this experiment, we configured an identical set of objects with different sizes to be stored both in HDFS cluster and RAMDisk space. All of the objects had been set to public to enable access without permission checking and the needs of JetS3t library. Siege, as a benchmarking tool was used for this test.

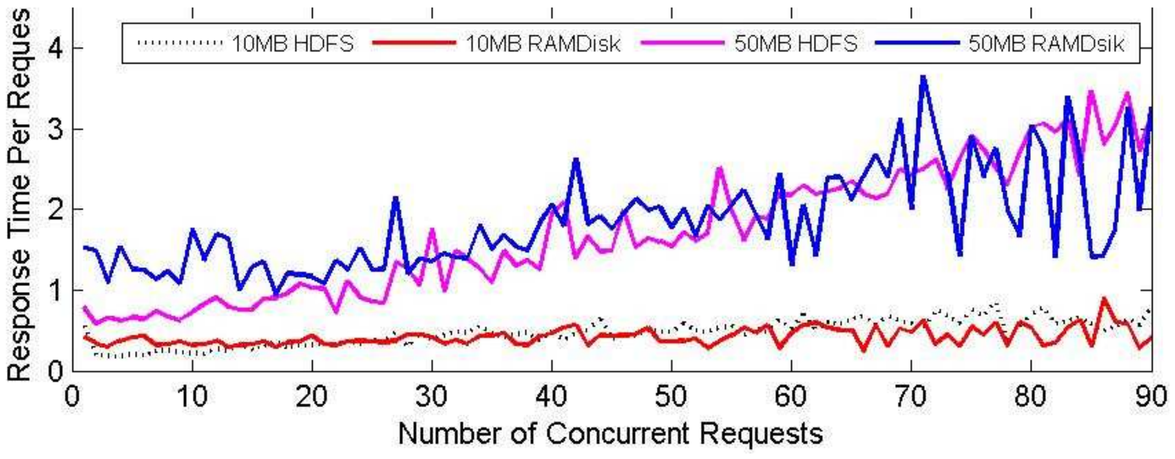
A number of concurrent object retrieving requests were performed on the same objects stored in RAMDisk and HDFS clusters respectively, with various sizes from 1K to 50MB. Corresponding request response times are shown in Figure 3.16. The results show significant response times and stability improvement for requests to objects stored in RAMDisk with sizes less than 5MB. For file sizes smaller than 1KB, the average response time for retrieving data from RAMDisk can be 20 times faster than the response time for retrieving data from HDFS. The main reason is that modern distributed file systems such as HDFS and GPFS [223] are not optimised to manage small files. For file sizes larger than 5MB, there is a negligible difference in the response time between both systems. It is likely this was caused by the network bottleneck. To summarise, the results from the experiment can be used as profiling data to configure the parameters and coefficients in the aforementioned LDC model.



(a)



(b)



(c)

Figure 3.16. Response time of concurrent requests compared between HDFS and RAMdisk read requests performance

3.7.3. Effectiveness of the Local Data Caching Model

This set of the experiments was conducted to evaluate how CACSS's data caching management performs under real-world conditions and scenarios. A custom simulator was developed to simulate object access requests based on the Wikipedia traffic statistics data in December 2012. On average, approximately 10 million different files from various Wikimedia projects are accessed per hour. Due to our limited computing resources, we chose to extract only a portion of requests for accessing JPEG image files.

We configured the parameters and set the objective function of the LDC model to maximise the provider's revenue. Following that, we used pricing information from Amazon S3 and CloudFront [224] to act as guidelines for the simulation of revenue gain. The assumption is that owners have the option to choose whether to enable the object caching service on the bucket level; if any object in the caching-enabled bucket is decided by the LDC model to be placed in the cache space, the owner of the object pays for the object's cache hits and data transfer. The owner will benefit from the superior performance of the caching service while the provider will subsequently gain revenue from the service exchange. We set the usage pricing to be $4x10^{-4}$ per data cache hit and 1.2x10-5 per MB of data transfer.

We used the LCC algorithm to determine the data caching placement specification. The result was used to determine the most appropriate file to be placed into the cache space. For comparison, three approaches were simulated:

- LCC-1DAY: LCC algorithm was used to apply the LDC model based on the past object data requests of one day.
- LCC-1HOUR: LCC algorithm was used to apply the LDC model based on the object data requests in the previous hour.
- TOP-1HOUR: this method used one of the current leading caching policies, based on ranking, to identify the most frequently accessed object files in the previous hour, before copying them into a cache until the cache is full.

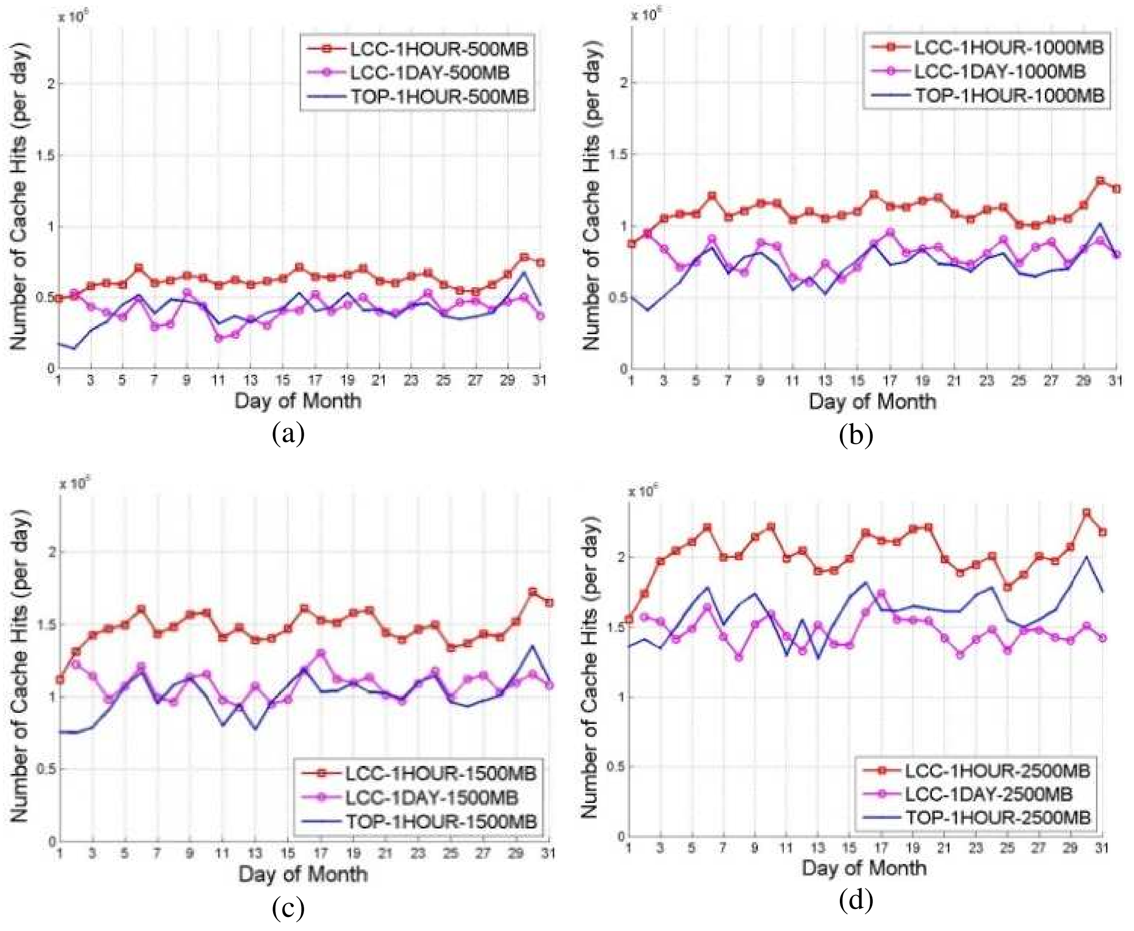
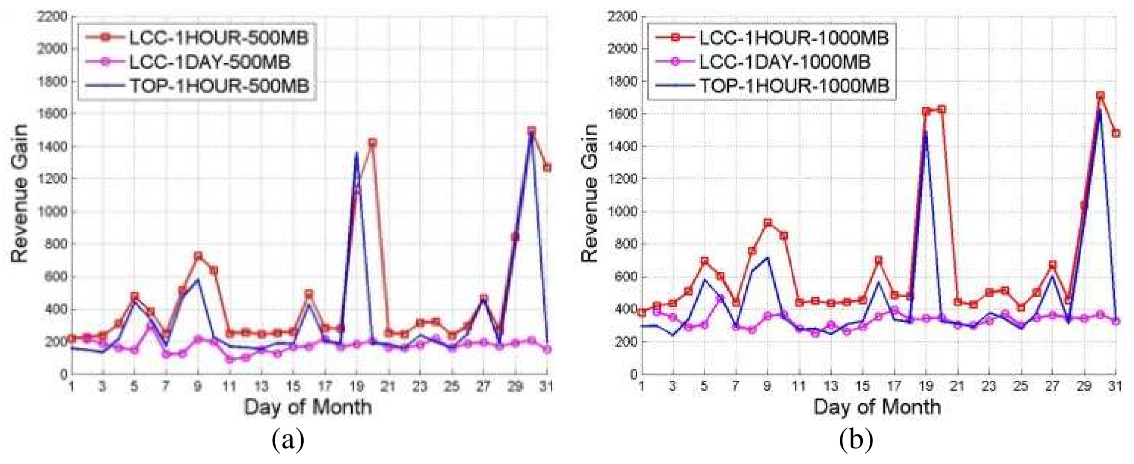
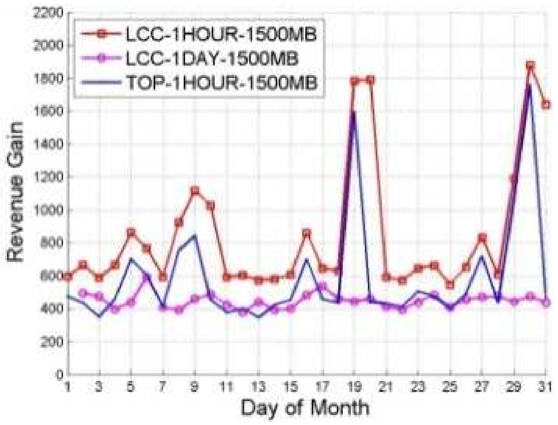
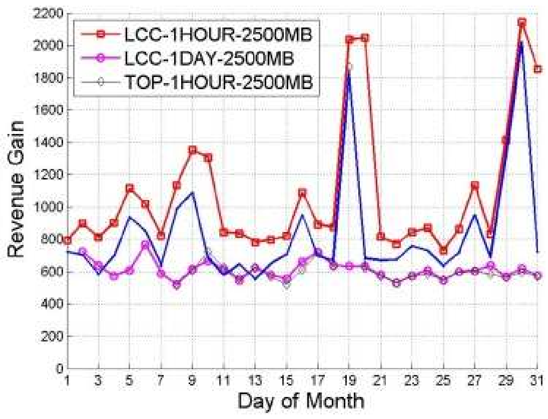


Figure 3.17. Comparison of total number of cache hits with different approaches and cache sizes





(c)



(d)

Figure 3.18. Comparison of simulated revenue gain with different approaches and cache sizes

Figure 3.17 depicts the result of the number of cache hits between different approaches with various sizes of total cache spaces; for example, from 500MB to 2500MB. The number of cache hits of all files is determined at the end of each day. LCC-1DAY requires one day of data access traces, so the first day is an invalid outlier data point that we, therefore, consistently removed from our data collection. Results show that LCC-1HOUR performed the strongest out of all the approaches. On average, it generated 43% more cache hits than the rank-based approach (TOP-1HOUR). Figure 3.18 indicates how much revenue the provider can earn if such a data caching service were to be charged by usage to users. The results show that the rank-based approach only generates more revenue than LCC-1HOUR on the 19th day (as shown in Figure 3.17a). Apart from that, the LCC-1HOUR performed the best, and on average 40% better than the rank-based approach for generating provider revenue. The peaks presented in Figure 3.18 are caused by a sudden increase in the access to large files that kept in the cache space as for the same number of cache hits, since larger files generate more revenue on data transfer fees than smaller files. As the parameters are set to maximise the provider's revenue in this experiment, the LDC model is able to choose the best files to put in the cache at the right times such that the overall revenue of the sum of cache hits and network transfer can be maximised.

Both cache hits and revenue-gain results demonstrate that the effectiveness of the LDC model can be utilised to improve service performance and increase total revenues for cloud service providers.

3.8. Summary

In this chapter, we have presented the design and implementation of CACSS, a cloud storage system based on the generic principles of scalability, performance, data durability, security, reliability, and storage efficiency. CACSS performance was found to be comparable to Amazon S3 in formal tests, with similar read/write capabilities. Although other features were difficult to compare directly, CACSS performance was highly adequate in terms of Put Object and List All Object requests. We also demonstrated how “performance as a service” can be constructed and implemented. The proposed Local Data Caching model can be adjusted and used for utilising a provider’s available caching resources to achieve different goals—such as increasing revenue or enhancing performance. We have made the comparison to one of the current leading caching policies based on ranking. Our experiments show that the LDC model outperforms the rank-based approach in the number of cache hits and results in average revenue gains of 40%.

In the next chapter, we introduce the full architecture of Wiki-Health platform and demonstrate how CACSS is used as the key component of Wiki-Health for managing unstructured data.

4. Wiki-Health: A Big Data Platform for Health Sensor Data Management

4.1. Introduction

Advances in smartphone-compatible technologies have resulted in new ways of gathering information through an array of embedded sensors such as cameras, microphones, accelerometers, proximity sensors and GPS. These advances have also had a significant impact on health information gathering aides: this revolution in sensing devices has expanded into arena of healthcare monitoring, where the latest generation of health sensors can now easily connect to smartphones to track a number of important health-related parameters. Existing wearable fitness sensors such as the Fitbit tracker [37], Zephyr heart rate monitor [38], and Withings blood pressure monitor [40] have already been available on the market for some time, and allow users to automatically collect data on their walking steps, activity levels, food intake, heart rate and blood pressure. All devices which can be readily synched with smartphones, tablets and iOS devices via dedicated apps. These technologies have provided more efficient and convenient ways to collect personal health information. As a result, the scale and richness of mobile sensor data now being collected and analysed is rapidly growing. However, it is still difficult for users to manage or utilise the data they collect in practical and meaningful ways. From the health service provider's perspective, such massive growth of these big health sensor data creates both data manageability and collaboration challenges.

In the previous chapter, we focused on tackling one of the emerging challenges in managing the unstructured and semi-structured types of sensor data by introducing a multi-tier cloud storage system – CACSS. In this chapter, we present the design and implementation of Wiki-

Health platform, which will take advantage of cloud computing and cloud storage capabilities for personal health sensor data management. Wiki-Health is not only designed to provide a unified solution for collecting, storing, tagging, retrieving, searching and analysing personal health sensor data of any type or format, but also allows users to reuse and remix data, along with analysis results, and analysis models to make health-related knowledge discovery more available to individual users at a massive scale.

4.2. Related Work

The pervasive use of mobile phones and the rising adoption of sensing devices enabling people to collect data at any time or place are leading to a torrent of sensor data. A considerable number of studies have so far been undertaken on managing sensor data storage and queries. Traditional sensor management systems such as TinyDB [2], Aurora [3], and COUGAR [4] offer limited support for collaboration and data processing abilities that are no longer able to cope with such the increased demand. Meanwhile, cloud computing is developing into a promising resource to deliver infrastructure, platforms and applications as services in a scalable and low-cost manner. Implementing cloud computing technologies appropriately can aid service providers in improving the efficiency of operations, sharing information, improving collaboration, and managing expenditures. For example, Sensor-Cloud [102] enables an infrastructure on which users can share different physical sensors with others through the implementation of virtual sensors. Commercial sensor network platforms such as Xively [9] and ThingSpeak [10] have taken off in recent years and provide an online scalable sensor data management platforms that allow users to connect sensor devices and applications through a web-based application programming interface (API). However, none of these platforms yet provide sufficient support for running computationally intensive analysis algorithms.

4.3. Overview

As shown in Figure 4.1, the functionalities of Wiki-Health can be grouped into three categories: Data Collection, Data Storage and Data Services. Wiki-Health is created with the intention of bridging the gap between the users who collect the data and health professionals who provide data analysis services. This will allow both parties to effectively utilise all of the collected data while reducing the complexity of dealing with its diversity. The functionalities and associated tools of Wiki-Health make it feasible for use as a test bed for undertaking health-related experiments and delivering novel analysis algorithms.

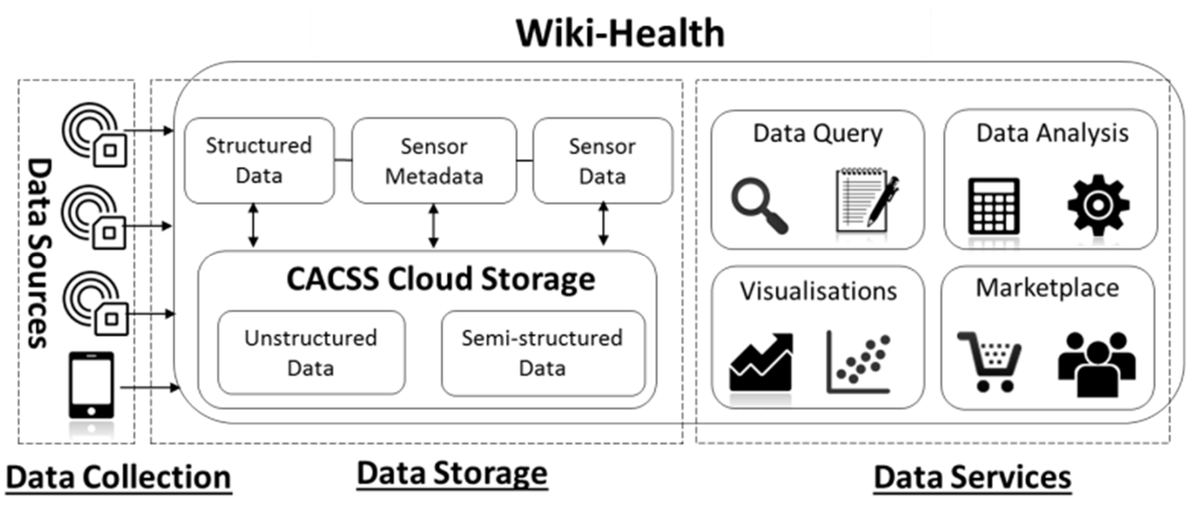


Figure 4.1: The overview of Wiki-Health platform.

4.4. Platform Design

Figure 4.2 illustrates the overall architecture of Wiki-Health. The Wiki-Health system is designed in three logical layers: data storage, query and analysis and application.

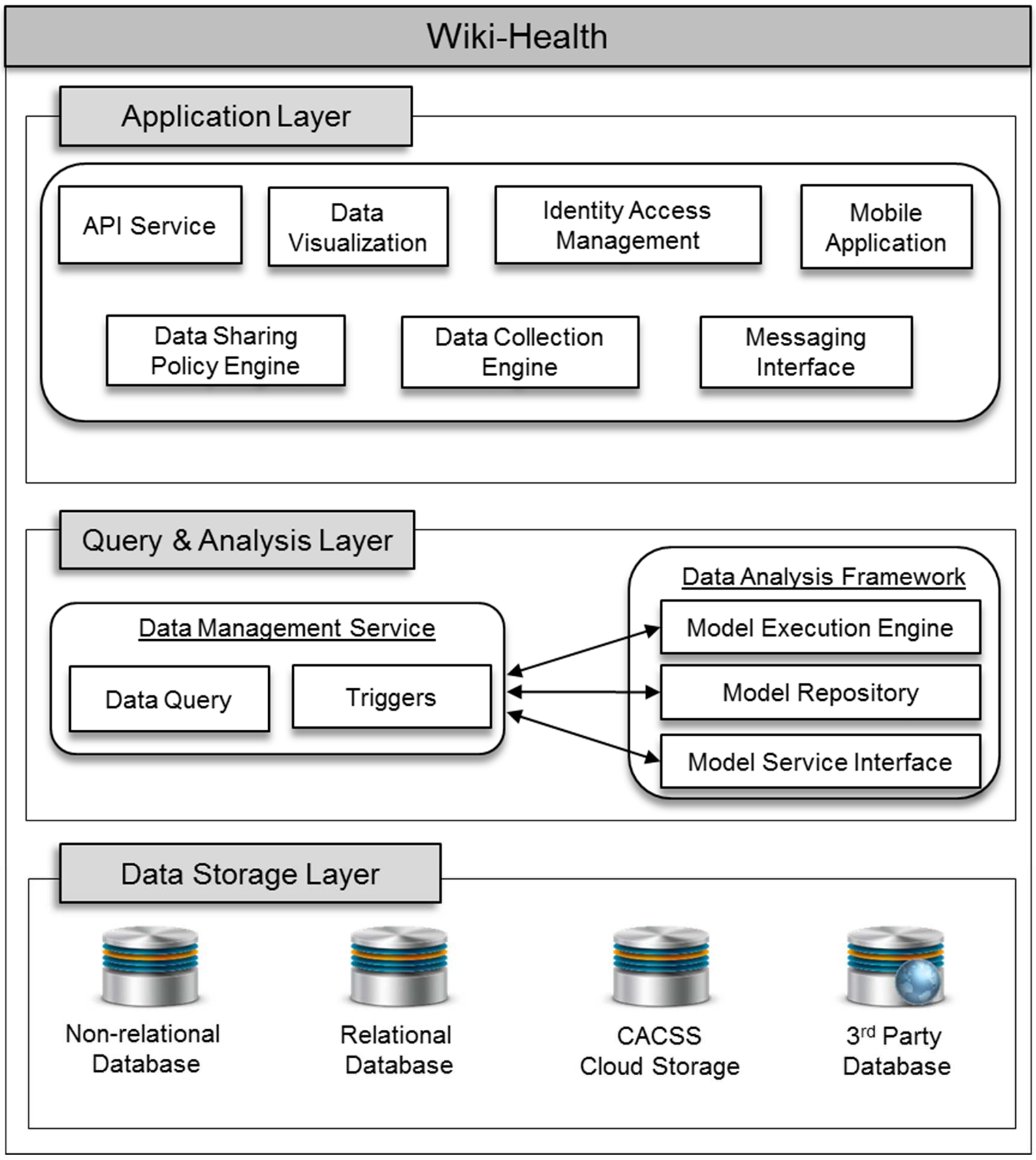


Figure 4.2: The architecture of Wiki-Health

4.4.1. Data Storage Layer

The data storage layer is a logical tier that contains a non-relational database, a relational database, CACSS cloud storage [225, 226], and data sources from third party databases.

The aim of the data storage layer is to provide a unified storage space for storing and handling different formats of health sensor data. For example, health sensor devices such as ECG and Electroencephalography (EEG) employ a number of data channels and possesses significantly higher temporal resolution than common environmental sensors like temperature and humidity. Such health sensor data, therefore, demands a scalable, fast and efficient system in order for it to be stored and processed. Traditional relational databases are designed for efficient transaction processing of small amounts of information in a large database. The data sets stored in these databases have no pre-defined notion of time unless timestamp attributes are explicitly added. Non-relational databases store data in a key-value structure and use looser consistency models than traditional relational databases, thus providing higher scalability and better performance [227-232]. We chose HBase [71, 232], which is designed to provide fast real time read/write data access, as the foundational non-relational database storage for storing the structured sensor data in Wiki-Health.

Storing the unstructured sensor data such as medical imaging files in a cloud storage system can potentially help health organisations and users in managing large amount of data and improving collaborations. As presented in the previous chapter, CACSS [225, 226] is a generic cloud storage system we have developed to help users reduce the costs, complexities and risks in managing large unstructured and semi-structured data growth. It allows access to the data through the S3 compatible interface. In Wiki-Health, the CACSS cloud storage system is used as default storage space for unstructured and semi-structured sensor data.

Sensor metadata can be understood as representing descriptions of data and contains background information such as type, accuracy and location of each sensor. Sensor data refers to the actual measurements of sensors. As shown in Figure 4.3, a hybrid data storage model is used in Wiki-Health to achieve high performance in data access and operations. Sensor metadata and sensor data are completely separate. The system uses MYSQL relational database to store all user information, in addition to sensor metadata such as sample rate, data format, sensor type and other Wiki-Health structured data. All sensor readings, sensor data tags, and other types of sensor data are stored in the non-relational database and CACSS Cloud Storage System.

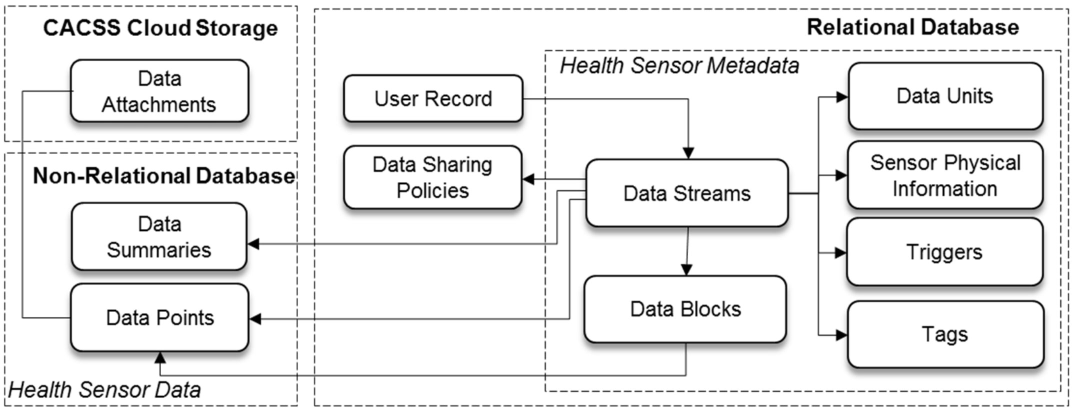


Figure 4.3. The hybrid data storage model

In Wiki-Health's hybrid data storage model, each data stream maps and holds all the information generated by the actual health sensor device. A single data stream can have many data units. Such data units are useful for health devices that provide multiple channel readings simultaneously. For example, a data stream for a blood pressure device contains data units for systolic and diastolic pressure and heart rate measurements. Data points contain sensor reading data, data attachment indexes, tag mappings, block mappings and other user defined data. They are stored as a collection of blocks addressed by an index using the data stream ID together with a timestamp (as shown in Figure 4.4). Data blocks are used to group a set of data points together so that required data can be found more quickly and accurately. Data tags are designed and implemented for developers and users to be able to add more dimensions to the data. They can be used to store and identify the correct event, stimulus or cause associated with a corresponding sequence of sensor data. Data summaries can be used for storing a summary or intermediate analysis result to improve the response time required for retrieving certain analysis and query results. A data stream can have multiple triggers which hold action and condition information and are used to perform actions when a defined condition is reached. For example, such an action can be configured to alert a caregiver when a certain threshold has been reached or an unusual reading is discovered.

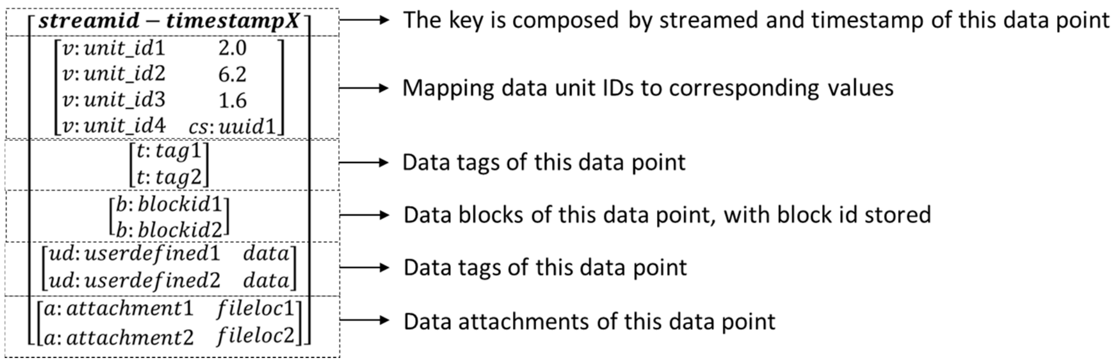

streamid − timestampX
v: unit_id1 2.0
v: unit_id2 6.2
v: unit_id3 1.6
v: unit_id4 cs: uuid1
t: tag1
t: tag2
b: blockid1
b: blockid2
ud: userdefined1 data
ud: userdefined2 data
a: attachment1 fileloc1
a: attachment2 fileloc2
The key is composed by streamed and timestamp of this data point
Mapping data unit IDs to corresponding values
Data tags of this data point
Data blocks of this data point, with block id stored
Data tags of this data point
Data attachments of this data point

Figure 4.4. Data points storage format

4.4.2. Query and Analysis Layer

The query and analysis layer is used for different data management and data analysis purposes. The data management service component handles all of the generic data access to different data sources in the data storage layer and triggers event actions under pre-defined conditions. The data query component supports query for data points by specifying their query properties such as data stream name/id and a time range, as well as optionally providing tags, block id and unit id. Table 4.1 shows the list of currently supported query properties and descriptions. For each request the name or the id of the data stream needs to be specified in order to locate the data.

Query Properties	Descriptions
data stream name	The name of the data stream containing requested data points.
data stream id	The id of the data stream containing requested data points.
start time	The time in milliseconds. Filter data points later than the defined start time.
end time	The time in milliseconds. Filter data points earlier than the defined end time.

date	The date in *yyyy-MM-dd* format. Filter data points by the requested date.
block id	Filter data points by the requested block id.
unit id	Filter data points by the requested unit id.
max	Set the maximum total number of data points to be retrieved.
time tag	Filter the records with requested time tag only.
value tag	Filter the records with requested value tag only.

Table 4.1: Query properties and descriptions of Wiki-Health query requests.

Wiki-Health Analysis Framework

The Wiki-Health Analysis Framework (WHAF) consists of the model execution engine, model service interface and model repository. The model service interface contains the model publisher, data and models marketplace, task scheduler and result viewer. One of the goals of our system is to create an ecosystem where users can contribute their data and models. The proposed WHAF serves this purpose.

WHAF supports the capacity for scientists, developers and professionals to publish their data analysis models as services. Users can pay for the published models and obtain analysis results and services by applying the models to their collected data; users can also sell their collected data to other parties.

The data analysis service consists of the model execution engine, model service interface and model repository. Model service interface contains model publisher, data and models marketplace, task scheduler, and result viewer. Figure 4.5 illustrates the overview of the Wiki-Health Analysis Framework. The components are defined at length in the following paragraph.

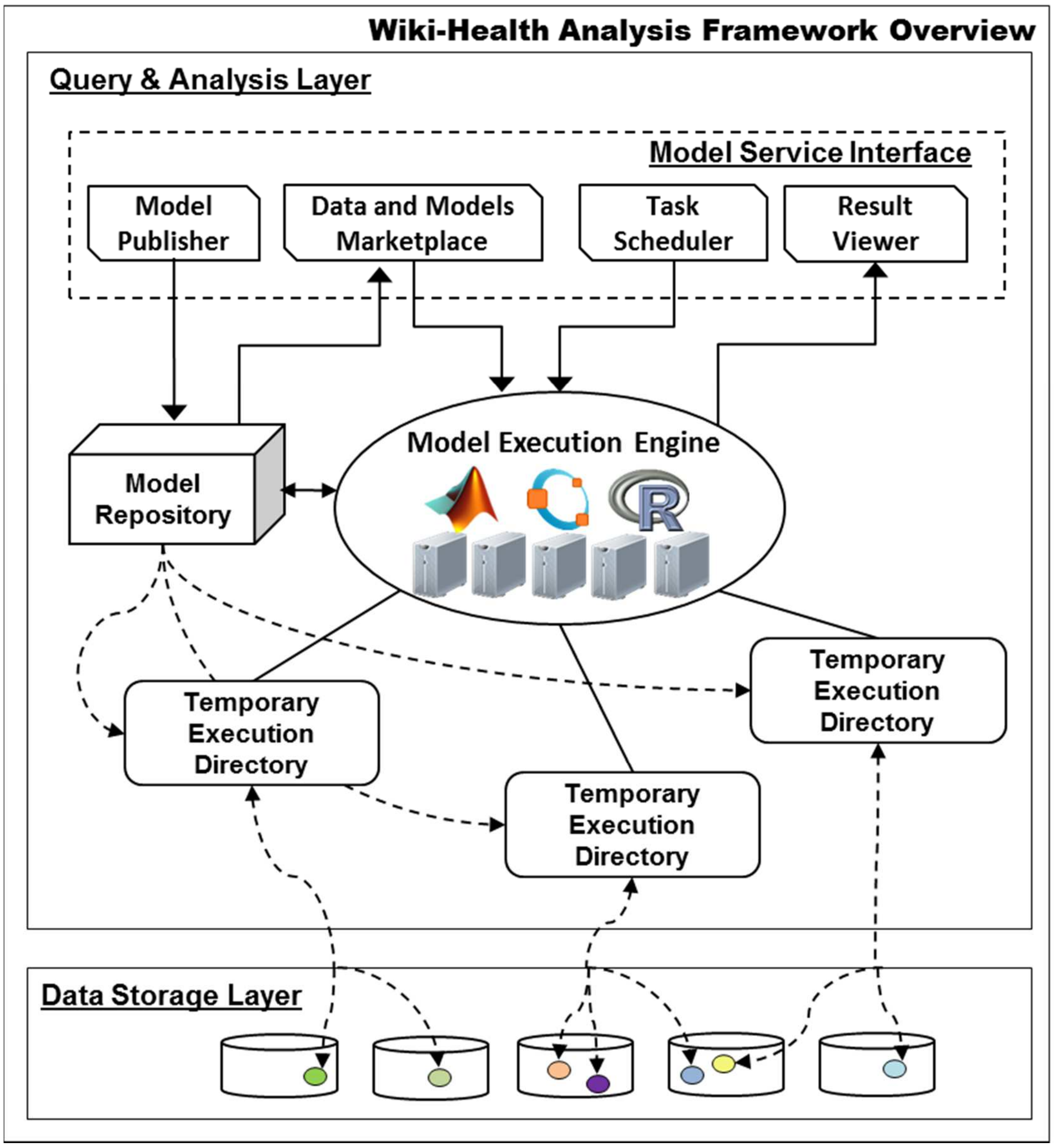

Wiki-Health Analysis Framework Overview
Query & Analysis Layer
Model Service Interface
Model Publisher
Data and Models Marketplace
Task Scheduler
Result Viewer
Model Repository
Model Execution Engine
Temporary Execution Directory
Temporary Execution Directory
Temporary Execution Directory
Data Storage Layer

Figure 4.5: The design of The Wiki-Health Analysis Framework

Model Service Interface

- Model Publisher – provides both a web page and an API-based access for model developers and experts to publish models as services. To publish a model, developers need to describe what the model service will do; specify the required permissions for accessing user's data; detail the terms and conditions of the model service; upload all the function files for the model to execute; define the formats of all the input and output parameters for the Main Function; and finally, set the price and how it is charged to the user. Currently, we have implemented pay-per-use and one-off pricing models for the proof of concept.

- Data and Models Marketplace (DMM) – serves two main purposes. The first is data commodity exchange. This enables end users as data sellers to share their collected data at a price they set and get paid when selling it. Data buyers who are interested in making use of the data, such as pharmaceutical companies, healthcare providers, insurance companies and research institutions can see a random sample of the data before it is purchased. The second purpose is to allow developers to offer models as services. Therefore the models marketplace includes developers and experts, as sellers who create and publish their developed analysis models and algorithms in the marketplace as well as end users, as buyers who can easily connect their collected data to ready-made models. In such a way, users can obtain professional data analysis services rapidly and without any programming or data analysis knowledge. To check the correctness and accuracy of a model being published in the DMM, a voting mechanism is adopted – users are able to provide their scores, feedback and reviews on the models that they have purchased and used.
- Task Scheduler – is a component that provides the ability to schedule the launch of model execution tasks to the Model Execution Engine. It allows developers and users to schedule tasks to run periodically on a given schedule.
- Result Viewer – provides an interface to allow users and developers to visualise and obtain analysis results and logs.

Model Execution Engine (MEE) – delivers services for executing data analysis models in cloud environments and preserving execution codes within existing problem solving environments such as MATLAB [109], Octave [110] and R [233]. For each model execution task, it first creates a Temporary Execution Directory (TED). Then, it retrieves the required model files from Model Repository and data through the Data Storage Layer, placing them all in the TED. Next, it sets TED as the working directory and start executing the Main Function that is used as the entry point for each model. The user declares the filename, inputs, outputs, and actionable information of the Main Function during the model publishing process. Within the Main Function, external functions and subroutines can be called and embedded for performing computations. Users and developers are able to configure the integration and connections between data sources to the model, such as the mappings of different data streams to the inputs and outputs. The Main Function provides the ability for developers to easily migrate their existing local data analysis functions and subroutines to the Wiki-Health Analysis Framework. All the output and intermediate files are kept in the TED. Finally, MEE conducts the corresponding actions such as persisting output files and data through Data Storage Layer to the underlying databases and trigger message alerts to the user.

Analysis Tasks Allocation Scheme (ATAS) for Model Execution Engine

In the experimental evaluation section of this chapter, we test the performance of processing multiple analysis tasks on a single virtual machine (VM) instance and the results have shown that using such a shared environment approach for real-time applications might not be feasible. Therefore, to guarantee analysis tasks do not interfere with each other, each analysis task in the Wiki-Health is performed at a VM level—each VM is only allowed to compute one analysis task at a time. This also enables the resource usage of each analysis task such as CPU, memory and computation time to be accurately recorded for billing and metering purposes within the proposed ecosystem. Cloud computing infrastructures offer the ability to dynamically scale up and down resources based on demand. However, booting-up a new VM can still take a few seconds or a few minutes depending on the physical hardware configurations and workload. Thus, having each analysis task allocated to a new VM might not be ideal for mission-critical or real-time applications. Although having a pool of running VMs for handling analysis tasks eliminates the waiting time of launching new ones, keeping too many VMs in the idle status (without any tasks) for long periods of time is a waste of resources and can generate unnecessary costs. Therefore, there is a clear need for an efficient way to manage resources and allocate analysis tasks.

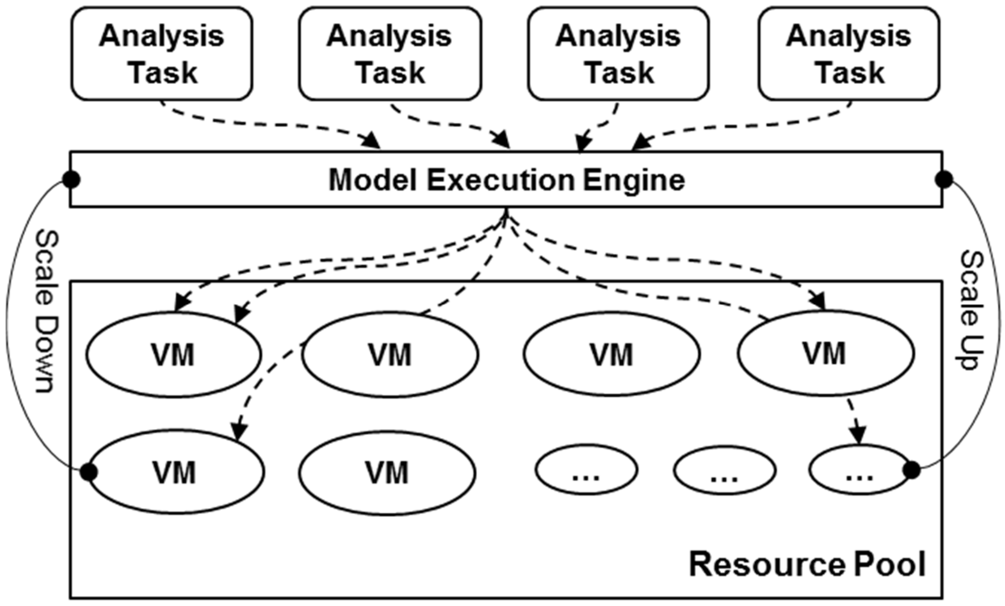


Figure 4.6: Analysis Tasks Allocation Scheme for Model Execution Engine

As shown in Figure 4.6, Wiki-Health's Analysis Tasks Allocation Scheme (ATAS) is designed to serve this purpose to target application level performance metric—the response time of launching a submitted task. The scheme is currently based on the reactive (immediate) mechanism. In this research, we assume that all of the VMs have the same configurations (CPU and memory) and that the resource pool is elastic and large enough to hold all of them. We also assume that the network bandwidth is large enough to support the data movements involved in the analysis tasks. ATAS manages the allocation of each incoming analysis task to either an idle VM, or add it to the task queue of the existing VM or launch a new VM for processing the task. It also monitors and updates the status of VMs and analysis tasks. Details of the proposed scheme are as follows:

Parameters	Semantics
S	Available resource pool for hosting virtual machine instances.
s	A virtual machine instance s from the resource pool S where $s \in S$

$t^{ef}(s)$	Expected latest time when all of allocated tasks on the VM s finish
$t(j)$	Pre-defined maximum allowed computational time (deadline) for analysis task j
$U(s)$	The status of s such as *idle* (no task running or queued), *busy* (has running tasks) and *destroyed* (no longer able to run any tasks)
t^0	Time needed for booting-up a VM (the maximum response time set by the platform provider in SLA)
t^d	Maximum idle time before destroying a VM.
$t^i(s)$	Total idle time of s
$t^b(s)$	Time elapsed for the current running task on s
J	Analysis task list
$Q(s)$	All the analysis tasks allocated and queued on s

Table 4.2: List of parameters used in the ATAS.

For each analysis task, a deadline $t(j)$ is set by the user or the model publisher before launching the task in order to better control resource consumption and prevent defective or faulty analysis models hanging in the VM. t^0 is the parameter set by the platform provider as a guarantee of maximum response time for launching an analysis task. It can be obtained by the provider by profiling the infrastructure. **Procedure 1** is used to handle new coming analysis tasks. For each new task, it first checks if there is any available idle VM. If there is, it launches the task on the VM. If not, it retrieves the VM s which has the shortest time $t^{ef}(s)$ and compares it with t^0. If the time it takes for booting-up a new VM to process the task is shorter than queuing for a busy VM, it launches the task on a new VM. Otherwise, the task will be queued for execution on that VM and the input data required by the task is transferred and preloaded to that VM.

Procedure 1: Analysis Task to VM Allocation

Input:

```
    $j$      New analysis task with pre-defined $t(j)$
-------------------------------------------------
1. Begin
```

```
2.  get a VM $s$ which $U(s)$ = *idle*
3.  **if** ($s \neq null$)
4.      launch $j$ on $s$ and add $j$ to $J$
5.      set $U(s)$=*busy*, and update $t^{ef}(s)$ with $t(j)$
6.  **return**
7.  **end if**
8.  get $s$ with the smallest $t^{ef}(s)$ and $U(s)$ ≠*destroyed*
9.  **if** ($s \neq null$ and $t^{ef}(s) \leq t^{0}$)
10.     add $j$ to $Q(s)$ and add $j$ to $J$
11.     update $t^{ef}(s)$ with $t(j)$
12. **return**
13. **end if**
14. boot new $s$ from $S$ //for all other conditions
15. add $j$ to $Q(s)$ and $j$ to $J$
16. set $U(s)$=*busy*, and update $t^{ef}(s)$ with $t(j)$
17. **End**
```

Procedure 2: Task Monitor for each VM instance

```
1.  **Begin**
2.  load and update $U(s)$
3.  **if** ($U(s)$==*idle*)
4.      **if** ($Q(s) \neq null$)
5.          perform next task from $Q(s)$
6.      **else**
7.          **if** ($t^{i}(s) > t^{d}$ and the number of idle VM > $m$)
8.              destroy $s$ from $S$, and set $U(s)$=*destroyed*
```

```
9.          end if
10.     end if
11.     return
12. end if
13. get current running task $j$ on $s$
14. if ($j \neq null$ and $t^b(s) \geq t(j)$)
15.     terminate $j$ on $s$, update $j$ on $J$
16.     perform next task from $Q(s)$
17. end if
18. End
```

Procedure 2 is used to monitor and update the status of VMs and tasks. It terminates the tasks that exceed the specified deadline, launches pending tasks from task queues, and destroys unnecessary idle VMs from the resource pool.

Model Repository – provides an environment to store analysis models and associated files. Model files are uploaded through Model Publisher into the Model Repository, storing all user and system defined functions, models and scripts for reuse of data and knowledge.

Service and Business Model

Extensive research has been performed to investigate business and pricing models for different levels of cloud services. Figure 4.7 shows the service delivery model of Wiki-Health. Three parties are involved in the service model: service vendors, service users and platform provider. Service vendors develop analysis models and publish them as services to be deployed by the platform provider. The platform provider hosts the infrastructure for storing all of the data, offering computational resources for executing data analysis services, metering all the resource and service usage, managing data sharing and permission controls, and providing the marketplace for trading services and data. Service vendors generate revenue by selling the analysis services to the service users. They are billed by the platform provider for the computational resource usage of published data analysis services, such as the virtual machine running time and network usage. Service users are able to sell their collected data to the service vendors who are interested in making use of the data.

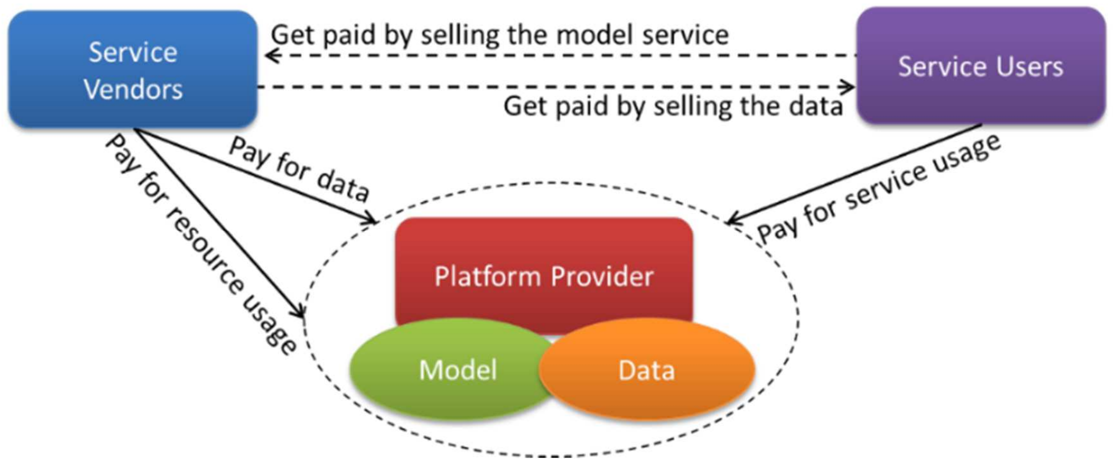


Figure 4.7: Service and Business Model.

4.4.3. Application Layer

The application layer consists of all the components required for managing data access, data collection, data security, data sharing and other features that support online collaboration.

The API service component – offers a web-based interface for access to all of the functionalities of Wiki-Health. Health sensor devices such as EEG and ECG record data from multiple sensors/channels simultaneously and have very high temporal resolution, typically at sampling rates between 250 and 2000 Hz in clinical and research settings. Wiki-health API service is designed to handle such multi-channel and multi-resolution nature of that data. Figure 4.8 illustrates an example of the structure of the sensor data query response by the API. Data tags are marked with red circles in the figure. They can be assigned and accessed on timestamp and data unit levels.

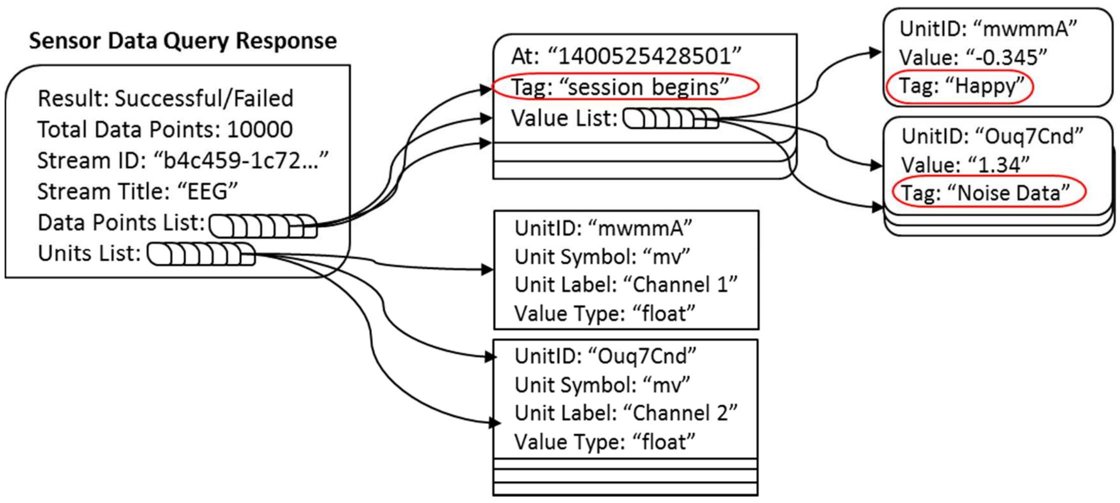
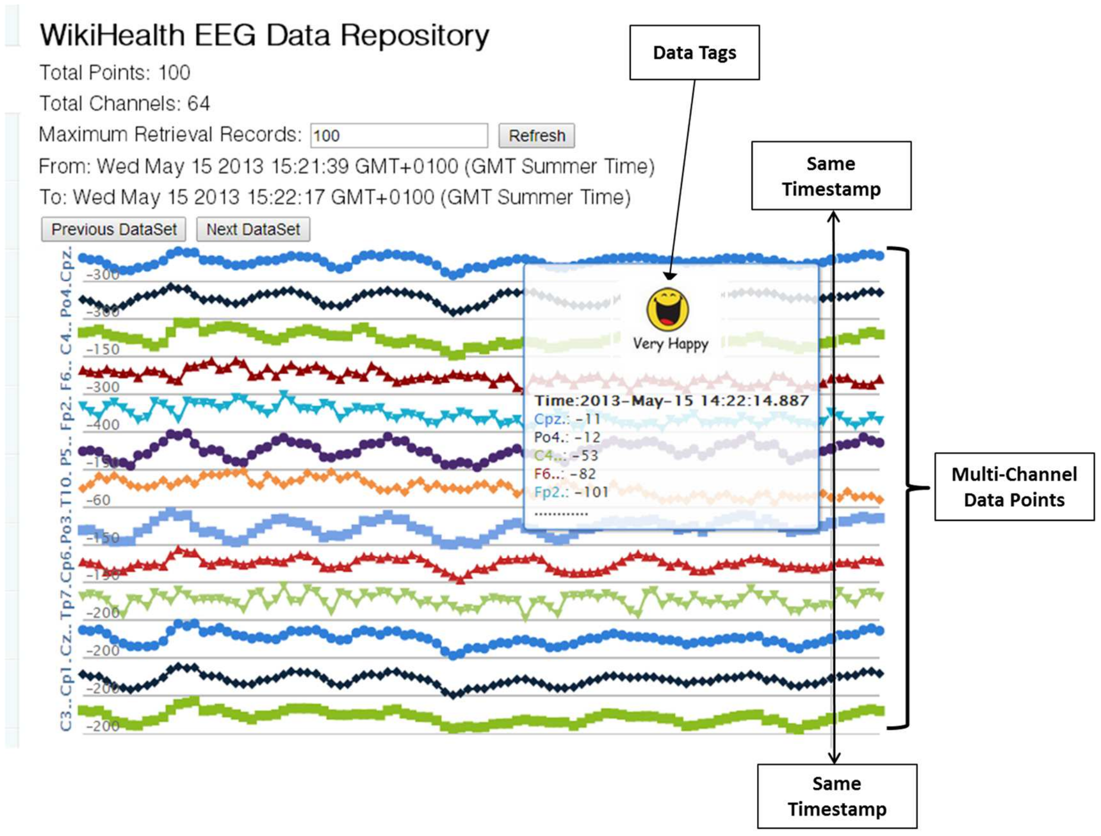


Figure 4.8: The structure of the response for a sensor data query request.

Application developers can adopt the Wiki-Health API to build their own service applications by handling the heterogeneous data and tags with their preferred methods. To demonstrate such idea, Figure 4.9 shows an example of an independent EEG Data visualisation tool that retrieves the data from Wiki-Health using the aforementioned structure of the API and displays the multi-channel sensor data and data tags using charts for users.


WikiHealth EEG Data Repository
Total Points: 100
Total Channels: 64
Maximum Retrieval Records: 100
From: Wed May 15 2013 15:21:39 GMT+0100 (GMT Summer Time)
To: Wed May 15 2013 15:22:17 GMT+0100 (GMT Summer Time)

Time:2013-May-15 14:22:14.887
Cpz.: -11
Po4.: -12
C4..: -53
F6..: -82
Fp2.: -101
....

Figure 4.9: Example of an EEG Data Viewer implemented by using Wiki-health API for storing and displaying multi-channel EEG data

The identity access management component – is a separate service that provides authorisation and access control of all requests. The messaging interface is used to connect Wiki-Health to external third party API and services. The data visualisation component is used to generate different graphical representations of raw sensor data as well as processed data.

The collaboration management service – comprises a data sharing policy engine to resolve and manage data sharing policies defined for data streams. These policies ensure that data can only be accessed by specific users, at prerequisite times and from the right locations. The social network integration component creates a data-sharing social environment for users to share their data and communicate with other people who have common needs and who want to learn more about themselves.

The data collection engine – is intended to liberate a user's personal health data collected through different providers, devices and platforms, by allowing a user to create a copy of all of his or her data and maintain it in the Wiki-Health platform through the third party API.

Wiki-Health Mobile App

The purpose of labelling, tagging or annotating sensor data is often to identify the correct event, stimulus or cause associated with a corresponding sequence of sensor data. Such labels are useful for data analysis algorithms. Therefore, we have developed a mobile application as part of the platform to help users collect and tag sensor data from smartphones. The application can also be used to trigger alerts and interact with feedback from the Wiki-Health through the platform API. As shown in Figure 4.10, the design of the mobile app can be divided into the following mechanisms:

- Background monitoring – provides all the necessary functions responsible for the periodic acquisition of measurements from the sensors of the device and their storage in the local database. Users can select the sensors for monitoring in the background and specify how frequently data is collected and stored.
- Activity labelling – refers to the mechanism of giving users the ability to enter information about their activities. It allows them to tag and enter additional information about their activities during specific hours of a selected date. Such information is annotated and linked to the collected sensor data and stored in the local database.
- Data uploading – is the mechanism containing all of the operations related to the communication between the application and the remote storage space. Users can choose whether to automatically upload the collected data to Wiki-Health periodically or else store the data locally in a private database from the application before uploading to Wiki-Health through the API. The latter option is designed to improve privacy control—users can remove any unwanted collected data, so that only user-selected data will be uploaded. In addition, since upload might not to be performed regularly by the user – and also if the sampling rate of the sensor is high – it is possible to store a large amount of data in the local database. To send the data efficiently, the collected data is divided into small chunks of data and sent successively. Data stored in the local database is deleted only if the upload has been successful, so as to prevent data loss during unexpected events.

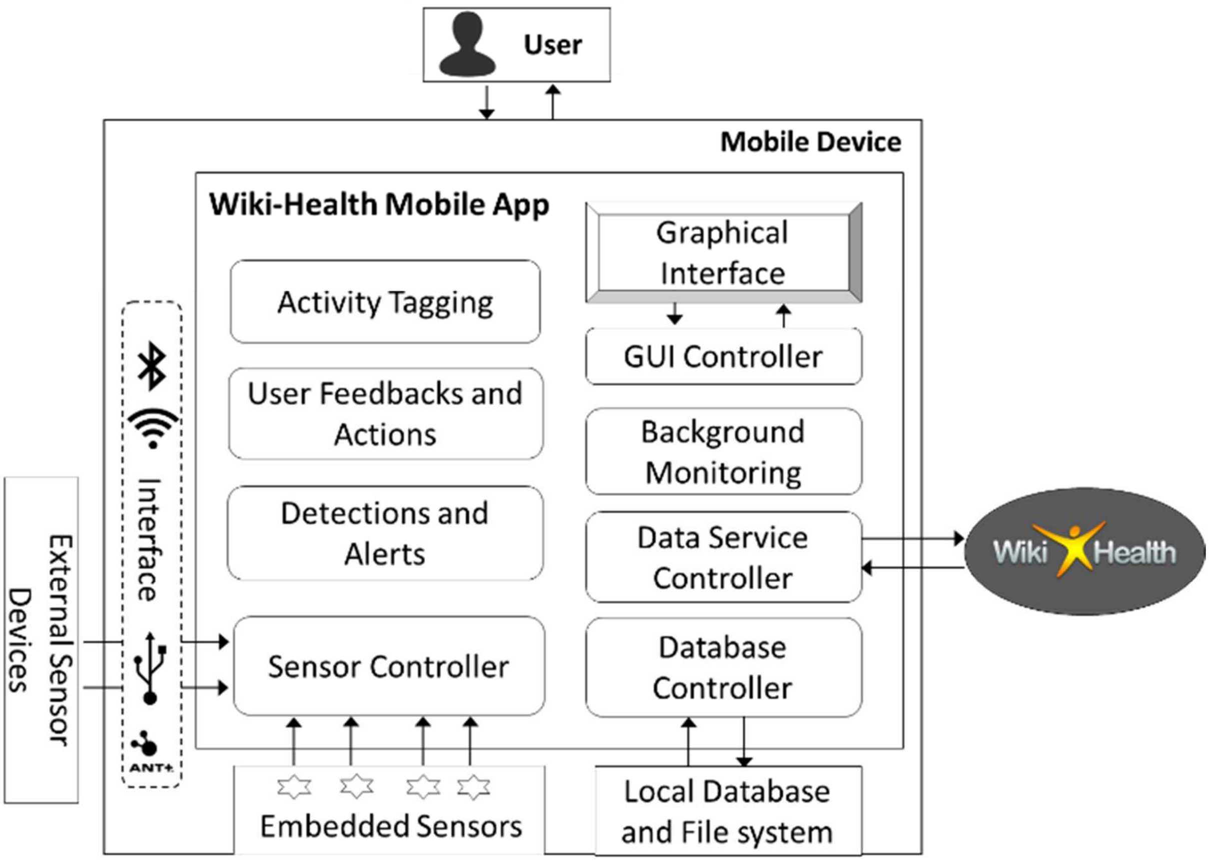


Figure 4.10. The architecture of Wiki-Health Mobile App.

4.5. Implementation

4.5.1. Wiki-Health Mobile App

Wiki-Health Mobile App is currently developed in Android and supports data collection with an array of embedded sensors including accelerometers, gravity sensors, gyroscopes, GPS as well as inputs from other external sensors such as ambient air temperature, heart rate and ECG. Figure 4.11 shows some screenshots of the current mobile application.

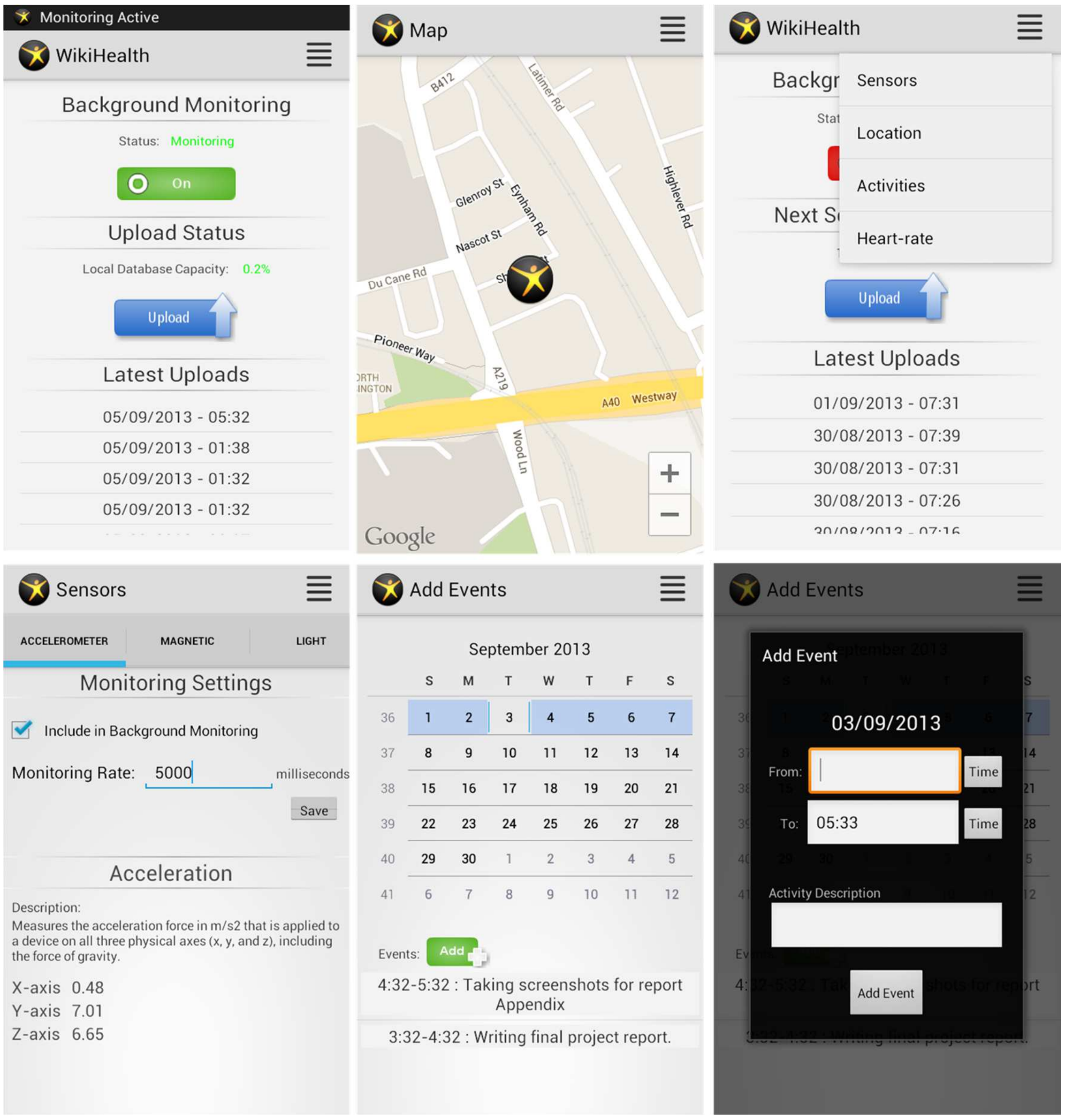


Figure 4.11: Screenshots of Wiki-Health Mobile App

4.5.2. Data Visualisation Tools

The process of designing Wiki-Health's data visualisation tools involve the initial conceptual design based on the goals, requirements and functionalities of the platform, followed by the information architecture and navigation design. Next, the interaction design with logical and physical UI prototypes was conducted before reaching the final design and implementation stage. 15 people of varying educational background, gender and age were asked to provide feedback and reviews during the design process.

Several interfaces have been developed for users to visualise their collected data. Figure 4.12 shows an interface with a pre-defined page layout for users to view their stored sensor data simply by selecting the interval (day, week, month or year) and the start date. All of the data displayed in the figure were collected from a 55 year-old male adult using various sensor devices. Wiki-Health users are also able to share their data with families, friends and caregivers.

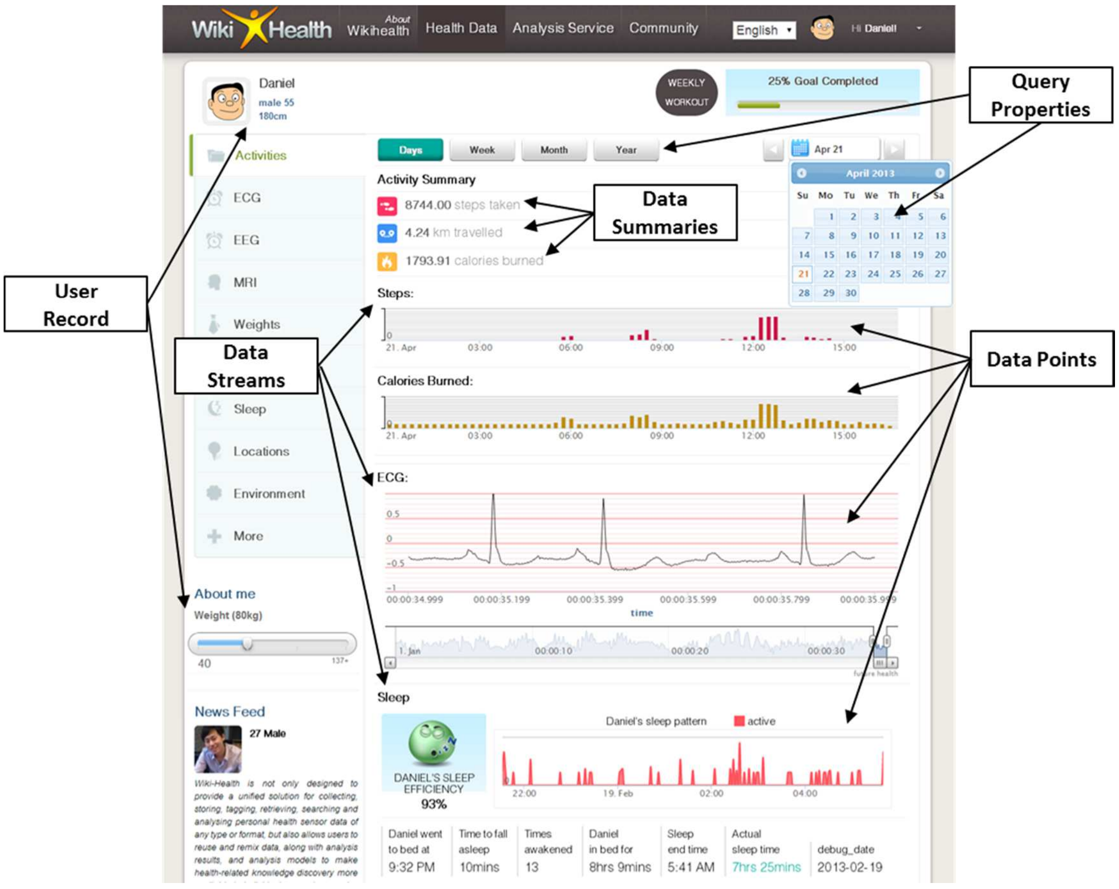


Figure 4.12: Wiki-Health's Data Visualisation Interface.

Figure 4.13 shows the screenshot of Wiki-Health's Data Management Portal. The portal provides graphical interface for users with a means to interact with the Wiki-Health platform through the API. It enables various features: users can add and edit customised data streams and data points, view data points from a particular data stream through texts or graphical charts.

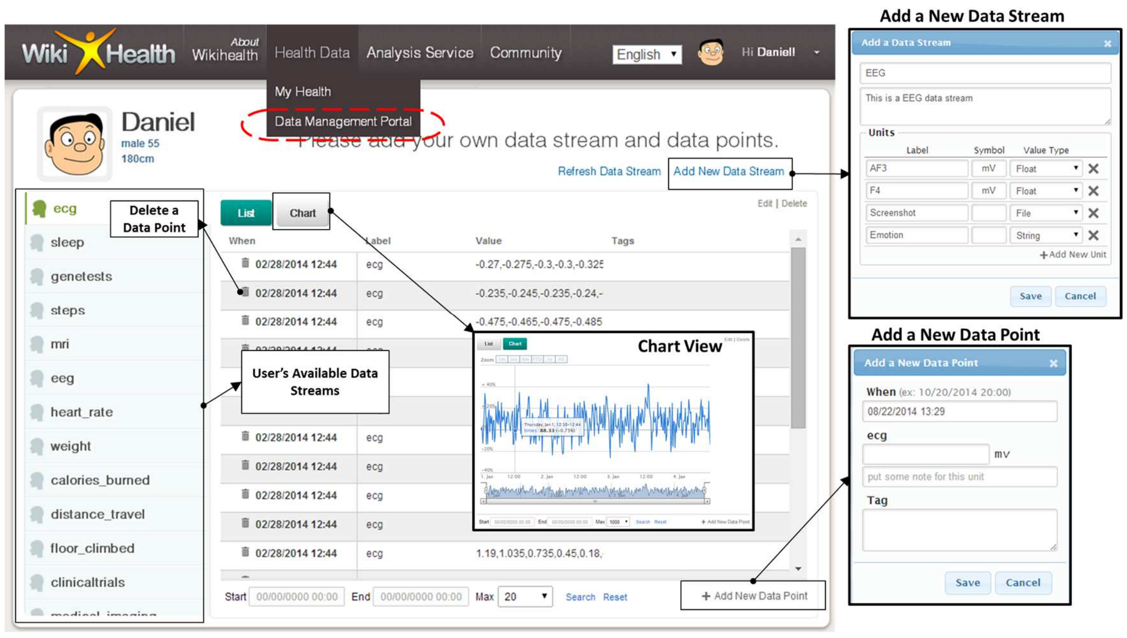
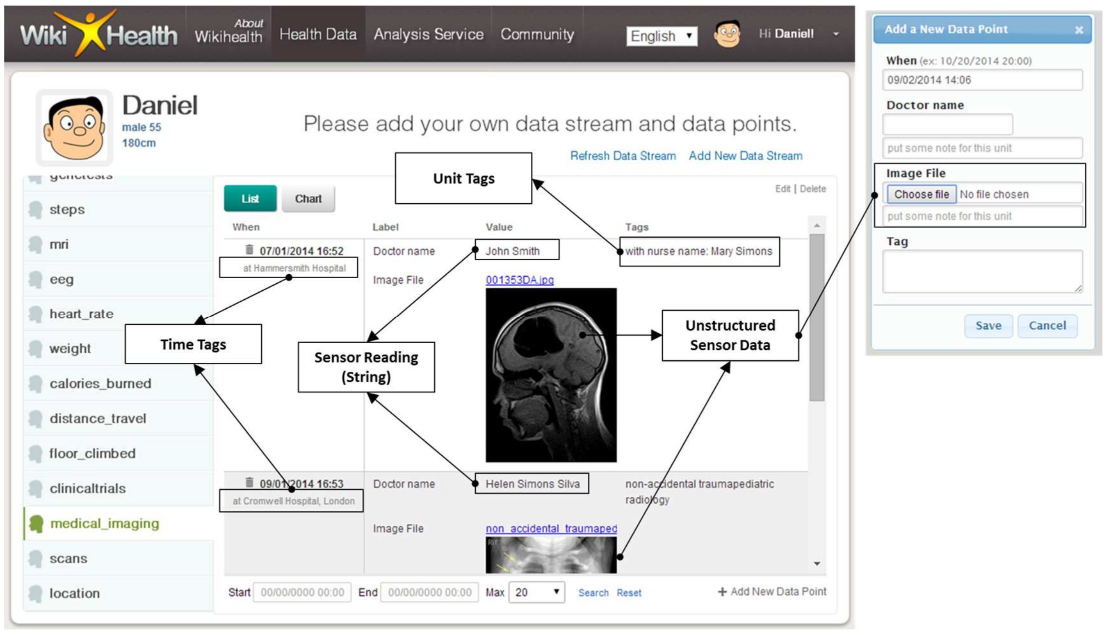


Figure 4.13: Wiki-Health's Data Management Portal for managing data streams and data points.

The Data Management Portal also allows users to upload unstructured sensor data such as medical imaging files to the Wiki-Health platform through the API. Figure 4.14 shows a screenshot of using Wiki-Health for managing personal medical imaging data. In the example, the data stream contains two data units: the name of the doctor and the image file. Users are also able to add additional information as tags to the data stream, such as the name of the hospital and diagnostic results.



Please add your own data stream and data points.

When	Label	Value	Tags
07/01/2014 16:52 at Hammersmith Hospital	Doctor name	John Smith	with nurse name: Mary Simons
	Image File	001353DA.jpg	
09/01/2014 16:53 at Cromwell Hospital, London	Doctor name	Helen Simons Silva	non-accidental traumapediatric radiology
	Image File	non_accidental_traumaped	

Start 00/00/0000 00:00 End 00/00/0000 00:00 Max 20

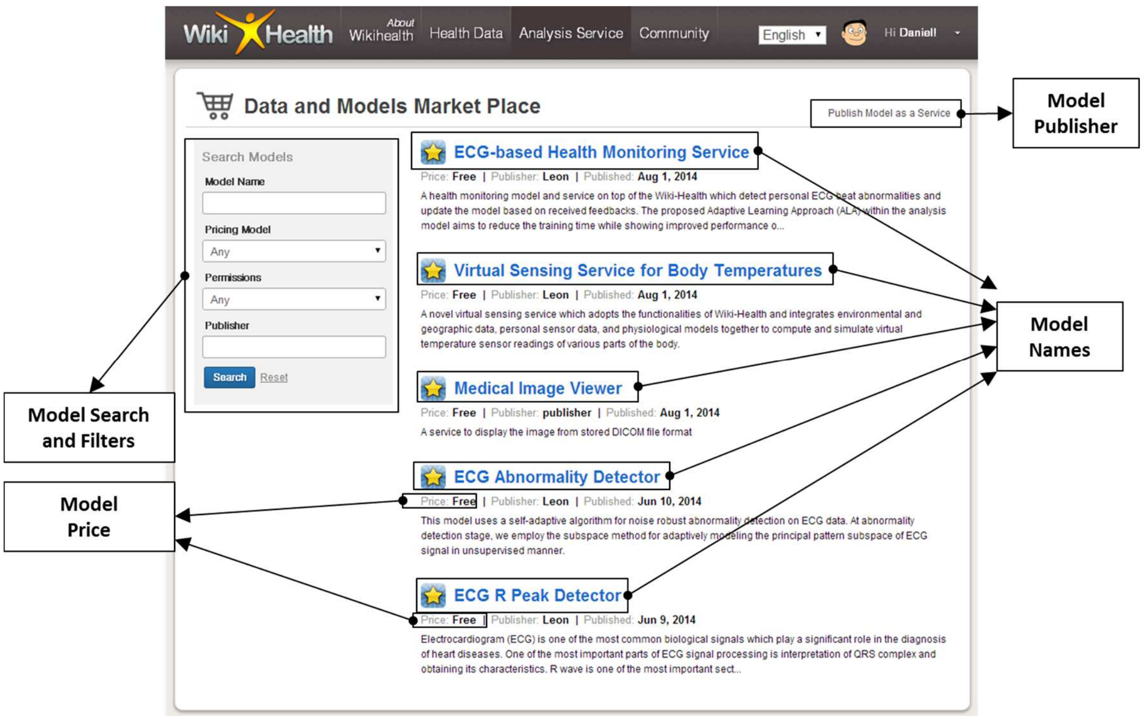
When (ex: 10/20/2014 20:00) 09/02/2014 14:06
Doctor name
put some note for this unit
Image File
No file chosen
put some note for this unit
Tag

Figure 4.14: Example of using Wiki-Health for managing unstructured sensor data and data tags.

4.5.3. Model Execution Engine

The Model Execution Engine (MEE) is currently implemented using Java and Octave [110]. Octave is the high level programming language most compatible with MATLAB, one of the most commonly used programming languages in domains such as bioinformatics, physics, and chemistry. MEE consists of a master node application for allocating incoming analysis tasks to task nodes (VMs) and managing the acquisition and release of VMs. The application for task nodes is responsible for processing allocated analysis tasks from the task queue and handling the lifecycle of the tasks. Both master and task node applications are deployed on Tomcat [234] on each VM. Task Scheduler is currently implemented using Java Quartz Scheduler [235]. All of the tasks are defined using time-based CRON expressions.

Figure 4.15 illustrates the interface of the Data and Models Marketplace. Model names, model descriptions, pricing and publisher information are listed on the interface. Users are able to search and view different models before making purchase. After a model is purchased, it will appear in the user's account (as shown in Figure 4.16). Users can configure the model with their desired inputs and outputs before launching it.



Data and Models Market Place

ECG-based Health Monitoring Service
Price: **Free** | Publisher: **Leon** | Published: **Aug 1, 2014**
A health monitoring model and service on top of the Wiki-Health which detect personal ECG beat abnormalities and update the model based on received feedbacks. The proposed Adaptive Learning Approach (ALA) within the analysis model aims to reduce the training time while showing improved performance o...

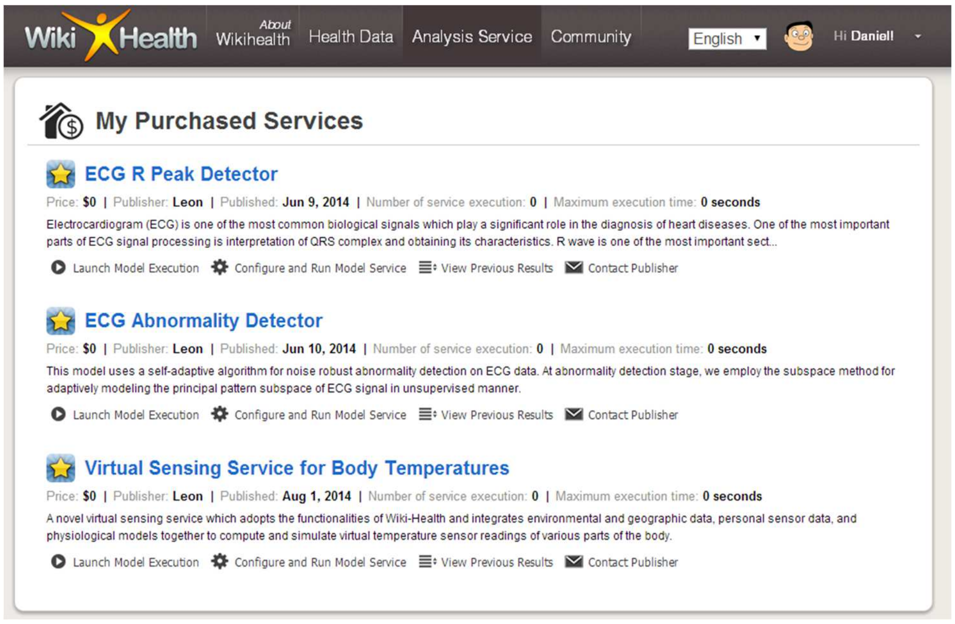
Virtual Sensing Service for Body Temperatures
Price: **Free** | Publisher: **Leon** | Published: **Aug 1, 2014**
A novel virtual sensing service which adopts the functionalities of Wiki-Health and integrates environmental and geographic data, personal sensor data, and physiological models together to compute and simulate virtual temperature sensor readings of various parts of the body.

Medical Image Viewer
Price: **Free** | Publisher: **publisher** | Published: **Aug 1, 2014**
A service to display the image from stored DICOM file format

ECG Abnormality Detector
Price: **Free** | Publisher: **Leon** | Published: **Jun 10, 2014**
This model uses a self-adaptive algorithm for noise robust abnormality detection on ECG data. At abnormality detection stage, we employ the subspace method for adaptively modeling the principal pattern subspace of ECG signal in unsupervised manner.

ECG R Peak Detector
Price: **Free** | Publisher: **Leon** | Published: **Jun 9, 2014**
Electrocardiogram (ECG) is one of the most common biological signals which play a significant role in the diagnosis of heart diseases. One of the most important parts of ECG signal processing is interpretation of QRS complex and obtaining its characteristics. R wave is one of the most important sect...

Figure 4.15: Interface of Data and Models Marketplace



My Purchased Services

ECG R Peak Detector
Price: **$0** | Publisher: **Leon** | Published: **Jun 9, 2014** | Number of service execution: **0** | Maximum execution time: **0 seconds**
Electrocardiogram (ECG) is one of the most common biological signals which play a significant role in the diagnosis of heart diseases. One of the most important parts of ECG signal processing is interpretation of QRS complex and obtaining its characteristics. R wave is one of the most important sect...
Launch Model Execution | Configure and Run Model Service | View Previous Results | Contact Publisher

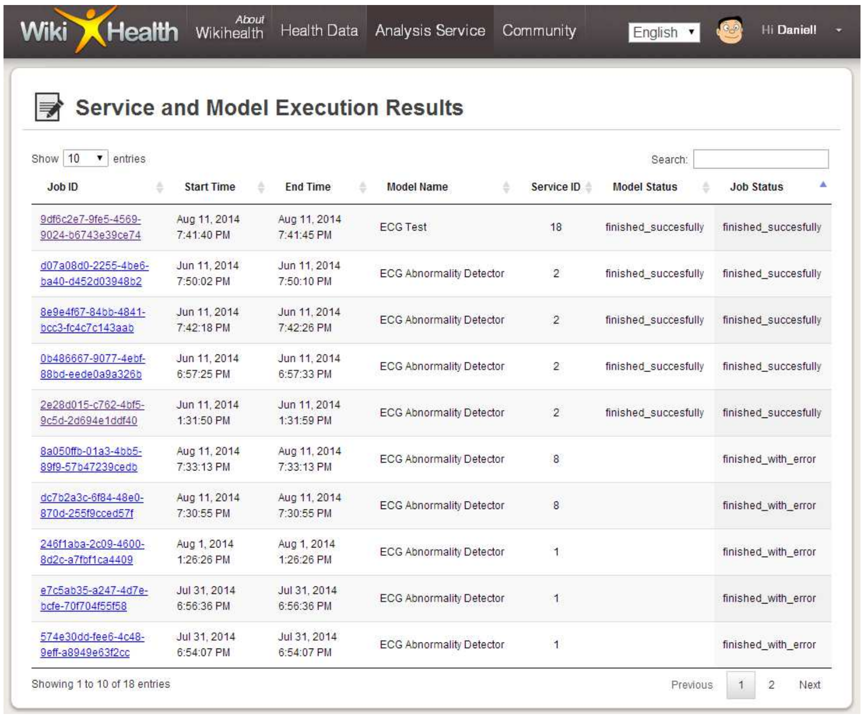
ECG Abnormality Detector
Price: **$0** | Publisher: **Leon** | Published: **Jun 10, 2014** | Number of service execution: **0** | Maximum execution time: **0 seconds**
This model uses a self-adaptive algorithm for noise robust abnormality detection on ECG data. At abnormality detection stage, we employ the subspace method for adaptively modeling the principal pattern subspace of ECG signal in unsupervised manner.
Launch Model Execution | Configure and Run Model Service | View Previous Results | Contact Publisher

Virtual Sensing Service for Body Temperatures
Price: **$0** | Publisher: **Leon** | Published: **Aug 1, 2014** | Number of service execution: **0** | Maximum execution time: **0 seconds**
A novel virtual sensing service which adopts the functionalities of Wiki-Health and integrates environmental and geographic data, personal sensor data, and physiological models together to compute and simulate virtual temperature sensor readings of various parts of the body.
Launch Model Execution | Configure and Run Model Service | View Previous Results | Contact Publisher

Figure 4.16: Example of the interface showing a user's purchased services.

In the model configuration interface, the user is able to configure data sources for inputs and outputs. For each entry, based on the data type defined by the model publisher, the user can select a target health sensor data stream from the list or choose an un-structured object from the CACSS storage system by filling the object key or entering any text data that is needed.

Each purchased model can be launched many times. Each launch will create an independent analysis task. As shown in Figure 4.17, the user can track and view all of the information for launched analysis tasks such as the start time, end time, model name, and completion status. If there is any error, details of the error and debugging information will also be shown on the results page.



Service and Model Execution Results

Show 10 entries

Search:

Job ID	Start Time	End Time	Model Name	Service ID	Model Status	Job Status
9df6c2e7-9fe5-4569-9024-b6743e39ce74	Aug 11, 2014 7:41:40 PM	Aug 11, 2014 7:41:45 PM	ECG Test	18	finished_succesfully	finished_succesfully
d07a08d0-2255-4be6-ba40-d452d03948b2	Jun 11, 2014 7:50:02 PM	Jun 11, 2014 7:50:10 PM	ECG Abnormality Detector	2	finished_succesfully	finished_succesfully
8e9e4f67-84bb-4841-bcc3-fc4c7c143aab	Jun 11, 2014 7:42:18 PM	Jun 11, 2014 7:42:26 PM	ECG Abnormality Detector	2	finished_succesfully	finished_succesfully
0b486667-9077-4ebf-88bd-eede0a9a326b	Jun 11, 2014 6:57:25 PM	Jun 11, 2014 6:57:33 PM	ECG Abnormality Detector	2	finished_succesfully	finished_succesfully
2e28d015-c762-4bf5-9c5d-2d694e1ddf40	Jun 11, 2014 1:31:50 PM	Jun 11, 2014 1:31:59 PM	ECG Abnormality Detector	2	finished_succesfully	finished_succesfully
8a050ffb-01a3-4bb5-89f9-57b47239cedb	Aug 11, 2014 7:33:13 PM	Aug 11, 2014 7:33:13 PM	ECG Abnormality Detector	8		finished_with_error
dc7b2a3c-6f84-48e0-870d-255f9cced57f	Aug 11, 2014 7:30:55 PM	Aug 11, 2014 7:30:55 PM	ECG Abnormality Detector	8		finished_with_error
246f1aba-2c09-4600-8d2c-a7fbf1ca4409	Aug 1, 2014 1:26:26 PM	Aug 1, 2014 1:26:26 PM	ECG Abnormality Detector	1		finished_with_error
e7c5ab35-a247-4d7e-bcfe-70f704f55f58	Jul 31, 2014 6:56:36 PM	Jul 31, 2014 6:56:36 PM	ECG Abnormality Detector	1		finished_with_error
574e30dd-fee6-4c48-9eff-a8949e63f2cc	Jul 31, 2014 6:54:07 PM	Jul 31, 2014 6:54:07 PM	ECG Abnormality Detector	1		finished_with_error

Showing 1 to 10 of 18 entries

Previous 1 2 Next

Figure 4.17: Example of the Wiki-Health interface showing all of the launched analysis tasks of a user.

All Wiki-Health components outlined by this thesis are deployed across different virtual machines (VMs) on top of IC-Cloud [173, 174]. IC-Cloud is a generic IaaS cloud-computing infrastructure which allows for the rapid design and deployment of a cloud environment in a flexible manner.

4.6. Experimental Evaluation

In the previous chapter, the performance of CACSS Cloud Storage System for unstructured and semi-structured data management has been evaluated from different aspects.

In this section, we test the performance of Wiki-Health platform for time-series based sensor data under different workloads.

4.6.1. Performance of Time-series based sensor data management

We constructed and configured the Wiki-Health platform on top of IC-Cloud. A total of six virtual machines (VMs) are deployed across two physical machines. Each physical machine has 24 physical cores (Xeon E5-2695V2@ 2.4 GHz processors). VMs are connected to a 10Gbit/s shared network. One VM with 16GB of memory and four CPU cores was used as the master node to run MySQL[195], HDFS [67] NameNode, HBase [71] Hmaster, Zookeeper, and Tomcat with the Wiki-Health platform. Five VMs were each configured with 4GB memory and two CPU cores. Four of them are used as slave nodes to run HDFS DataNodes and HBase Regionservers. The last one is used as the emulator to simulate the requests of the users with the Apache benchmarking tool [236]. ECG signals from the MIT-BIH database Physiobank [237, 238], a large archive of well-characterized digital recordings of physiologic signals and related data for use in biomedical research, were used to generate the data points for the tests. The data contains time-series based ECG signals digitised at a sample rate of 360 samples per second, which is much higher than many existing sensors on smartphones such as GPS, gravity and accelerometer, as well as other health devices such as blood pressure monitor and pulse oximeter (SpO2).

The first workload was to test Wiki-Health's performance in handling concurrent data upload requests. We simulated the uploading requests of 100, 1,000, and 10,000 data points each to an individual data stream by different numbers of concurrent users. The corresponding average request response times are shown in Figure 4.18.

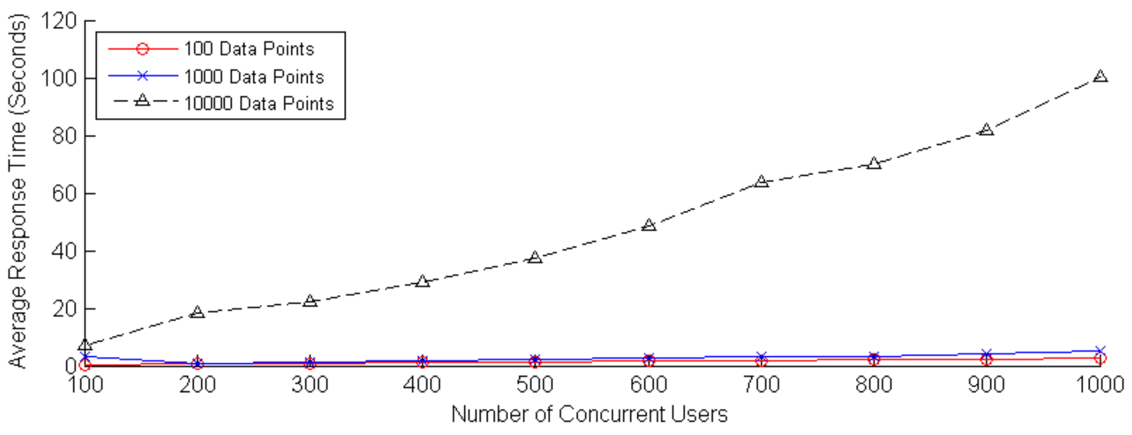


Figure 4.18. Response times for sensor data uploading requests by increasing the number of concurrent users.

In the second workload, we simulated the data querying and retrieving requests with window sizes of 100, 1,000, and 10,000 data points by different numbers of concurrent users. All requests are to retrieve the data from the same data stream. This workload was designed to test the performance in scenarios in which multiple devices or users are retrieving data from the same data stream simultaneously. The third workload was similar to the second workload, the only difference being that all of the requests were to different data streams. This was designed to test the performance of scenarios in which there are many devices or users retrieving data from different data streams. Corresponding average request response times are shown in Figure 4.19 and Figure 4.20, respectively.

In Figure 4.19, there were some fluctuations in the response times for the window of 10000 data points marked in black. These were caused by the shared network and storage environment. Results from all three workloads show that the response time is dependent on the number of data points and increases linearly with the number of concurrent users.

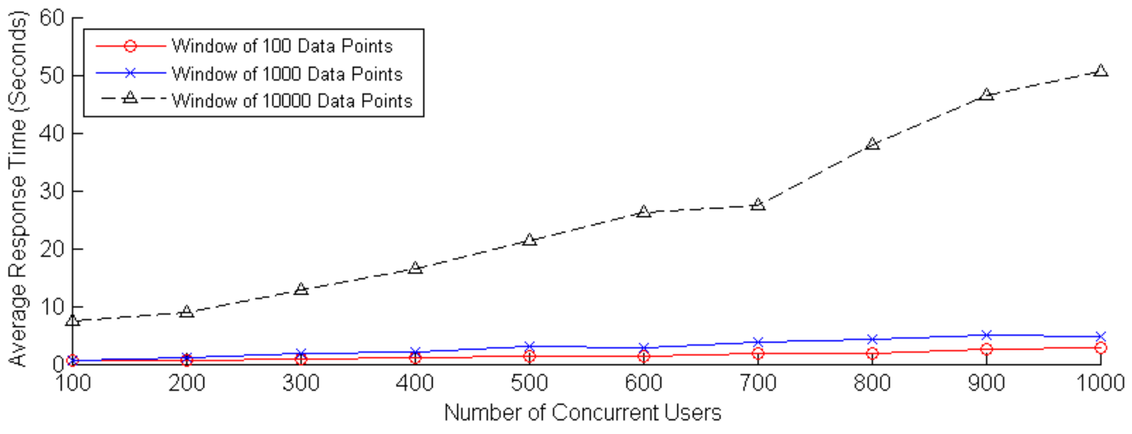


Figure 4.19. Response times for sensor data retrieving requests from a single data stream by increasing the number of concurrent users.

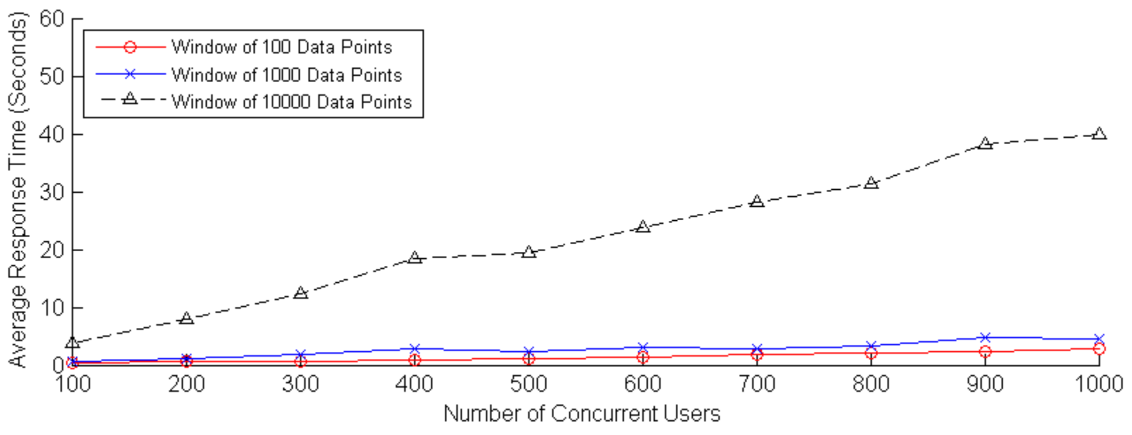


Figure 4.20. Response times for sensor data retrieving requests from multiple data streams by increasing the number of concurrent users.

4.6.2. Concurrent Model Executions on a Single VM

This experiment was conducted to evaluate the performance of processing analysis tasks on a single VM, i.e. without Analysis Tasks Allocation Scheme. The model of ECG QRS detection algorithm by Pan and Tompkins is implemented using MATLAB/Octave language and published as a service in the Wiki-Health. The experimental environment is the same as described in the previous experiment. Service users are able to create analysis tasks that take their collected ECG data (uploaded from the previous experiment) as inputs and outputs ECG signal plots with annotated R peaks. The master node was used as the only VM for performing all the analysis tasks. The emulator node was used to create live analysis tasks using 100, 1,000, and 10,000 data points from each individual data stream as inputs by different number of concurrent requests. Corresponding average request response times are depicted in Figure 4.21. The results show that response time is dependent on the number of data points and increases linearly with the number of concurrent requests. Each individual analysis task occupies some CPU time and memory and does interfere the performance of other tasks. Therefore, having a large number of tasks concurrently running in the same environment might not be feasible for applications that require near real-time response.

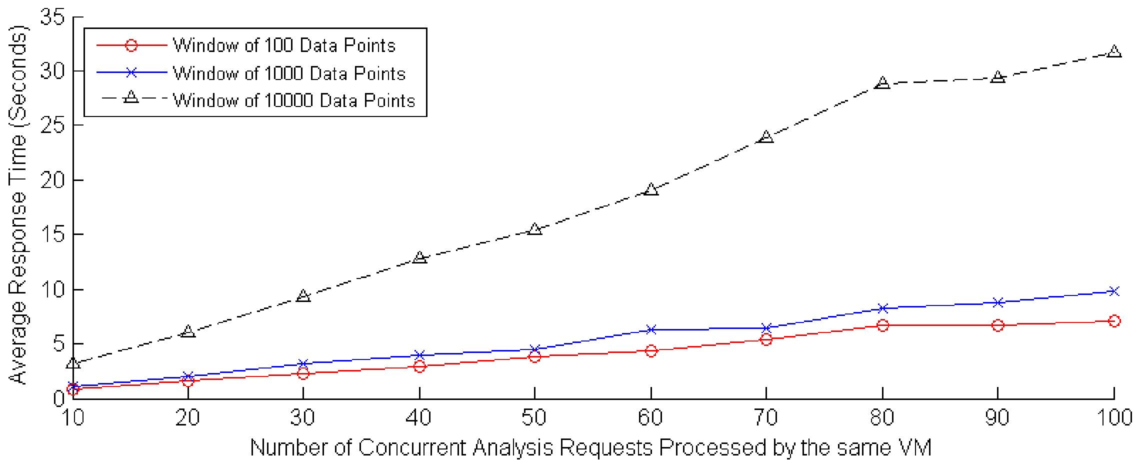


Figure 4.21: Response times for live analysis tasks allocated on the same VM by increasing the number of concurrent requests.

4.6.3. Effectiveness of Analysis Tasks Allocation Scheme for Model Execution Engine

This set of the experiments was conducted to evaluate how Analysis Tasks Allocation Scheme performs under different conditions and scenarios. Due to our limited computing resources, evaluating the scheme on a large scale was not feasible in the real environment of IC-Cloud. A custom simulator was developed to simulate incoming analysis task requests based on two types of workload patterns shown in Figure 4.22. Each stage within a workload lasts for 60 seconds. For comparison, two reactive approaches were simulated. Both of them adopt the elasticity characteristic of the cloud computing in which computational resources can be obtained and released to match the current demand. Details of the two are as follows:

- NVPT: boot-up a new VM to handle every single analysis task request and destroy the VM on task completion.
- ATAS: allocate each incoming analysis task either to an idle VM, adding to the task queue or boot-up a new VM for processing based on the ATAS scheme. The VM is not destroyed immediately after tasks finished.

In the current IC-Cloud environment (with SSD hard drives on physical machines), booting-up a VM with 2 CPU cores, 8GB memory and 20GB disk and load all the necessary programs completely takes approximately 20-30 seconds. Therefore, for both approaches the time needed to boot-up a VM (t^0) was set to 30 seconds. Two settings of deadline parameter $t(j)$ were used: 20 seconds in Setting 1 and 5 seconds in Setting 2 for any job j. The actual time of each task was chosen randomly between $0.2 \times\ t(j)$ and $t(j)$. t^d was set to 60 seconds for the ATAS. For both approaches under different workloads and settings, the total number of live VMs and total waiting time for launching analysis tasks in each second are recorded. The results are plotted in Figure 4.23 and Figure 4.24, showing that the ATAS approach can significantly reduce the total VM running times and total response times for executing the tasks than the NVPT approach. In a cloud computing environment where t^0 is longer than IC-Cloud's 30 seconds, the difference between the two approaches can be larger and smaller if t^0 is shorter.

In the current ATAS, we have assumed that the network bandwidth is large enough to hold all the data movements between the VMs and the Wiki-Health data storage layer. In reality, although ATAS can handle linear scalability of analysis tasks efficiently, the network can potentially produce a bottleneck, depending on the bandwidth.

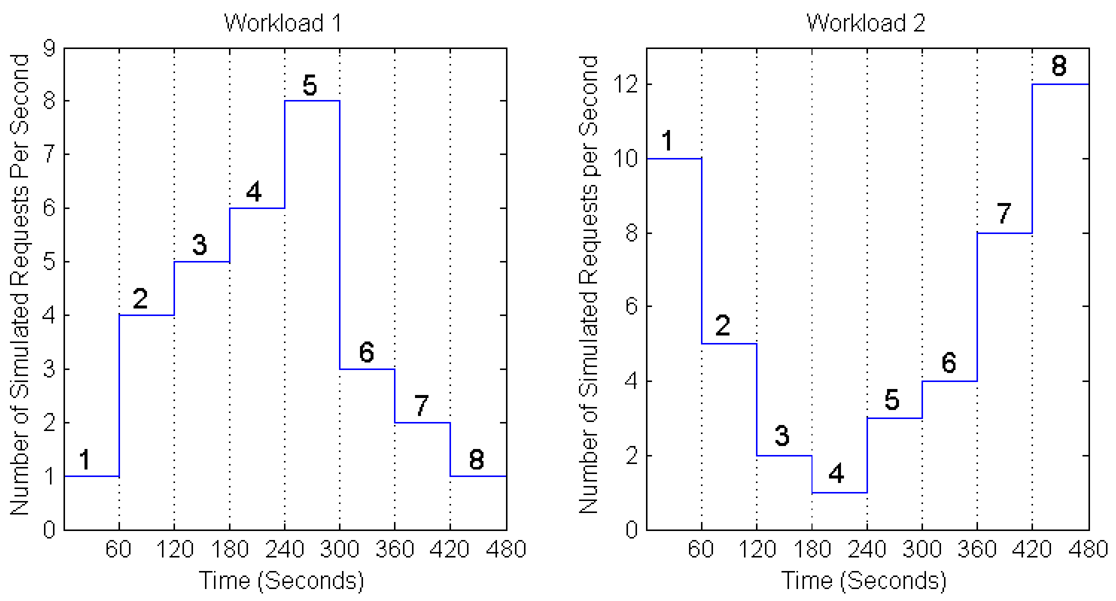


Figure 4.22: Two workloads with different number of simulated concurrent requests ranging from stage 1 to 8.

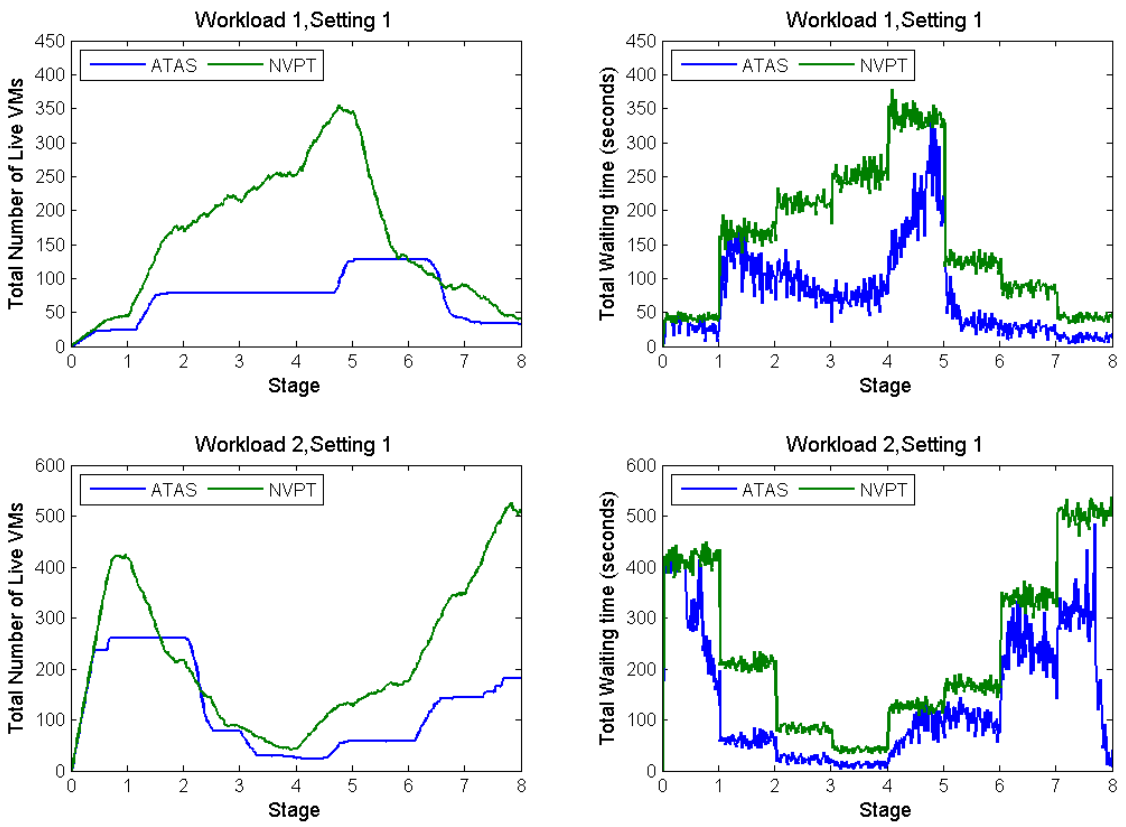


Figure 4.23: Comparison of total number of live VMs and waiting times for the analysis tasks at different stages using Setting 1 with ATAS and NVPT approaches.

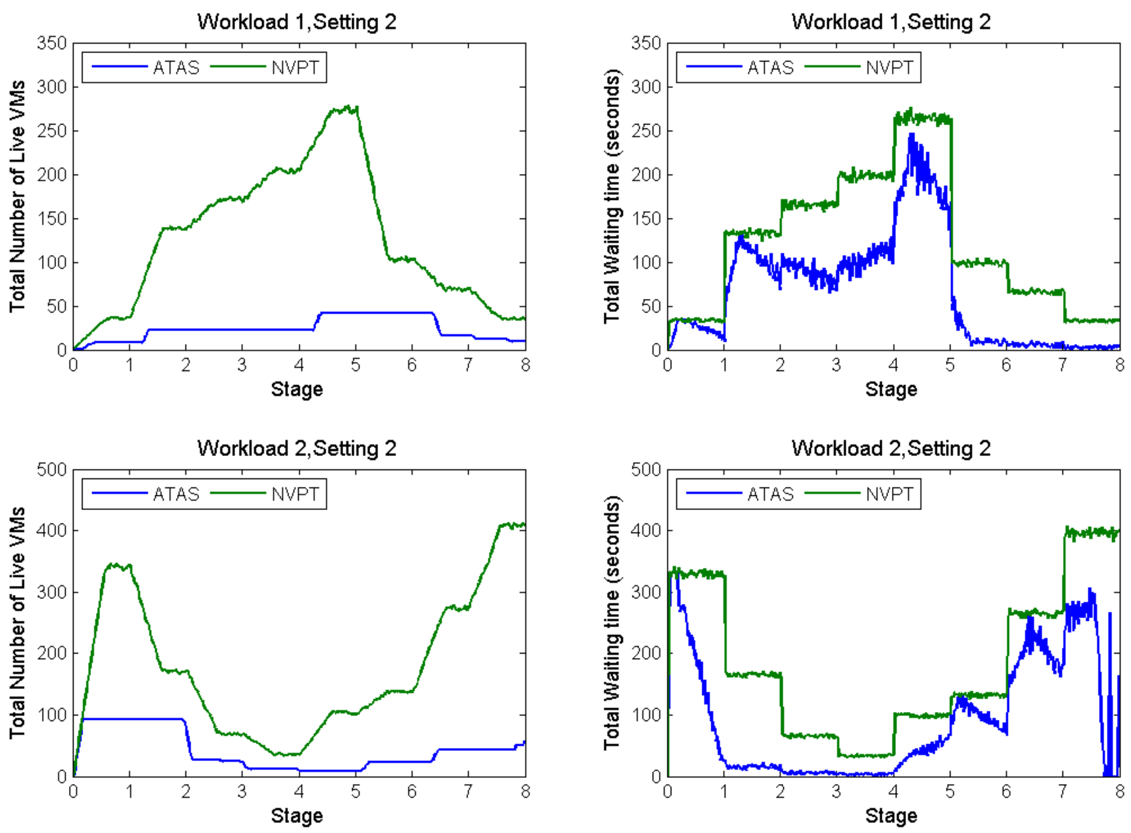


Figure 4.24: Comparison of total number of live VMs and waiting times for the analysis tasks at different stages using Setting 2 with ATAS and NVPT approaches.

4.7. Summary

This chapter presented the design rationale and implementation of Wiki-Health, a cloud-based platform for personal health sensor data management. The proposed hybrid storage model supports the storage of different types of sensor data and facilitates high performance for both data accessing and analysis operations. We have also introduced a collaborative approach for health sensor data management that allows users not only to collect, annotate, update and share sensor data, but also to create, reuse and remix data analysis results and models. The performance of Wiki-Health's sensor data management is evaluated based on the response time of different workloads. The Analysis Tasks Allocation Scheme proposed in the research aims to aid the management of data analysis tasks on a large scale and utilises the elasticity nature of the cloud infrastructure by considering aspects of both performance and cost. From the performance perspective, the scheme allocates each task to the “best” VM that can minimise the waiting time for execution. From the cost perspective, it acquires and keeps an appropriate number of VMs throughout the time, thus reducing the cost of the resources. The ATAS is currently based on a reactive approach and has the potential to combine with a predictive method for future work. In the next two chapters, we present the design and implementation details of two healthcare applications running on top of Wiki-Health to demonstrate the feasibility of using the platform to make health-related knowledge discovery available to individual users.

5. Enabling Health Monitoring as a Service with Wiki-Health Platform

5.1. Introduction

In this chapter, we present a step along the road towards a proactive healthcare and well-being management approach by adapting the Wiki-Health platform for healthcare monitoring applications. "Health-monitoring-as-a-service" is introduced as a key application for future personal health management. This is achieved through demonstration of the analysis model, design and implementation of an ECG-based personal health monitoring service application for personal abnormality detections and classifications, and the illustration of experimental results. The proposed Adaptive Learning Approach (ALA) within the analysis model aims to reduce the training time necessary to master the application while showing improved performance over existing methods.

5.2. Problem Statement

The advancement of the Internet of Things and wireless sensors has paved the way for the development of new services for next-generation healthcare systems to enable superior communication between healthcare professionals. Healthcare is shifting from the traditional reactive approach—treating problems at the crisis level to a proactive health management approach in which issues can be discovered and addressed at an early stage. Being able to monitor the user's bio-signals over a longer term can contribute to an understanding of how their typical lifestyles and behaviours impact on their health status. More importantly, it allows for the tracking and discovery of any change-signals that could predict potential health issues. With the evolution of technologies such as wearable health devices and smartphones, more and more patients are able to continuously monitor their bio-signals at any time and in any location.

The scale and richness of the sensor data being collected and analysed is rapidly growing. However, there are still many challenges to face. For example, health sensor devices, such as Electrocardiogram (ECG), Electroencephalogram (EEG) and electrocardiography (EKG) monitors, employ a number of data channels and are capable of generating huge quantities of data. Consequently, visual inspection and interpretation of data on such massive scale is extremely difficult. There is still a lack of both support and tools to allow individual users to manage or utilise their collected data in practical ways, for example to monitor and track specific physical parameters, like heart rate, which could alert them to significant changes in their health conditions. Traditional approaches of collecting, storing, querying, visualising, and analysing health sensor data are no longer viable given the volume and diversity of the information that can now be monitored with contemporary technologies.

5.3. Related work

ECG is one of the most widely used physiological signals in helping doctors diagnose cardiovascular problems and identify abnormal levels of specific minerals in the blood. Traditionally, a standard ECG screening records the electrical activity of your heart for only a few minutes and is carried out in a hospital. If symptoms continue to occur without a definitive diagnosis obtained with a standard ECG, a Holter monitor will be used to record 24 to 48 hours of ECG signals, and then patients take the monitor back to their doctor. Efforts are being made to enable remote ECG monitoring systems. For example, the AliveCor device [22, 23] is one of a growing number of health sensors that not only allow users to record and view medical grade single-channel ECG signals on a smartphone/tablet, but also to upload and share their recordings directly with their physicians or cardiologists through the internet.

Considerable research has already been undertaken on various scopes for ECG data analysis such as QRS complex detection [239, 240] and beat classification algorithms [241-246]. Many of the scopes apply supervised-based learning methods on features obtained from RR-interval and QRS complex for identifying cardiac abnormalities. They are useful at a professional level; i.e. by doctors to classify ECG beats. However, for individual users, common mistake such as improper electrode placement and variability in inter-individual morphologies of ECG signals can lead to misinterpretation of ECG examinations using the traditional supervised approach. While considerable studies [247-250] have been undertaken on adopting unsupervised/semi-supervised learning methods that aim to detect abnormal (outliers) heartbeats, there is still a lack of research on how to enable health monitoring as a service in the cloud computing and big data environment in an efficient manner, with regards to the architecture, analysis models, feedback mechanisms and performance evaluations.

5.4. Proposed Data Analysis Method for Wiki-Health's ECG-based Health Monitoring Service

Figure 5.1 and Figure 5.2 illustrate the analysis workflow of a proposed ECG-based health monitoring service, containing three mechanisms: personal profiling, global training and personal abnormality detection. The personal profiling mechanism takes the user's previous ECG signals and builds a Personal ECG Model (PEM) for each user. Global training mechanism will take only the data containing expert diagnosed abnormal ECG signals and build a Global Model of Abnormalities (GMA). Through monitoring newly measured ECG data, the personal abnormality detection mechanism first compared new data with pre-trained PEM to detect if there are any suspicious signals that are significantly different from prior ECG readings, i.e. outliers. If so, it extracts these suspicious ECG signals and passes them into the GMA to try to classify the closest cardiac abnormality. The raw data of suspicious ECG signals together with classification are then sent to health professionals for visual inspection and feedback. Feedback includes classifying the suspicious ECG signals as usual signals and update the PEM for false positive alerts or call in the patient for further tests to diagnose if suspicious signals can be signs of issues. In the following paragraphs, we discuss each step of the analysis workflow and key elements of the method in detail.

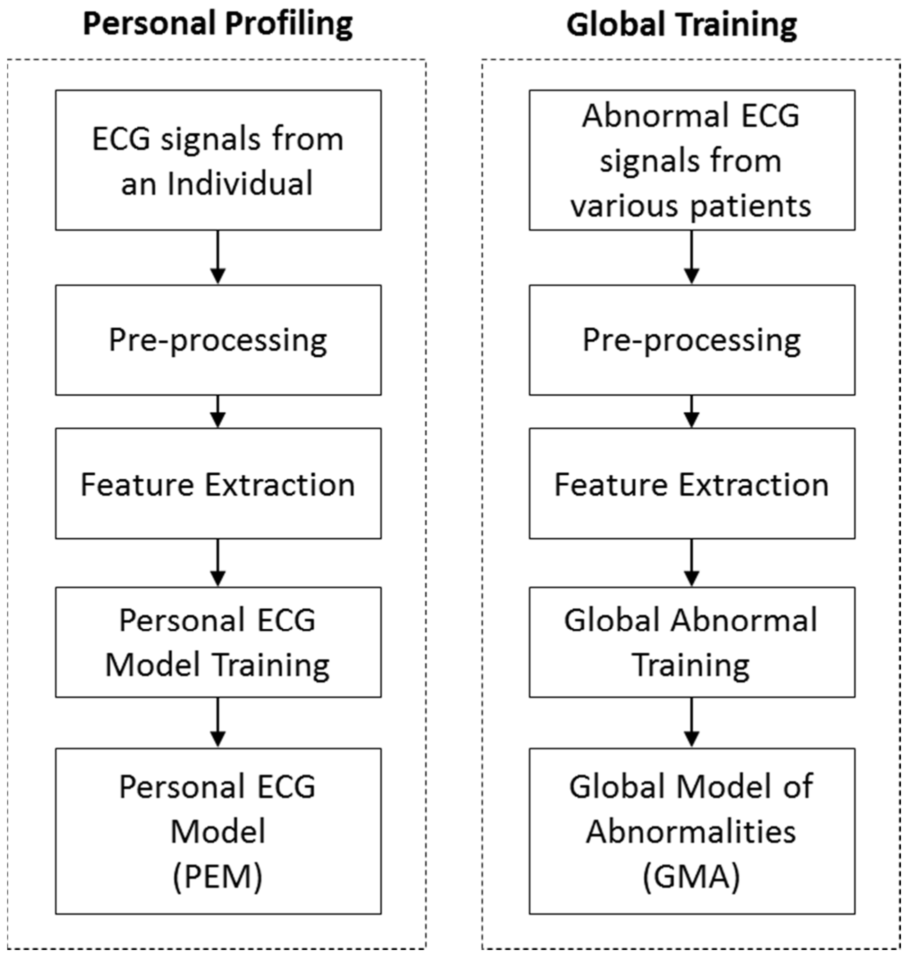


Figure 5.1: The workflow of personal profiling and global profiling mechanisms.

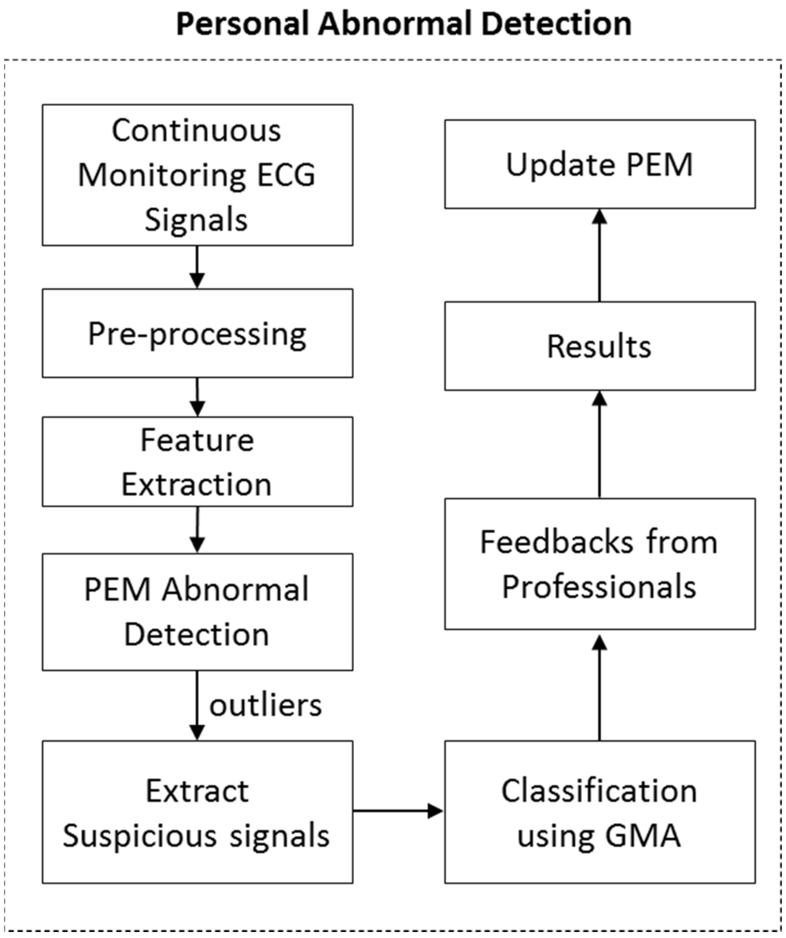

Personal Abnormal Detection
Continuous Monitoring ECG Signals
Pre-processing
Feature Extraction
PEM Abnormal Detection
outliers
Extract Suspicious signals
Classification using GMA
Feedbacks from Professionals
Results
Update PEM

Figure 5.2. The workflow of personal abnormality detection mechanism for health monitoring service

5.4.1. Data Pre-Processing

To reduce noise and eliminate baseline drift from the raw ECG signals, all the raw data is pre-processed by moving average filter and applying wavelet transformation (as shown in Figure 5.3).

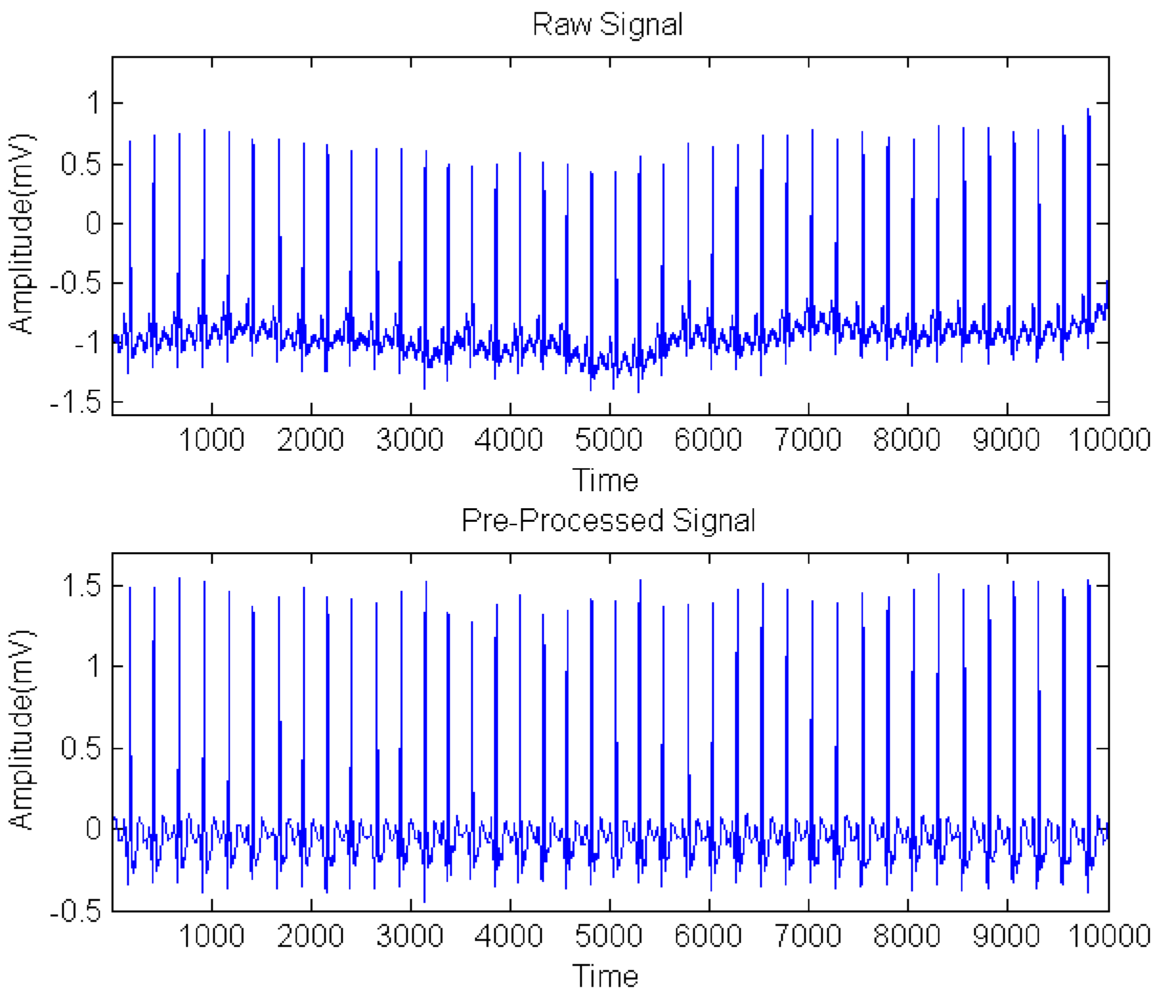


Figure 5.3. Sample beat waveforms of raw ECG signals and pre-processed signals.

5.4.2. Beat blocks

First, R-peaks are detected from the long ECG signals by implementing the ECG QRS detection algorithm by Pan and Tompkins[239]. We then adapted a beat block window to split the samples into beat blocks with R-peak centred in each block. The duration, amplitude, and shape of the QRS complex are vital in diagnosing disease conditions. A significant change in QRS complex can indicate something has happened that affects the beating of the heart. The size of the beat block window is set as 0.6 seconds, which should capture most of the information from a single heartbeat cycle (as shown in Figure 5.5).

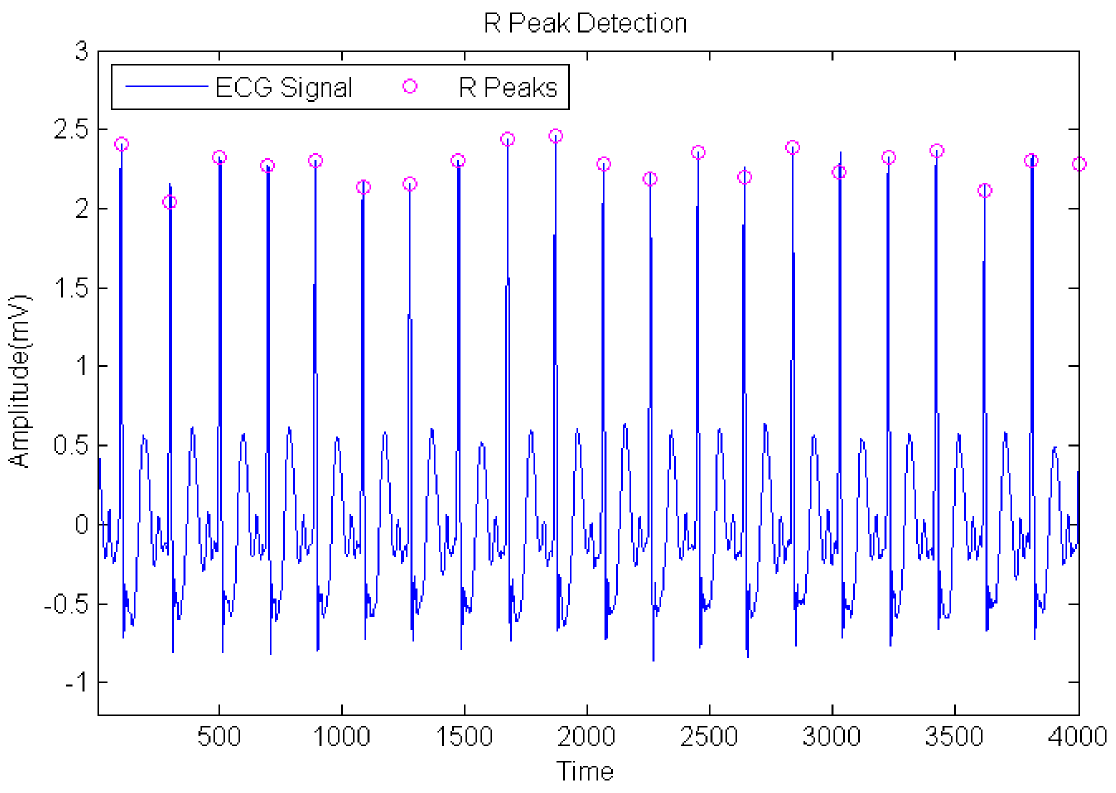


Figure 5.4: R Peak Detection on ECG signals.

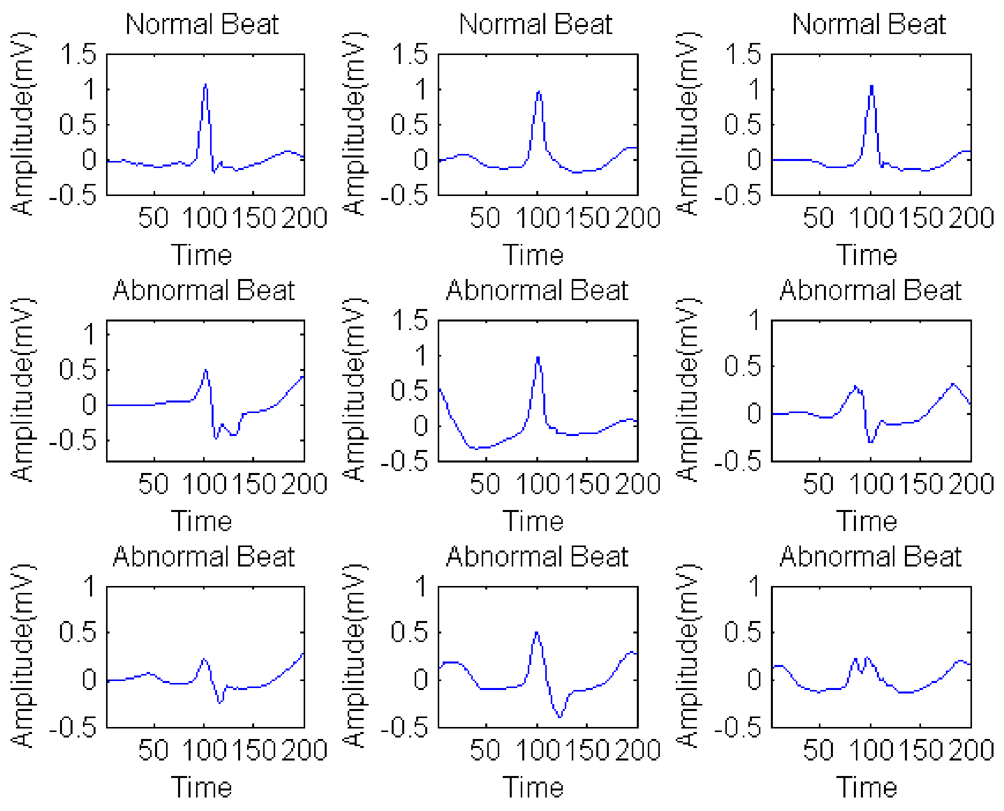


Figure 5.5. Samples of normal and abnormal beat blocks.

5.4.3. Features

In this work, it has been determined that we explore the combination of signal and spectral features extracted from the beat blocks for classification. For each beat block, signal features are calculated by neglecting the time t coordinate entirely and consider the normalized signal $x(t)$ as a distribution over x. The signal data from the first channel of the MIT-BIH database Physiobank [237, 238], is used for designing and testing the analysis model. The signal information contained in this database was obtained by placing the electrodes on the chest of the subject being monitored, i.e. using a modified limb lead II (MLII). The MIT-BIH database contains two-channel ECG signals together with corresponding rhythm annotations and signal quality comments of the sample contributed by cardiologists. For consistency of comparison, signal bin boundaries are set between -0.3 and 0.5, which should be able to capture most amplitude distribution of QRS complex for MLII signals. We currently use 60 linearly spaced bins for signal features. For other lead types, different sample rates, or changes to the window used for features, a different range might be required.

In each ECG beat block, every peak produces a significant response in frequency domain. Spectral features are calculated by applying the one-sided discrete Fourier transform (ODFS) of each beat block.

$$\mathrm{DFT(f)} = \hat{\mathrm{F}}(\omega_j) = \sum_{t_i=t_{min}}^{t_{max}} \mathrm{f}(t_i)e^{-2\pi i t_i \omega_j} \qquad (1)$$

Where f is the original signal, t_i are the time points at which the ECG signal is sampled, ω_j is the frequency in HZ. Compared to the commonly used fast Fourier transform (FFT), ODFS offers the ability to adjust frequency resolution and frequency bands, independently on the resolution of the signal. If the signal is real, ODFS will omit the redundant parts of the negative frequency spectrum.

We apply the one-sided DFT with a pass band of [0,20] Hz. Typical signals contain relatively few peaks, so the high-frequency information is mostly noise. The computed spectrum of $\hat{\mathrm{F}}$ has 40 complex coefficients. We use both their real and imaginary parts as features, giving us 80 spectral features.

5.4.4. One-Class Support Vector Machine

One-Class Support Vector Machine (OCSVM) [251] is a novelty detection method that attempts to find a hyperplane in the feature space corresponding to a kernel function that separates data points from the origin and maximises the distance from this hyperplane to the origin. Different from classical SVM, it uses two classes for training. OCSVM only requires data from one class in the training stage and is able to classify if newly encountered data belong to the same class or not. Therefore, it is used for detecting personal abnormal or unusual beats. Consider a single class training set $x_i \epsilon \Re^m, i = 1, \dots, n$, each x_i is associated to a class label $y_i \epsilon \{+1\}$, n is the number of training instances, the objective function of the OCSVM model can be written as follows:

$$\min \quad \left(\frac{1}{2} w^T w - \rho + \frac{1}{\nu n} \sum_{i=1}^{n} \xi_i \right) \tag{2}$$

$$\text{s.t. } (w \cdot \phi(x_i)) \geq \rho - \xi_i, \xi_i \geq 0 \text{ for all } i = 1, \dots, n$$

ϕ is the mapping function to map the sample vectors to a higher dimensional feature space. ν is the trade-off parameter to specify the fraction of outliers allowed. By setting the parameter $\nu \epsilon (0,1)$, it gives us a way to adjust the trade-off between over fitting and generalization. ξ_i are introduced slack variables relax the optimality constraints. w and ρ are hyperplane parameters that we want to compute to minimise the objective function (2). Then, the decision function $f(x) = sgn(w \cdot \phi(x) - \rho)$ will be positive for most training points x_i.

5.4.5. Adaptive Learning Approach (ALA) for PEM

Suppose we are only using one classifier. When new training data arrives, the traditional way is to add new data to the existing training data and then re-train the model using all of the data. The training time often increases linearly as the size of the training samples grows. Thus, this type of system is not feasible for near real-time analysis or large-scale data. Therefore, we propose a simple and effective Adaptive Learning Approach (ALA) for PEM. In this approach, ALA constructs a cluster of weighted classifiers. Instead of re-training one model with all the existing data, ALA adds new classifiers that are trained using only the new data, and updates the weights of classifiers based on the previous prediction results, i.e. feedback from doctors. The prediction is accomplished by taking the votes from all the available classifiers from the cluster.

A person's health status may also change over time, being influenced by factors such as alcohol consumption, diet, tobacco smoking, medications, and pregnancy. Such temporal evolution features are important in identifying the correct event, stimulus or cause associated with the corresponding model or sequence of sensor data. Integrating ALA with Wiki-Health not only provides the computational capabilities of training and making predictions, but also allows the trained models and additional information such as prediction logs, to be persisted, tagged, searched, and reused for any future studies.

The parameters and the procedures of ALA are defined as the following:

Parameters	**Semantics**
H	A cluster of trained and weighted OCSVM classifiers: $H = \{w_1 h_1, \dots, w_T h_T\}$
T	The size of cluster H
h_t	A trained OCSVM classifier from H where $h_t \epsilon H, t \epsilon [1, T]$
w_t	Weight for classifier h_t

α_t	Total number of correct predictions by h_t
λ_t	Total number of samples trained by h_t
ϑ	$\boldsymbol{\vartheta}$ is the maximum number of training samples that a single OCSVM classifier can take
β	Minimum number of samples to be reached before removing a classifier
γ	Accuracy threshold for removing the classifier
x^i	Feature vectors, $x^i \epsilon\ \mathcal{X}$
y_i	Binary label, $y_i \epsilon \{-1, +1\}$
L	Last training set which h_T used to train.

Table 5.1: Parameters and semantics of ALA

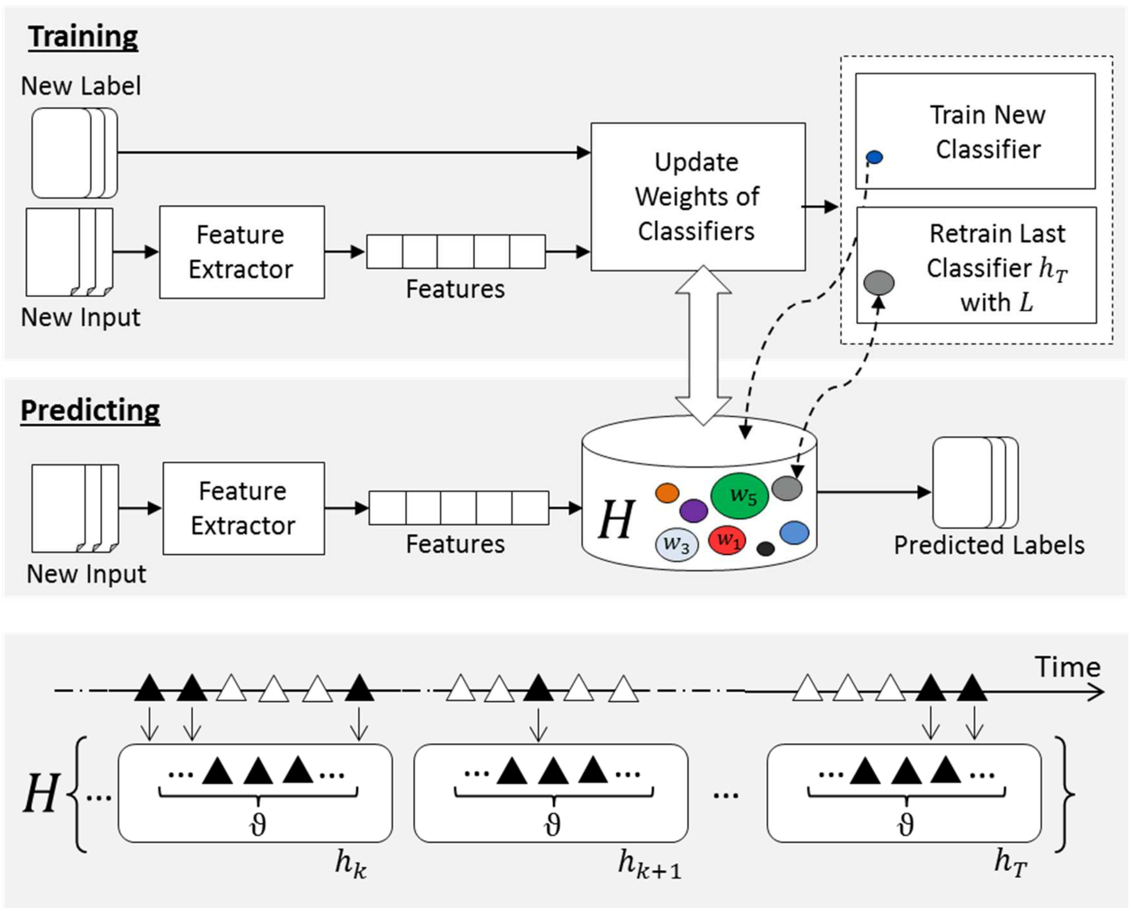


Figure 5.6. The overview of ALA for a long-term health monitoring service.

Procedure 1 illustrates how ALA updates the PEM model. When new training data arrives, it goes through every classifier from H and updates the corresponding weight. Numbers of correct predicted total predicted times are stored and updated. The weight is increased with more correct predictions and decreased with incorrect ones. Constants such as β and γ are introduced to eliminate "out-of-date" classifiers from H; i.e. no longer able to make good predictions. A boundary ϑ is configured to limit the total number of training samples a classifier can train and also decides when to train and add a new classifier. To make the procedure simple to understand, we only demonstrate the pseudo code for a single input of x and y; it can be easily modified for batch inputs of x and y.

Procedure 1: Update PEM Model

Inputs: $\boldsymbol{x}, \boldsymbol{y}$

x	Training features
y	Binary label corresponding to x

```
1.  Begin
2.  $T$=sizeof ($H$)
3.  for $t$=1:$T$
4.      prediction=model_predict($h_t$, $x$)
5.      if (prediction== y)
6.          $\alpha_t$=$\alpha_t$+1
7.      end if
8.      $\lambda_t$= $\lambda_t$+1
9.      set $w_t$= $\alpha_t$/ $\lambda_t$
10.     if ($w_t$<$\gamma$ and $\beta \leq \lambda_t$)
11.         remove $H(t)$
12.     end if
13. end for
14. append $x$ and $y$ to $L$
15. if (sizeof ($L$) == $\vartheta$)
16.     set T=T+1
```

```
17.     train model $h_T$ with $L$
18.     add $h_T$ to $H$
19.     set $\lambda_T$=sizeof($L$), $w_T$= $eps$, $\lambda_T$=0, $\alpha_T$=0, $L$={} //eps is floating-point relative accuracy
20. end if
21. return $H$
22. End
```

Procedure 2 illustrates how ALA is used to detect outliers. Only positive weighted classifiers are elected to vote for the results. The voting power depends on the weight of each classifier. The sign of the combination votes decides the final predictions.

Procedure 2: Outlier Detection (Voting by classifiers from *H*)

Inputs:

x	Features for prediction

```
1. Begin
2. $T$=sizeof ($H$)
3. for $t$=1:$T$
4.     local_prediction= model_predict($h_t$, x)
5.     votes= votes + $w_t$ × local_prediction;
6. end for
7. predictions = sign(votes);
8. return predictions
9. End
```

5.4.6. Classification for GMA

A K-Nearest Neighbor (K-NN) algorithm is used as the classifier for the global training and classifications for the GMA. The same features are used in both the PEM and the GMA. To determine the nearest neighbour, various distance measures can be used. For example, between two points $a = (a_1, a_2, \dots, a_k,)$ and $b = (b_1, b_2, \dots, b_k,)$, with k dimensions, these distances can be calculated as follows:

$$\text{Euclidean distance} = \sqrt{\sum_{j=1}^{k}(a_j - b_j)^2}$$

$$\text{City Block distance} = \sum_{j=1}^{k}\left|a_j - b_j\right|$$

$$\text{Correlation distance} = 1 - \frac{\sum_{j=1}^{k}(a_j-\overline{a})(b_j-\overline{b})}{\sqrt{\sum_{j=1}^{k}(a_j-\overline{a})^2}\sqrt{\sum_{j=1}^{k}(b_j-\overline{b})^2}}$$

5.5. Integration with Wiki-Health Platform

Figure 5.7 shows the overview of the proposed health monitoring service application deployed on top of Wiki-Health platform. The Wiki-Health Mobile App can be used for collecting and uploading sensor data, receiving alerts and communicating with health professionals.

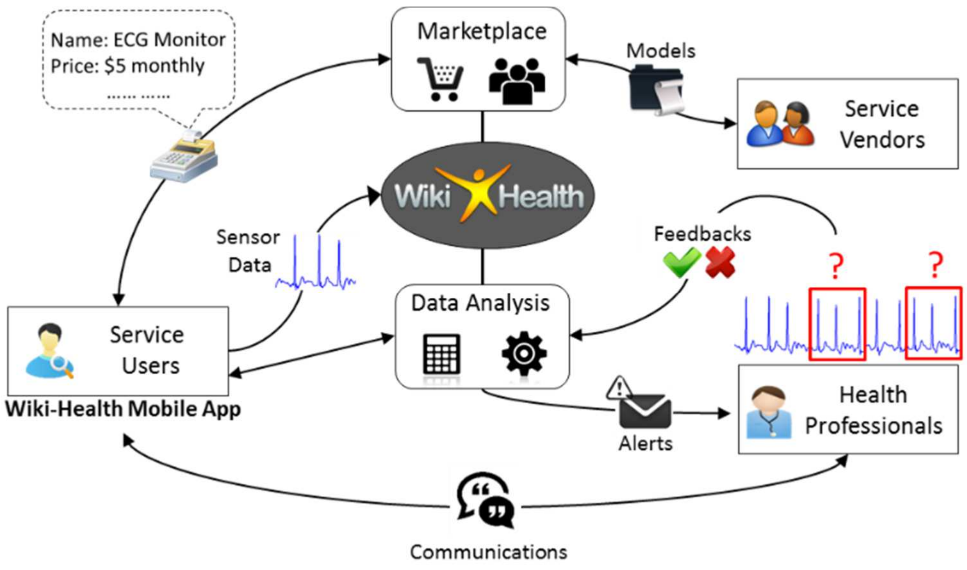


Figure 5.7. Implementation of ECG-based health monitoring service with Wiki-Health platform.

All data processing functions and analysis models presented in the chapter are developed in MATLAB [109]/Octave programming language. LIBSVM library [252] is used for implementing the OCSVM. Task Scheduler is currently implemented using Java Quartz Scheduler [235]. All the tasks are defined using time-based CRON expressions. The executing function files are uploaded into Wiki-Health's model repository and published as a health monitoring service in the data and models marketplace. Task Scheduler is used to launch different tasks based on pre-defined schedules, such as detecting and labelling R peak for new data every 2 minutes, detecting personal abnormalities for R peak labelled data every 5 minutes, and sharing suspicious signals and predicted labels with doctors every 6 hours.

For users, the enabling of a personal health monitoring service with Wiki-Health requires almost no knowledge about the computation. The user only needs to select the service from the data and models marketplace, configure the mappings of inputs and outputs (such as selecting the ECG data stream) to be used for the health monitoring service and enter the email address of the doctor for triggered alerts.

5.6. Experimental Evaluation

This section first describes an experiment used to test the performance of our proposed analysis method and features, then evaluates the effectiveness of ALA from different perspectives, and finally accesses the performance of ECG classifications using the global model of abnormalities (GMA).

ECG records from the MIT-BIH database [237, 238] are used for the experiments. These records were captured under ideal conditions, in other words, with the subject sitting or lying down. Each record contains two channels of 30-minute recordings, which were digitised at a sample rate of 360 samples per second and there are approximately 650,000 samples per record per channel. For our experiments, the signal data from the first channel (modified limb lead II) is used. The entire database is separated into two groups. The first group of 23 records were used for the evaluation of Part A and B. The remaining records are assigned as the second group, which is used as a training set for GMA in Part C.

True positive (TP) refers to the number of abnormal ECG beats that are correctly detected. True negative (TN) refers to the number of normal beats that are correctly labelled. TB refers to the total number of beats. The following statistical parameters are used to compare the detection algorithms:

- $Classification\ Rate\ (CR) = \frac{TP+TN}{TB}$
- $Specificity\ (SP) = \frac{TN}{FP+TN}$
- $Sensitivity\ (SE) = \frac{TP}{TP+FN}$

Where Classification Rate (CR) is the fraction of both normal and abnormal ECG beats that are correctly classified among all the ECG beats analysed. Specificity (SP) is the fraction of normal ECG beats that are correctly classified among all the normal ECG beats, Sensitivity (SE) is the fraction of abnormal ECG beats that are correctly detected among all the abnormal ECG beats.

The results were tested with different kernels, and the summary is displayed in Table 5.2. A 5-fold cross-validation is used for each training set to select the parameters. LIBSVM's degree variable is set to 3 for the Polynomial kernel, while the gamma variable is set to be 0.001 for Gaussian RBF kernel [253]. The trade-off parameter ν is set to be 1% for all tests (at most, 1% of the training samples being misclassified). The detailed results from Table 5.5 are obtained using Gaussian radial basis function kernel (RBF).

	Classification Rate			Specificity			Sensitivity		
	SF	VS	ALA	SF	VS	ALA	SF	VS	ALA

Linear	81.5	95.2	89.5	79.3	97.3	88.6	95.8	85.0	94.7
Polynomial	86.7	96	92	85.7	98	92	94.2	83	93
Gaussian	85.5	96.3	91.8	81.1	98.5	91.1	98.1	85.2	96

Table 5.2: Comparison of overall performance of PEM using different Kernels. All items are in percent (%).

5.6.1. (Part A) Using single fixed one-class SVM

The purpose of this experiment is to test the performance of the selected features with a single fixed OCSVM (SF). The first group of 48 records are split into two sets. To simulate the scenario of using a small portion of a user's typical previous ECG shapes and detecting personal abnormalities, for each record, after the first 3 minutes of the data is used for training the fixed OCSVM model and the remaining 27 minutes as the testing data.

As shown in Table 5.5, SF provides an average classification rate of 85.5%, specificity of 81.1%, and sensitivity of 98.1%. SF's classification rate and specificity are not as good as the sensitivity for some patients such as 116, 202, 205, 222 and 223. For ECG-based health monitoring services, sensitivity is important so that we capture and highlight more alerts than missing one. In the next experiment, we demonstrate how ALA can improve classification rate and specificity while still maintaining high sensitivity based on feedback mechanisms.

5.6.2. (Part B) Effectiveness of the Adaptive Learning Approach

To simulate the scenarios of receiving feedback from the professionals, we created two different cases for PEM:

- VS: using a single OCSVM as the classifier for PEM to trigger the alerts (i.e. abnormalities), these alerts are compared with the labels and the false positives are sent back. The PEM is then re-trained with existing normal data together with newly added false positives.

- ALA: using the Adaptive Learning Approach by creating a cluster of weighted OCSVMs for PEM to detect personal abnormalities. Weights are adjusted and newly trained classifiers are added to the cluster when false positive alerts are triggered. Constants ϑ and β are both set to 250, with γ set to 0.6.

Similar to the previous experiment, the first group of 23 records are also split into two sets. For each record, the first 3 minutes of the data are used for training and the remainder for testing and feedback. As shown in Table 5.5, classification rate and specificity are increased for both cases. However, the sensitivity of VS is decreased more than 10% as the size of the training samples grows. ALA is able to achieve 96.6% sensitivity.

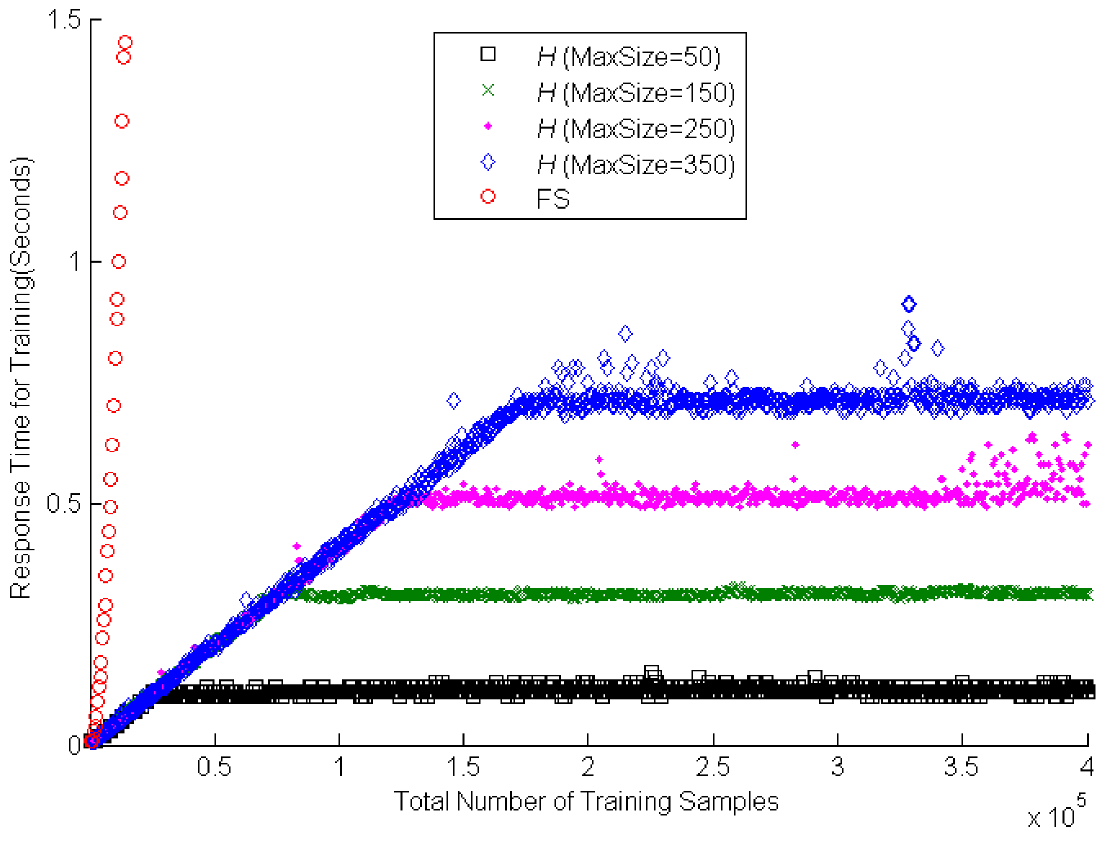


Figure 5.8. Relationship between system response time for training and the number of training samples.

In comparing the computation time between VS and ALA, we simulated the scenario of adding 500 training samples each time for VS and ALA. To keep experimental results consistent, the entire computation was performed in the VM on top of IC-Cloud with two allocated cores and 8GB memory. Each time, a new classifier is trained and added until the size of H reaches the maximum. Different maximum sizes are used for H. The comparison of the system response time for training is shown in Figure 5.8. Compared to VS, ALA reduces the time of training and updating the PEM model significantly. A similar test was completed to evaluate the system response time for prediction. As shown in Figure 5.9, the prediction time for ALA is dependent on the size of cluster H and it increases linearly with the number of predicting samples.

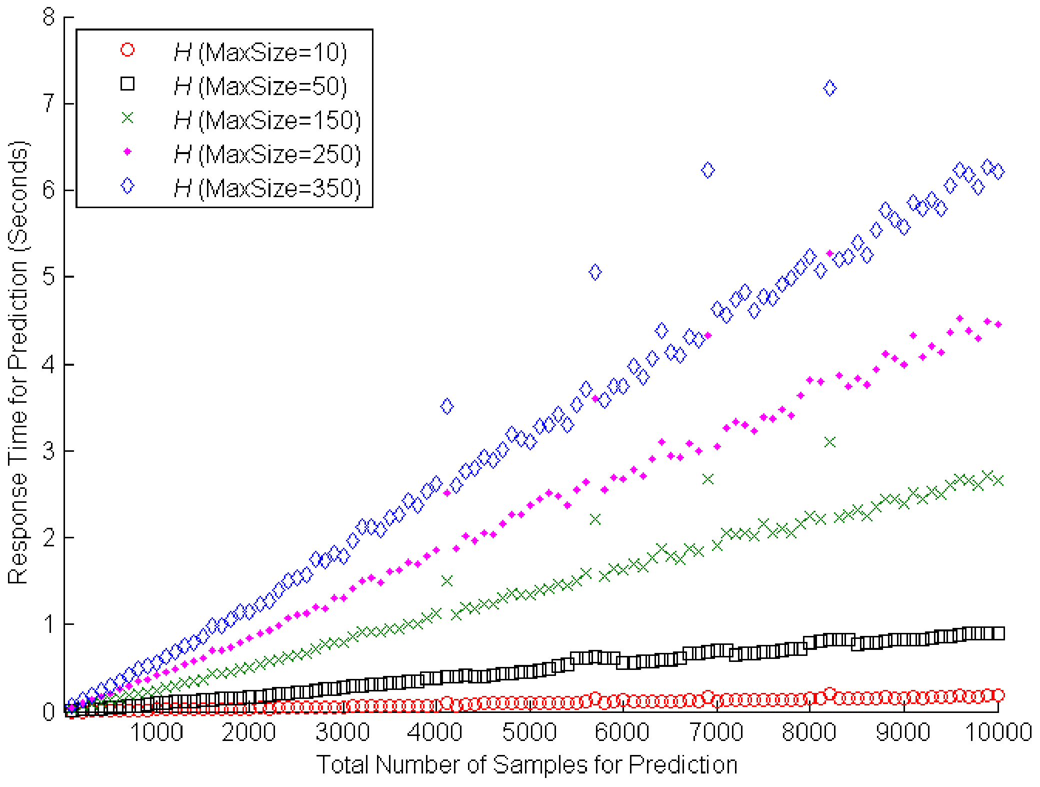


Figure 5.9. Relationship between system response time for prediction and the number of samples.

5.6.3. (Part C) Performance of Global Model of Abnormalities

Distance metrics such as Euclidean distance, City Block distance and Correlation distance have been discussed in the earlier section describing the GMA. In this experiment, different distance metrics and values of K are used to test the performance of the GMA. The following most common types of beats are extracted: atrial premature beat (A), ventricular premature beat (V) and right bundle branch block beat (R) from the second group of data to form the training data. The triggered beats from Part A are used as the test data. The results shown in Table 5.3 and Table 5.4 are obtained by applying three different distance metrics and five different values of K. The best result is obtained using City Block distance (CB).

Distance Metrics	K=1	K=2	K=3	K=4	K=5
City Block	82.2	72.9	71.9	72.3	71.3
Euclidean	78.5	72.7	70.2	71.4	69.9
Correlation	77.6	72.5	69.9	71.0	69.7

Table 5.3: Comparison of accuracy between different distance metrics and values of *K* for GMA. All items are in percent (%).

Beat Types	A	R	V	Total
Train Beats (Group 2)	1642	4176	1669	7487
Test Beats (Group 1)	785	2795	4666	8246
Correctly Labelled	506	2012	4258	6776
Accuracy (%)	64.5	72.0	91.3	82.2

Table 5.4: Performance of GMA for different beat types using City Block distance metric.

Record			CR (%)			SP (%)			SE (%)		
ID	N	NN	SF	VS	ALA	SF	VS	ALA	SF	VS	ALA
116	2066	105	73.9	98.6	76.5	72.6	98.5	75.3	100.0	100.0	100.0
119	1386	494	93.3	99.1	93.6	90.9	98.8	91.3	100.0	100.0	100.0
201	1422	408	88.5	96.8	94.3	86.1	98.4	94.2	96.8	91.2	94.9
202	1849	82	29.6	97.3	83.3	26.5	98.5	83.3	100.0	69.5	82.9
203	2276	468	91.4	96.6	92.9	90.3	97.1	91.8	97.0	94.2	98.1

205	2304	98	74.1	97.6	89.9	73.0	98.0	89.5	100.0	88.8	99.0
208	1430	1282	92.8	94.7	94.8	88.7	97.2	92.4	97.4	91.8	97.4
209	2325	403	94.8	85.7	94.9	95.4	98.5	96.0	91.1	11.4	88.8
210	2176	223	93.2	98.1	95.5	92.5	98.6	95.1	100.0	92.8	99.6
212	793	1682	99.5	99.7	99.5	98.4	99.2	98.4	100.0	99.9	100.0
213	2361	601	84.9	94.7	93.6	83.0	99.2	94.5	92.3	77.0	89.7
215	2881	149	96.4	98.9	97.3	96.2	98.8	97.2	100.0	100.0	100.0
217	242	1806	93.0	99.1	93.0	40.5	93.4	40.5	100.0	99.8	100.0
219	1865	212	87.8	98.5	91.9	87.3	99.1	91.4	91.5	92.9	96.7
220	1747	109	54.1	98.0	85.3	51.2	98.1	84.5	100.0	96.3	98.2
221	1831	373	98.5	99.4	98.3	98.1	99.3	98.0	100.0	100.0	100.0
222	1799	556	56.2	79.3	70.6	44.4	95.8	67.2	94.2	26.1	81.7
223	1795	573	79.6	94.0	90.2	73.4	98.5	88.5	98.8	79.9	95.3
228	1540	365	97.6	98.3	96.9	97.3	98.2	96.4	99.2	98.6	99.2
230	2024	191	97.9	99.1	95.4	97.7	99.0	95.0	100.0	100.0	100.0
231	302	1122	97.8	98.9	98.0	89.7	95.0	90.7	100.0	100.0	100.0
233	2021	814	98.3	99.3	98.5	98.0	99.4	98.0	99.1	99.1	99.8
234	2424	55	93.9	99.0	96.0	93.8	99.0	95.9	100	100	100
Average			85.5	96.5	92.2	81.1	98.1	88.9	98.1	87.4	96.6

Table 5.5: Comparison of performance of SF, VS and ALA using Gaussian RBF kernel. N: Normal, NN: Abnormal

5.7. Analysis Method Discussion

Supervised learning is common in classification problems; it takes a known set of input data and known responses to that data, and seeks to build a predictor model that can generate reasonable predictions in response to new data. To accomplish this, labeled training data is required. Many supervised learning algorithms exist such as support vector machines (SVM), neural networks [254], decision trees [255], Naive Bayes, and Nearest Neighbor [256]. However, in many health monitoring applications, training data is hard to label because some or all the training data belong to one class (normal rhythms), so we don’t know what the data belonging to other classes looks like (arrhythmia). Unsupervised and semi-supervised learning algorithms try to find hidden structure in unlabeled data. These kinds of algorithms assumes there is no pre-discovered classification of examples. Common unsupervised and semi-supervised learning include: k-Means clustering [257], Gaussian mixture models [258], Hidden Markov models [259], and One-Class SVM.

In this study, in addition to using the One-Class SVM learning, we have also attempted to make use of k-Means clustering algorithm. Unfortunately, based on some initial testing with data from several patients' data in MIT-BIH database, k-Means does not perform well with our selected features. In future work, we may investigate different combination of features and algorithms such as Gaussian mixture models.

The Adaptive Learning Approach proposed in this study belongs to the category of boosting of machine learning methods [260], in which the output is calculated using several different models and the result is averaged using a weighted average approach. Other approaches may also improve the performance of the model. For example, bagging (i.e. Bootstrap Aggregation) [261, 262] is an approach used to decrease the variance of the prediction by generating additional data for training from your original dataset using combinations with repetitions to produce multisets of the same size as the original data. Increasing the size of your training set, reduces variance and helps avoid overfitting. The use of the bagging method may also be investigated in the future work.

5.8. Summary

This chapter presented the key components for enabling "Health Monitoring as a Service" on the Wiki-Health platform. An ECG-based health monitoring service application is illustrated with a specific focus on an analysis model applicable to the big data environment of mobile phones and health sensing devices. The possibility of detecting personal ECG abnormalities using the One-Class Support Vector Machine is investigated and proven to be effective using the combination of beat blocks and selected features. The proposed Adaptive Learning Approach (ALA) within our analysis model was able to detect personal abnormalities and update the model based on received feedback. We also found the ALA approach was successful in improving classification rate and specificity while still maintaining high sensitivity. A comparison was also made to determine the training time required for updating the model and predicting performance. The positive performance of the approach is supported by the results obtained in experimental trials and shows significant potential for real-world applications.

6. Enabling Virtual Sensing as a Service with Wiki-Health Platform

6.1. Introduction

Exposure to hot and cold environments can be not only hazardous to our health but also life threatening [263], especially when extensive workloads are being carried out. Being able to monitor human core body temperature and other vital signs can provide a way of reducing the risk for these workers. Although the developments of modern technologies such as cloud computing, wearable sensor devices and big data have provided a means of continuous, non-intrusive monitoring of our physiological signals and vital signs, in many situations, placing the physical sensor such as esophageal and rectal probes at the ideal location to acquire the sensing information like temperature is challenging [264]. Virtual sensors, as opposed to physical sensors, can provide indirect measurements by using other available sensor data. Although a few studies have been undertaken on implementing virtual sensors for specific applications, there is still a lack of research on utilising virtual sensors for healthcare.

In this chapter, we present the design and implementation of a virtual sensing application that can be deployed as a service on top of the Wiki-Health platform, using the functionalities of Wiki-Health by integrating human physiological models with collected data from available sensor readings to compute and simulate virtual sensor readings of human body temperatures. The proposed service demonstrates the feasibility and potential of using the Wiki-Health framework to make health-related knowledge discovery more readily available to the general public.

6.2. Related Work

In this section, we first present the related research in virtual sensors for different domains and then discuss the key background information on human body core temperature measurements.

6.2.1. Virtual Sensors

Virtual sensors, as opposed to physical sensors, can provide indirect measurements by using other available sensor data, models and knowledge. There have been a few proof-of-concept studies, examining the feasibility of virtual sensors for various applications. Stéphant et al. [265] used linear and nonlinear models to develop virtual sensors to simulate and compare vehicle sideslip angle with actual road tests. Kusiak et al. [266] constructed a virtual wind-speed sensor for wind turbines by using models built from historical wind-farm data and data-mining algorithms. Raveendranathan et al. [267] proposed a virtual sensor framework in the gait analysis domain for the purpose of enabling real-time activity and posture recognition. In their 2011 study, van Beek et al. [268] simulated the time trial of athletes in the Tour de France with various physiological and mechanical models.

6.2.2. The Importance of Human Core Body Temperature

Workers in a number of occupations are at high risk of illness related to extreme temperature exposure (high heat and intense cold), including construction builders, telecommunication lineman, fire fighters, transport drivers, athletes, refrigerated warehouse workers, and military crews. Human core body temperature is a vital sign measuring one of the body's most basic functions. Too high or too low core body temperature can cause conditions such as hyperthermia and hypothermia. Being able to monitor human core body temperature can provide a way of reducing the risk for these workers. Although a variety of wearable sensor devices have been developed in the recent years, offering the capability of capturing different physiological parameters at real-time, in many situations, placing a physical sensor, such as an esophageal or rectal probe, at the ideal location to acquire the sensing information is still very difficult [264]. In comparison to core temperature, heart rate and other environmental measurements can be more easily recorded. In this study, we propose a virtual sensing service that aims to estimate human body temperatures by integrating human physiological models with measurable sensor data.

6.3. Virtual Sensing as a Service Overview

Virtual sensors can be useful in many situations such as simulating the performance of athletes and workers under extreme physical conditions; obtaining real-time indirect measurements using other measured variables and triggering alerts. For this study, virtual sensors are constructed as a new service layer adopting the functionalities of Wiki-Health platform (as shown in Figure 6.1). Data collected from the Wiki-Health Mobile App and physical sensors can be used for data analysis and simulation purposes. For users, the enabling of a virtual sensing service with Wiki-Health requires almost no knowledge about the computation. The user only needs to select the service from the data and models marketplace, configure the mappings of inputs and outputs (such as selecting the physical sensor streams to be used for the virtual sensing service) and enter the mobile phone number or email address of the health supervisor for triggered alerts.

In the following sections, we illustrate the thermoregulation and physiological models of humans, demonstrating how these models can be integrated with available measurable sensors to implement the proposed virtual sensing service.

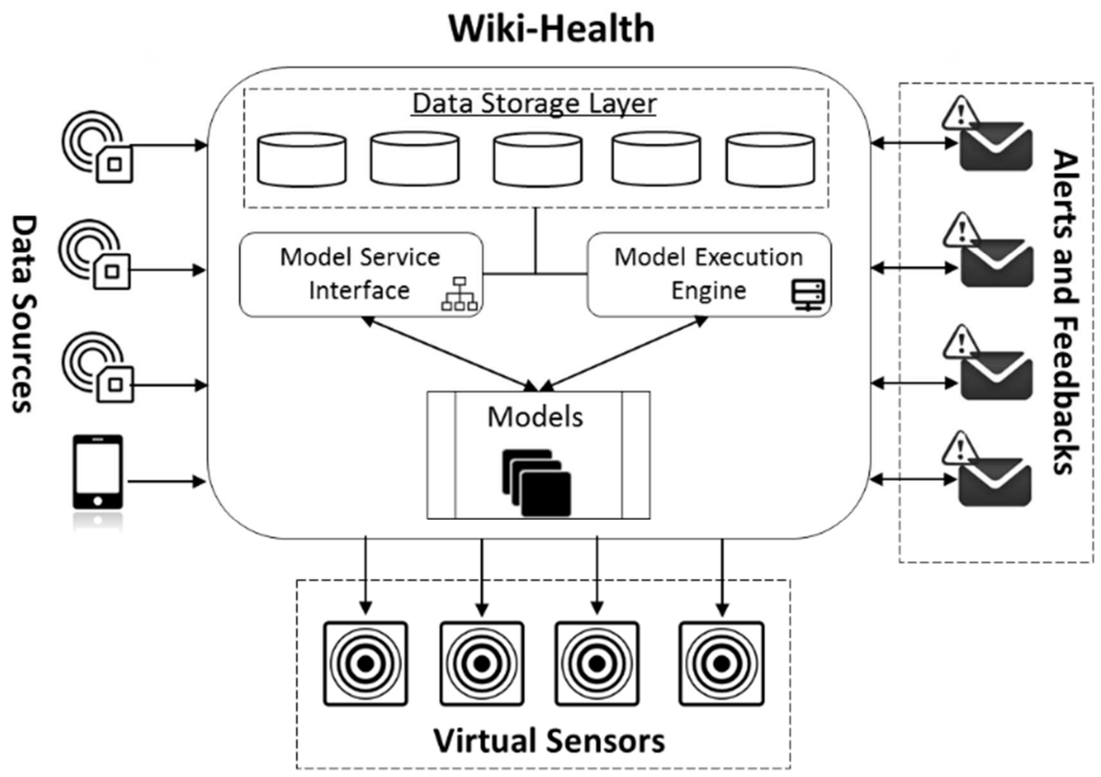


Figure 6.1: The proposed implementation of virtual sensors on Wiki-Health.

6.4. Full body model associated with human temperature and energy transformations

Humans, like other mammals, are able to maintain a stable body temperature in different environments. The thermoregulatory system of human body is a complex, sophisticated system that detects, interprets and reacts to changes in internal and external environment conditions. The mechanisms of thermoregulation are designed to keep us healthy within narrow body temperature limits, generally between 36.5°C to 37.5 °C [269]. Humans may experience both physiological and psychological signs related to body temperatures significantly above or below that range. The regulation of human body temperature can be affected by many factors, such as climate, illness, physical activity level, emotions, age and clothing. Understanding the human thermoregulatory system can help us save time and effort in identifying potential health risks and conditions of danger.

A significant volume of research has already been conducted by scientists with regard to human thermoregulation, which includes studies focused on modelling, simulating and predicting thermoregulatory behaviours in humans.

A mathematical multi-node model of human physiology proposed by Stolwijk [270] and employed by NASA in the Apollo program, has been one of the most influential works, and has served as the basis for many physiological research studies aimed at establishing a more comprehensive human thermoregulatory model. In this section, some key equations and parameters of Stolwijk's model [270] are presented.

6.4.1. The Controlled System

In Stolwijk's model, two separate systems are proposed: the controlled system and the controlling system. The controlled system of the human body consists of six segments: head ($i = 1$), trunk ($i = 2$), arms ($i = 3$), hands ($i = 4$), legs ($i = 5$) and feet ($i = 6$). Each of the six segments has four layers: skin, fat, muscle and core. In total, they contribute up to 24 nodes (as shown in Figure 6.2). Central blood is considered the 25th node which flows through the other 24 nodes and exchanges heat with the body tissues.

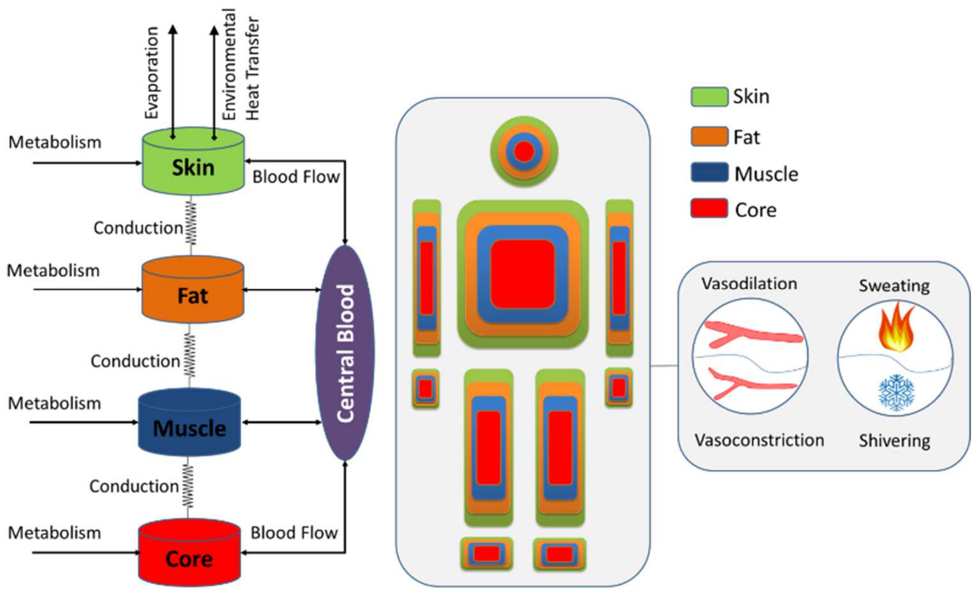


Figure 6.2: The multi-node model of human body temperature regulation.

Symbol	*Meaning*	*Unit*
$T[N]$	Temperature of node N	°C
$Q[N]$	Total metabolic heat production in node N	kcal/hour
$E[N]$	Total evaporative heat loss from node N	kcal/hour
$BC[N]$	Convective heat transfer between central blood and node N	kcal/hour
$TD[N]$	Conductive heat transfer between node N and adjacent node $N + 1$	kcal/hour
$HR[N]$	Radiant heat transfer coefficient of segment N	kcal/meter^2/hour/°C
$HC[N]$	Convective and conductive heat transfer coefficient of segment N	kcal/meter^2/hour/°C
$H[N]$	Total combined heat transfer coefficient of segment i	kcal /hour/°C
S	Total surface area of segment i	meter^2
$VAIR$	Air velocity	meter/second
$TAIR$	The temperature of surrounding environment	°C

Table 6.1: Definition of symbols in the human body temperature regulation model

Node	*Segment*	*Layer*	*C*	*QB*	*EB*	*BFB*	*TC*
(N)	*(i)*		*(kcal/°C)*	*(kcal/°C)*	*(kcal/°C)*	*(Litre/Hour)*	*(Kcal/Hour//°C)*
1	Head	Core	2.22	12.84	0.00	45.00	1.38
2		Muscle	0.33	0.10	0.00	0.12	11.40
3		Fat	0.22	0.11	0.00	0.13	13.80
4		Skin	0.24	0.08	0.63	1.44	0.00
5	Trunk	Core	9.82	45.38	9.00	210.00	1.37
6		Muscle	16.15	5.00	0.00	6.00	4.75
7		Fat	4.25	2.13	0.00	2.56	19.80
8		Skin	1.21	0.40	3.25	2.10	0.00
9	Arm	Core	1.41	0.70	0.00	0.84	1.20
10		Muscle	3.04	0.95	0.00	1.14	8.90
11		Fat	0.58	0.17	0.00	0.20	26.20
12		Skin	0.43	0.13	1.20	0.50	0.00
13	Hand	Core	0.14	0.08	0.00	0.10	5.50
14		Muscle	0.06	0.20	0.00	0.24	9.65
15		Fat	0.09	0.03	0.00	0.04	9.90
16		Skin	0.17	0.05	0.45	2.00	0.00
17	Leg	Core	4.24	2.23	0.00	2.69	9.00
18		Muscle	9.17	2.86	0.00	3.43	12.40
19		Fat	1.43	0.43	0.00	0.52	64.00
20		Skin	1.08	0.32	2.85	2.85	0.00
21	Feet	Core	0.23	0.13	0.00	0.16	14.00
22		Muscle	0.06	0.02	0.00	0.02	17.70
23		Fat	0.13	0.04	0.00	0.05	14.10
24		Skin	0.22	0.07	0.62	3.00	0.00
25	Central Blood		2.25	n/a	n/a	n/a	n/a

Table 6.2: Constants and parameters used in Stolwijk's model

Table 6.1 and Table 6.2 illustrate some of the constants and parameters used for the model, where N is the number of the node ranging from 1 to 25. For each node N, $C[N]$ is the heat capacity of the node, $QB[N]$ is the basal metabolic heat production of the node, $EB[N]$ is the basal evaporative heat loss including the respiratory heat loss from the trunk core, $BFB[N]$ is the basal effective blood flow of the node and $TC[N]$ is the thermal conductance between two adjacent nodes (N and $N + 1$). The basic principle of the heat exchange for any node can be written as:

$$HF[N] = Heat\ Production(HP) - Heat\ Loss(HL)$$

$$HP = \ Metabolic\ Heat\ Produced$$

$$HL = \ Convection + Conduction + Radiation + Evaporation$$

Where HF represents the heat gained or lost from node N. For different nodes, heat can be gained or lost through the processes of: convective heat exchange with the central blood, conductive heat exchange with adjacent nodes, radiant and evaporative heat exchange with the environment. For core nodes, muscle nodes, fat nodes, skin nodes and central blood node, HF can be calculated respectively using the following formulae:

$$HF[i] = Q[i] - E[i] - BC[i] - TD[i]$$

$$HF[i + 1] = Q[i + 1] - BC[i + 1] + TD[i] - TD[i + 1]$$

$$HF[i + 2] = Q[i + 2] - BC[i + 2] + TD[i + 1] - TD[i + 2]$$

$$HF[i + 3] = Q[i + 3] - BC[i + 3] - E[i + 3] + TD[i + 2] - H[i] \times (T[i + 3] - TAIR)$$

$$HF[25] = \sum_{N=1}^{6} HF[N] + BC[N]$$

Where $H[i] = (HR[i] + HC[i] \times \sqrt{\left(\frac{VAIR}{0.1}\right)}) \times S[i]$

The conductive heat transfer TD from node N to node $N + 1$ can be written as:

$$TD[N] = TC[N] \times (T[N] - T[N + 1])$$

The convective heat transfer BC can be calculated by:

$$BC[N] = BF[N] \times (T[N] - T[25])$$

For any object, the relationship between the heat transferred (ΔHT), the change in temperature(ΔT), and the heat capacity (C) is:

$$\Delta T = \frac{\Delta HT}{C}$$

Therefore, the rate of change in temperature of any node i can be calculated by:

$$\Delta T[N] = \frac{HF[N]}{C[N]}$$

6.4.2. The Controlling System

In Stolwijk's model, the controlling system consists of thermal sensors throughout the body. The brain receives signals from the sensors and sends commands to various effector systems. The difference between reference signals which can be seen as the thresholds for initiating effector responses, and the received signals of actual temperature is regarded as the error signal [271]. If body temperature is increased to a level higher than the reference, positive error is generated. The system will send commands to various organs to promote heat loss through sweating and vasodilation. If the temperature is decreased to a point lower than the reference, negative error is obtained. The system will try to preserve body heat through vasoconstriction and shivering.

6.5. Model Integration and Implementation

To simulate the thermoregulation of human under different environmental conditions and physical activities, the original model proposed by Stolwijk [270] required inputs of air temperature, air velocity, relative humidity in the environment, metabolic rate made by external work, basal metabolic rate and initial body temperatures of all the nodes. Many constants such as the surface area of each body segment and the heat capacitance of each body node are based on a man with a body weight of 74.4 kg and a height of 172cm. To obtain accurate simulation results, these input parameters required by the model need to be set accurately. However, many of these parameters were hard to measure and record at real-time in the past.

Recent developments of modern technologies such as cloud computing, wearable sensor devices and big data have provided a way of continuous, non-intrusive monitoring of our physiological signals, vital signs and environmental conditions. In this section, we describe the proposed approach that integrates different models and takes inputs from various data sources to compute and simulate virtual temperature sensor readings of various parts of the body.

6.5.1. Heart rate based metabolic rate estimation

To estimate the metabolic rate (WORKM) required by Stolwijk's model, we used the equations based on the mixed model proposed by Keytel et al. [272] which were derived from physical exercise experiments by 115 individuals. The equations are converted it into the following form:

$$\mathrm{WORKM} = 60 \times \frac{1}{4.184}((-55.0969 + 0.6309 \times \mathrm{H} + 0.1988 \times \mathrm{W} + 0.2017 \times \mathrm{A}) \times \mathrm{G}$$

$$+ (-20.4022 + 0.4472 \times \mathrm{H} - 0.1263 \times \mathrm{W} + 0.074 \times \mathrm{A}) \times (1 - \mathrm{G}))$$

Where H is the heart rate (beat per minute), W is the body weight in kg, A is the age in years. G is the gender (0 for females and 1 for males), and WORKM is the estimation of the work rate in total heat production in Kcal/hour.

6.5.2. Basal Metabolic Rate Estimation

The Basal Metabolic Rate (BMR) is the rate at which your body uses energy while resting in a neutral temperature environment during the post-absorptive state. In such a state, the energy is sufficient only for keeping your vital organs functioning which include the heart, lungs, liver, intestine, kidneys, sex organs, nervous system, muscles, skin and brain. The Resting Metabolic Rate (RMR) is closely related to BMR, which is usually measured under less restricted conditions. To estimate BMR, we use the equations for predicting RMR proposed by Mifflin et al. [273], derived from data from 498 healthy subjects obtained by indirect calorimetry. The original equations can be converted into the following form:

$$BMR = \frac{1}{24}(9.99 \times W + 6.25 \times HT - 4.92 \times A + 5 \times \mathrm{G} - 161 \times (1 - \mathrm{G}))$$

Where HT stands for the body height of the subject in centimetres, W is the body weight in kg, A is the age in years. G is the gender (0 for females and 1 for males), and BMR is the estimation of basal metabolic rate (kcal/hour).

6.5.3. Personal Sensor data

The growing global popularity of smartphones has resulted in new ways of gathering information through an array of embedded sensors such as cameras, microphones, accelerometers, proximity sensors and GPS. The latest generation of professional wearable health sensors can easily connect to smartphones and track significant physiological parameters. For example, using Wiki-Health Mobile App and a heart rate chest strap such as

Zephyr heart rate monitor [38], heart rate measurements can be easily obtained for estimation of the work rate ($WORKM$). Basic information of the user such as age, weight and height can also be stored in the Wiki-Health platform.

6.5.4. Environmental and Geographic Data

Environmental conditions influence human health and lifestyles in a variety of ways. Environmental data covers a range of factors such as weather conditions, climate change, air quality, food quality, traffic flows, pollutions, pests and parasites, radiations, and natural disasters. Traditionally, many of these environmental conditions were rarely measured or could be only measured by government agencies. With modern advances such as the Internet of Things and social networks, more and more environmental conditions can be measured directly at an individual level and integrated to provide an overall understanding.

Geographic information such as altitude, terrain, and the location of buildings and roads provide extra contextual information for our activities. Many third party databases are now made available which allow users to access different types of environmental and geographic information such as *OpenWeatherMap* and Weather Underground[274-277].

6.5.5. Virtual Sensing Service

All of the aforementioned human physiological models in this thesis have been implemented as functions in MATLAB/Octave which are then uploaded into the Wiki-Health's model repository and published as a virtual sensing service in the data and models marketplace. As shown in Figure 6.3, the proposed virtual sensing service takes physical sensor data, such as heart rate, speed, locations, humidity and wind speed as inputs and computes the simulated readings of virtual sensors of whole body temperatures and metabolic rates.

For users, obtaining readings from the virtual sensing service with Wiki-Health requires almost no knowledge about the computation involved in determining the readings. The user only needs to select the service from the data and models marketplace and configure the mappings of inputs and outputs, such as selecting the heart rate data stream, to be used for the virtual sensing service.

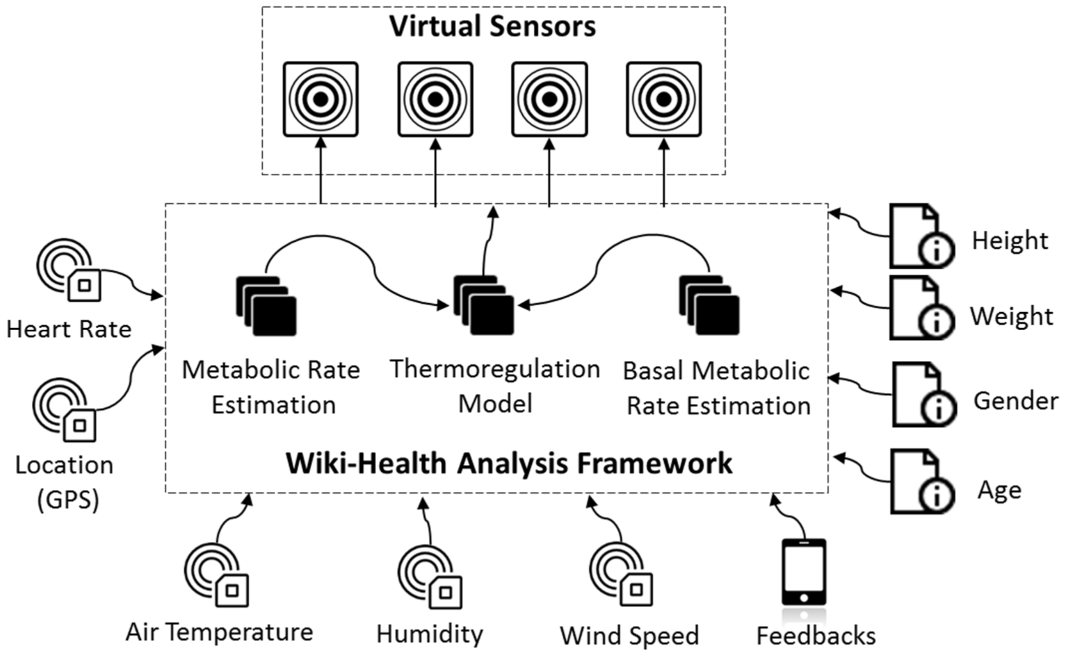


Figure 6.3: Current implementation of virtual sensors formed by integrating different models and various data sources together.

6.6. Experimental Evaluation

The experiments in this section are designed to evaluate the feasibility of the proposed virtual sensing service using a case study.

Participants and Setup

Two subjects participated in the experiments: a 27-year-old male adult (weight 80kg, height 185cm) and a 26-year-old female adult (weight 45kg, height of 163cm). As shown in Figure 6.4, a Zephyr heart rate belt [38] was used to measure the heart rate and instantaneous speed, and it was connected to a smartphone through Bluetooth. The subjects wore both devices during the experiment. A separate infrared thermometer was used to record the environmental temperature and skin temperatures from the subject's right forearm, right lower leg, and forehead approximately every five minutes. The smartphone was installed with the Wiki-Health App to record and upload the physical measurements of heart rate, instantaneous speed, subject's physical location and temperature readings. Humidity and wind speed measurements were obtained from the BBC Weather website[278].

The male subject was asked to perform the following tasks in order: to walk from Imperial College to a nearby park, then either to walk or run in the park freely before walking back to the college and remaining in the office for twenty minutes. Environmental data such as humidity and wind speed were obtained using third-party databases. The outdoor temperature, humidity and wind speed were approximately 27°C, 57% - 61%, and 4.47m/s respectively.

The female subject was asked to perform the following tasks inside an indoor gym: to either run or walk on a cross trainer machine for approximately forty minutes, then to sit on a chair to rest for twenty minutes. The room temperature in the gym and the humidity were stable at about 22.3°C and 50% respectively.

No fluids were consumed during either experiment. The duration of both experiments was approximately one hour.

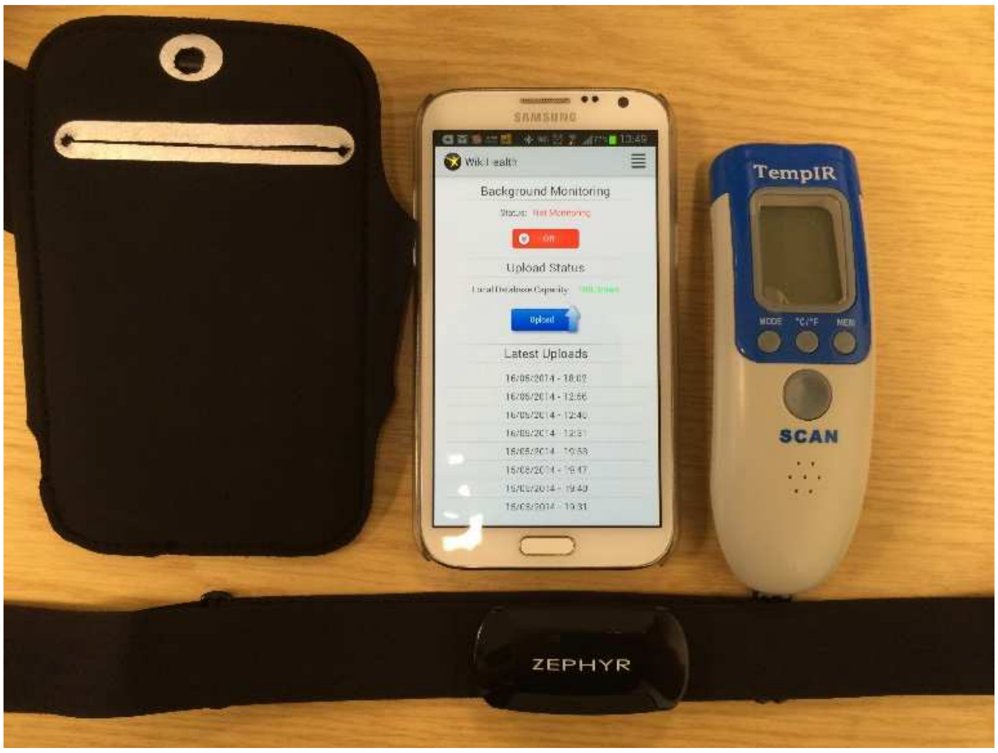


Figure 6.4: Devices used for the experiments (mobile phone armband, a smartphone with Wiki-Health Mobile App installed, infrared thermometer, and heart rate belt.).

Results

Figure 6.5 illustrates the results obtained from the male subject. The top left graph in the figure shows the location change traces of the subject, marked with letters in alphabetical order, while the top right graph shows the heart rate at different times and locations. The next three graphs show the comparison between physical skin temperature measurements and virtual sensor predicted values at different time and locations to evaluate the proposed approach. Similar results obtained from the female subject are depicted in Figure 6.6. Since the subject was in the same location as the male subject throughout the experiment, the location traces were not plotted. The graphs of measured skin-temperatures and predicted virtual sensor readings are also shown.

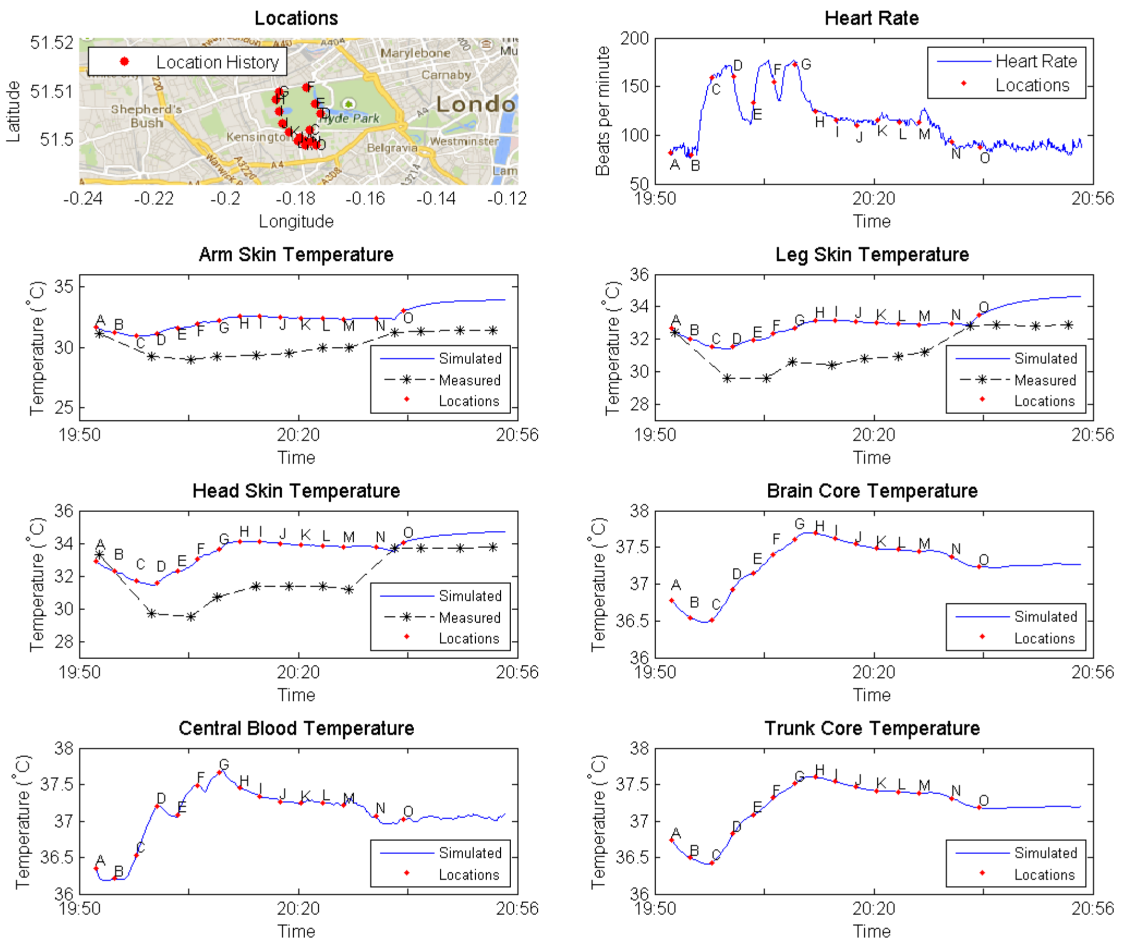


Figure 6.5. Analysis results from the proposed virtual sensing service containing location races, heart rate measurements, and physical and virtual temperature sensor readings of various parts of the body (male subject).

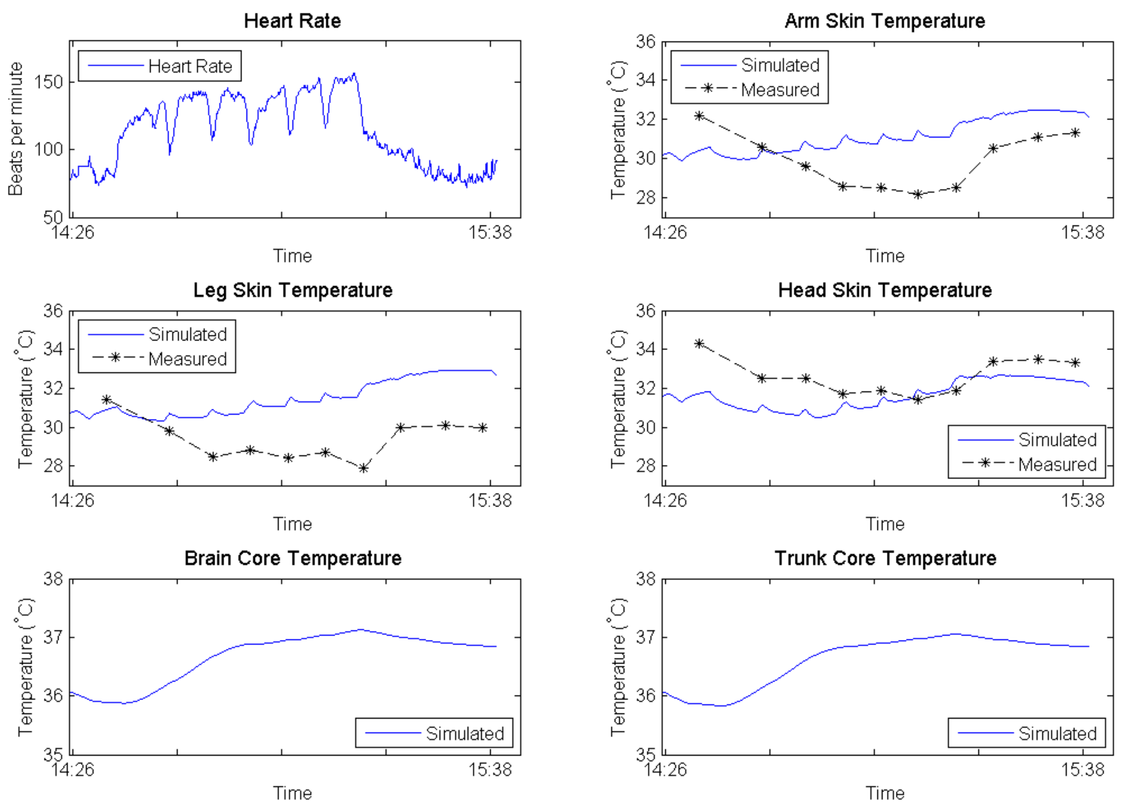


Figure 6.6: Analysis results from the proposed virtual sensing service containing heart rate measurements, and physical and virtual temperature sensor readings of various parts of the body (female subject).

The results show that, for both experiments, the trend of measured skin-temperature changes matches the virtual sensor readings. There are many possible factors that could cause a difference between the two, such as the degree of accuracy of the infrared thermometer and the heart rate monitor, and inaccurate measurements caused by sweating during exercise, as sweat cools skin. Most of the constants and coefficients from Stolwijk's original model were based on a man with body weight of 74.1 kg and surface area of $1.89m^2$. The large variation in the shapes of human bodies and sizes is another factor that could influence the results. The remaining graphs show temperature readings from different locations in the body, such as brain core, central blood, and trunk core simulated by the virtual sensing service.

The case study demonstrates that by using the proposed service, we are able to generate our own models for inferring temperature readings with 25 virtual sensors on various parts of the body. However, validation and evaluation of such models are beyond the scope of this study.

6.7. Summary

We present a virtual sensing application that is deployed as a service on top of Wiki-Health platform. The proposed service adopts the functionalities of Wiki-Health and integrates environmental and geographic data, personal sensor data, and physiological models together to compute and simulate virtual temperature sensor readings of various parts of the body. A comparison was made between physical measurements and simulated readings from the virtual sensors. The proposed service application and performance evaluations demonstrate the feasibility of using Wiki-Health framework to enable better utilisation and understanding of the vast amounts of sensor data from difference sources and shows potential for developing virtual sensors for use in data acquisition on the platform. The approach we have proposed is an example of the use of Wiki-Health, and there may be other human body models in existence that could potentially be used as part of this platform.

In the future work, one direction can be adding another process of data assimilation (e.g. Ensemble Kalman Filter [279]) in the virtual sensing application, so that physical measurements of body temperatures can be incorporated into the virtual sensing model to balance the uncertainty in the data and in the prediction.

7. Conclusions

7.1. Summary

This research has focused on the development of a Wiki-based platform to support healthcare service provision. All the challenges specified in Section 1.2 have been investigated and addressed through the design and development of the CACSS Cloud Storage System and the Wiki-Health platform.

The data and models marketplace within the Wiki-Health platform serves a means to bridge the gap between data owners and data analysis service providers. It also comes with another challenge in terms of evaluating the correctness and accuracy of a model that have been published on the platform. We currently use a voting mechanism that relies on users to contributing feedback to the model. Other approaches, such as comparing different models that attempt to do the same thing with the same input data, i.e. algorithm-evaluation-as-a-service could potentially help verify new models.

The Analysis Tasks Allocation Scheme (ATAS) has been used to address the challenges of real-time analysis on a massive scale by targeting the response time necessary for launching each analysis task (Challenge V in Section 1.2). Within the ATAS, we have assumed that the network bandwidth is large enough to support the data movements involved in the analysis tasks. However, in many situations, the network bandwidth may not always be large enough or elastic enough to support the linear scalability of analysis tasks that require large amounts of data to be moved around. The potential for combine ATAS with a predictive method could help tackle this problem.

The current design of the Wiki-Health web interfaces and Mobile APP were based on the feedback of a small group of users (15 people) who do not have any serious health conditions.

To improve the usability of the interfaces, feedback from a large number of users with different background and health conditions may be needed. Designing more customised and personalised interfaces for different users may also be possible and useful.

Although modern technologies such as smartphones and wireless sensors have the capabilities for tracking a number of physiological parameters, throughout the work described in this thesis, we have discovered several limitations and challenges to adopting these technologies: the performance observed for collecting, storing and uploading the sensor data using different brands and models of smartphones varies significantly. For example, some smartphones can achieve a sampling rate of 100 Hz from their accelerometer sensors, while some can hardly reach 20 Hz. In addition, incorrect use of sensor devices such as electrode misplacement when using ECG monitors, can result in the collection of noisy data which cannot be used in data analysis models or which is otherwise unusable to doctors.

The field of research in data management systems has been evolving rapidly both with respect to the work of the research community and industry since an increasing number of users and companies are shifting into the cloud environments. Giant enterprises such as Amazon and Google have been continuously developing and improving their cloud-based products. Many aspects of the work we have proposed in the thesis are likely to be overtaken by the work of these industry leaders, for example, performance-as-a-service and security-as-a-service. Some aspects of this research such as the hybrid data storage approach, the business model of data and models marketplace, the Analysis Tasks Allocation Scheme, and the Adaptive Learning Approach may remain as a long term contribution of this work.

7.2. Contributions

In conclusion we claim the following contributions resulting from this thesis:

- A big data service platform, named Wiki-Health, is presented to provide a unified solution for collecting, storing, tagging, retrieving, searching and analysing personal health sensor data. To bridge the gaps between the users who collect the data and health professionals who provide data analysis services, Wiki-Health's Analysis Framework enables both parties to effectively utilise all of the collected data while reducing the complexity of dealing with its diversity. The functionalities and associated tools of Wiki-Health also make it feasible for use as a test bed for undertaking health-related experiments and delivering novel analysis algorithms.
- We introduced a hybrid data storage approach that takes advantages of cloud computing, cloud storage, as well as relational and non-relational database features to achieve high performance and efficient data management for heterogeneous health sensor data.
- To address the challenge of managing unstructured data such as medical images, we presented the design, implementation and evaluation of a multi-tier and scalable cloud storage system (CACSS) as a key component of Wiki-Health for managing unstructured and semi-structured data. CACSS has enabled features that are not available in many existing storage systems such as data replications, data versioning, data de-duplications, and performance-as-a-service, as well as user-defined metadata storage and operations. The system also eliminated the limitations on a single file size and the number of files in a directory.
- To demonstrate the feasibility of using Wiki-Health platform to track and discover change-signals that could indicate potential health issues, we presented the design and implementation of a health monitoring model and service on top of the Wiki-Health for personal ECG beat abnormalities detection. The detection mechanism is based on the analysis method of using One-Class Support Vector Machine and the combination of beat blocks and selected features is investigated. The proposed Adaptive Learning Approach (ALA) constructs a cluster of weighted classifiers and makes prediction by

taking the votes from the classifiers. The experimental results show ALA can reduce the training time necessary to update the analysis model and efficiently update the analysis model based on incoming feedback, as well as improve classification rate and specificity while still maintaining high sensitivity.

- Collaborative capabilities of Wiki-Health have been explored through the design and implementation of a virtual sensing application which adopts the functionalities of Wiki-Health and integrates environmental, geographic, and personal sensor data with physiological models to compute and simulate virtual temperature sensor readings of various parts of the body. The proposed virtual sensing application can be useful in many situations such as simulating the performance of athletes and workers under extreme physical conditions and obtaining real-time indirect measurements using other measured variables and triggering alerts. The proposed application also demonstrated the feasibility and potential of using the Wiki-Health for health-related knowledge discovery and movement towards self-understanding.

7.3. Future work

Our work in this thesis focused on investigating the foundations of enabling a unified solution for personal health sensor data management. As a result, avenues for potential research and alternate applications have emerged future work as follows:

- **Large-scale processing of medical imaging data**: Wiki-Health's analysis framework is currently implemented with MATLAB/OCTAVE as the main data processing and analysis environment. In future work, we plan to improve Wiki-Health's capacity of for distributed processing of large-scale medical imaging data distributed across a set of machines with the support of distributed MapReduce framework.
- **Monitoring and assessing the disease progression in Multiple Sclerosis**: the ability to

assess clinically meaningful measures of disability and disease progression MS patients in the “real life” or home environment have shown potential in clinical research involving patients with neurological diseases including multiple sclerosis (MS) [280-283]. In future work, we intend to further evaluate and improve Wiki-Health by adapting it for similar clinical research applications. An example of this is the pre-diagnostic process self-tracking of attacks and symptoms for MS patients with non-invasive wireless sensors.

- **Analysing Physiological Signals in Mental Disorders**: millions of people live with various types of mental health problems such as social anxiety, sleep disorders and personality disorders. Long-term electroencephalograms (EEG) monitoring can aid the detection and diagnosis of many of these mental disorders. Recent studies [284, 285] have also shown that ECG signals measured from the heart can be used to classify emotion states. The approach of combining the two types of time-series signals by data fusion from different levels can potentially help understand mental disorders and predict early symptoms. For future work, one direction is to investigate and validate such approach with Wiki-Health platform and evaluate the performance.

Bibliography

[1] B. Feldman, E. M. Martin, and T. Skotnes, "Big Data in Healthcare Hype and Hope," *October 2012. Dr. Bonnie,* vol. 360, 2012.

[2] S. R. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong, "TinyDB: an acquisitional query processing system for sensor networks," *ACM Transactions on database systems (TODS),* vol. 30, pp. 122-173, 2005.

[3] D. J. Abadi, D. Carney, U. Çetintemel, M. Cherniack, C. Convey, S. Lee*, et al.*, "Aurora: a new model and architecture for data stream management," *The VLDB Journal—The International Journal on Very Large Data Bases,* vol. 12, pp. 120-139, 2003.

[4] Y. Yao and J. Gehrke, "The cougar approach to in-network query processing in sensor networks," *ACM Sigmod Record,* vol. 31, pp. 9-18, 2002.

[5] Amazon. *Amazon Simple Storage Service (S3)*. Available: http://aws.amazon.com/s3 [Accessed 2015]

[6] Google. *Google Cloud Storage Service*. Available: http://code.google.com/apis/storage [Accessed July 2015]

[7] B. aDam LeVenthaL, "Flash storage memory," *Communications of the ACM,* vol. 51, 2008.

[8] Y. Kim, S. Gurumurthi, and A. Sivasubramaniam, "Understanding the performance-temperature interactions in disk I/O of server workloads," in *High-Performance Computer Architecture, 2006. The Twelfth International Symposium on*, 2006, pp. 176-186.

[9] *ThingSpeak*. Available: http://thingspeak.com [Accessed 2015]

[10] *Xively*. Available: https://xively.com [Accessed July 2015]

[11] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig, "Small-file access in parallel file systems," in *Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on*, 2009, pp. 1-11.

[12] M. Larobina and L. Murino, "Medical Image File Formats," *Journal of digital*

imaging, vol. 27, pp. 200-206, 2014.

[13] O. S. Pianykh, *Digital imaging and communications in medicine (DICOM)*: Springer, 2012.

[14] R. Robb, D. Hanson, R. Karwoski, A. Larson, E. Workman, and M. Stacy, "Analyze: a comprehensive, operator-interactive software package for multidimensional medical image display and analysis," *Computerized Medical Imaging and Graphics,* vol. 13, pp. 433-454, 1989.

[15] *MINC software library and tools.* Available: http://www.bic.mni.mcgill.ca/ServicesSoftware/MINC [Accessed July 2015]

[16] *NIFTI documentation.* Available: http://nifti.nimh.nih.gov/

[17] P. Groves, B. Kayyali, D. Knott, and S. Van Kuiken, "The 'big data'revolution in healthcare," *McKinsey Quarterly,* 2013.

[18] "Drowning in Big Data? Reducing Information Technology Complexities and Costs For Healthcare Organizations," Frost and Sullivan.

[19] "Medical Image Sharing And Management Drives Collaborative Care: Overcoming Fragmentation To Create Unity," Frost and Sullivan.

[20] *NHS Annual Imaging and Radiodiagnostics Data.* Available: http://www.england.nhs.uk/statistics/statistical-work-areas/diagnostics-waiting-times-and-activity/imaging-and-radiodiagnostics-annual-data/

[21] B. Feldman, E. M. Martin, and T. Skotnes, "Big Data in Healthcare Hype and Hope," 2012.

[22] *AliveCor heart monitor.* Available: http://www.alivecor.com [Accessed July 2015]

[23] C. A. Volgman, S. Wang, D. Mehta, N. Nazir, S. Alexander, K. Krishnan*, et al.*, "O016 AliveCor Heart Monitoring: Is it a practical alternative to a traditional ECG monitor for a developing nation?," *Global Heart,* vol. 9, pp. e4-e5, 2014.

[24] H. Laufs, A. Kleinschmidt, A. Beyerle, E. Eger, A. Salek-Haddadi, C. Preibisch*, et al.*, "EEG-correlated fMRI of human alpha activity," *Neuroimage,* vol. 19, pp. 1463-1476, 2003.

[25] S. Debener, M. Ullsperger, M. Siegel, and A. K. Engel, "Single-trial EEG–fMRI

reveals the dynamics of cognitive function," *Trends in cognitive sciences,* vol. 10, pp. 558-563, 2006.

[26] A. Salek-Haddadi, B. Diehl, K. Hamandi, M. Merschhemke, A. Liston, K. Friston*, et al.*, "Hemodynamic correlates of epileptiform discharges: an EEG-fMRI study of 63 patients with focal epilepsy," *Brain research,* vol. 1088, pp. 148-166, 2006.

[27] Y. Li, C. Wu, L. Guo, C.-H. Lee, and Y. Guo, "Wiki-Health: A Big Data Platform for Health Sensor Data Management," in *Cloud Computing Applications for Quality Health Care Delivery*

M. Anastasius and N. K. Anastasia, Eds., ed Hershey, PA, USA: IGI Global

pp. 59-77

[28] H. Gevensleben, B. Holl, B. Albrecht, D. Schlamp, O. Kratz, P. Studer*, et al.*, "Distinct EEG effects related to neurofeedback training in children with ADHD: a randomized controlled trial," *International Journal of Psychophysiology,* vol. 74, pp. 149-157, 2009.

[29] I. Pušnik and A. Miklavec, "Dilemmas in measurement of human body temperature," *Instrumentation Science and Technology,* vol. 37, pp. 516-530, 2009.

[30] G. Dean, "Maintaining Body Temperature," *Nursing Care and the Activities of Living,* p. 210, 2010.

[31] L. McCallum and D. Higgins, "Measuring body temperature," *Nursing times,* vol. 108, pp. 20-22, 2011.

[32] D. W. Hensley, A. E. Mark, J. R. Abella, G. M. Netscher, E. H. Wissler, and K. R. Diller, "50 Years of Computer Simulation of the Human Thermoregulatory System," *Journal of biomechanical engineering,* vol. 135, p. 021006, 2013.

[33] D. Fiala, K. J. Lomas, and M. Stohrer, "Computer prediction of human thermoregulatory and temperature responses to a wide range of environmental conditions," *International Journal of Biometeorology,* vol. 45, pp. 143-159, 2001.

[34] O. Kimberger, R. Thell, M. Schuh, J. Koch, D. Sessler, and A. Kurz, "Accuracy and precision of a novel non-invasive core thermometer," *British journal of anaesthesia,*

p. aep134, 2009.

[35] A. Fonte, F. Alimenti, D. Zito, B. Neri, D. De Rossi, A. Lanata*, et al.*, "Wearable system-on-a-chip radiometer for remote temperature sensing and its application to the safeguard of emergency operators," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, 2007, pp. 5715-5718.

[36] C. Byrne and C. L. Lim, "The ingestible telemetric body core temperature sensor: a review of validity and exercise applications," *British journal of sports medicine,* vol. 41, pp. 126-133, 2007.

[37] *Fitbit*. Available: http://www.fitbit.com [Accessed July 2015]

[38] *Zephyr*. Available: http://www.zephyranywhere.com [Accessed July 2015]

[39] *NIKE+ FUELBAND*. Available: http://www.nike.com [Accessed July 2015]

[40] *Withings*. Available: www.withings.com [Accessed July 2015]

[41] *iRhythm patch*. Available: http://www.irhythmtech.com/ [Accessed July 2015]

[42] *CardioNet*. Available: http://www.cardionet.com [Accessed July 2015]

[43] *Neurosky Headset*. Available: http://www.neurosky.com/ [Accessed July 2015]

[44] *Emotiv Headsets*. Available: http://www.emotiv.com/ [Accessed July 2015]

[45] J. C. Venter, M. D. Adams, E. W. Myers, P. W. Li, R. J. Mural, G. G. Sutton*, et al.*, "The sequence of the human genome," *science,* vol. 291, pp. 1304-1351, 2001.

[46] E. S. Lander, L. M. Linton, B. Birren, C. Nusbaum, M. C. Zody, J. Baldwin*, et al.*, "Initial sequencing and analysis of the human genome," *Nature,* vol. 409, pp. 860-921, 2001.

[47] G. P. Consortium, "A map of human genome variation from population-scale sequencing," *Nature,* vol. 467, pp. 1061-1073, 2010.

[48] S. B. Ng, K. J. Buckingham, C. Lee, A. W. Bigham, H. K. Tabor, K. M. Dent*, et al.*, "Exome sequencing identifies the cause of a mendelian disorder," *Nature genetics,* vol. 42, pp. 30-35, 2010.

[49] J. R. Lupski, J. G. Reid, C. Gonzaga-Jauregui, D. Rio Deiros, D. C. Chen, L. Nazareth*,*

et al., "Whole-genome sequencing in a patient with Charcot–Marie–Tooth neuropathy," *New England Journal of Medicine,* vol. 362, pp. 1181-1191, 2010.

[50] E. A. Ashley, A. J. Butte, M. T. Wheeler, R. Chen, T. E. Klein, F. E. Dewey*, et al.*, "Clinical assessment incorporating a personal genome," *The Lancet,* vol. 375, pp. 1525-1535, 2010.

[51] N. Siva, "1000 Genomes project," *Nature biotechnology,* vol. 26, pp. 256-256, 2008.

[52] L. D. Stein, "The case for cloud computing in genome informatics," *Genome Biol,* vol. 11, p. 207, 2010.

[53] L. Clarke, X. Zheng-Bradley, R. Smith, E. Kulesha, C. Xiao, I. Toneva*, et al.*, "The 1000 Genomes Project: data management and community access," *Nature methods,* vol. 9, pp. 459-462, 2012.

[54] G. Wolf, A. Carmichael, and K. Kelly, "The quantified self," *TED http://www. ted. com/talks/gary_wolf_the_quantified_self. html,* 2010.

[55] M. Swan, "The quantified self: Fundamental disruption in big data science and biological discovery," *Big Data,* vol. 1, pp. 85-99, 2013.

[56] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "A survey on sensor networks," *Communications magazine, IEEE,* vol. 40, pp. 102-114, 2002.

[57] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer networks,* vol. 38, pp. 393-422, 2002.

[58] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on,* vol. 40, pp. 1-12, 2010.

[59] N. Xu, "A survey of sensor network applications," *IEEE Communications Magazine,* vol. 40, pp. 102-114, 2002.

[60] S. AlSairafi, F.-S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos*, et al.*, "The design of discovery net: Towards open grid services for knowledge discovery," *International Journal of High Performance Computing Applications,* vol. 17, pp. 297-315, 2003.

[61] R. N. Murty, G. Mainland, I. Rose, A. R. Chowdhury, A. Gosain, J. Bers*, et al.*,

"Citysense: An urban-scale wireless sensor network and testbed," in *Technologies for Homeland Security, 2008 IEEE Conference on*, 2008, pp. 583-588.

[62] D. S. Roselli, J. R. Lorch, and T. E. Anderson, "A Comparison of File System Workloads," in *USENIX Annual Technical Conference, General Track*, 2000, pp. 41-54.

[63] G. A. Gibson and R. Van Meter, "Network attached storage architecture," *Communications of the ACM,* vol. 43, pp. 37-45, 2000.

[64] P. Schwan, "Lustre: Building a file system for 1000-node clusters," presented at the Proceedings of the Linux Symposium, 2003.

[65] W. Yu, R. Noronha, S. Liang, and D. K. Panda, "Benefits of high speed interconnects to cluster file systems: a case study with lustre," in *Parallel and Distributed Processing Symposium, 2006. IPDPS 2006. 20th International*, 2006, p. 8 pp.

[66] S. Ghemawat, H. Gobioff, and S.-T. Leung, "The Google file system," in *ACM SIGOPS Operating Systems Review*, 2003, pp. 29-43.

[67] D. Borthakur, "The hadoop distributed file system: Architecture and design," *Hadoop Project Website,* 2007.

[68] J. Zawodny, "Title," unpublished|.

[69] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows*, et al.*, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS),* vol. 26, p. 4, 2008.

[70] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *SIGOPS Oper. Syst. Rev.,* vol. 44, pp. 35-40, 2010.

[71] A. HBase. Available: http://hbase.apache.org [Accessed July 2015]

[72] L. George, *HBase: the definitive guide*: " O'Reilly Media, Inc.", 2011.

[73] *Twitter*. Available: https://twitter.com [Accessed July 2015]

[74] *MongoDB*. Available: http://www.mongodb.org

[75] *Redis* Available: http://redis.io/ [Accessed July 2015]

[76] *Wikipedia*. Available: http://www.wikipedia.org/ [Accessed July 2015]

[77] B. Mons, M. Ashburner, C. Chichester, E. van Mulligen, M. Weeber, J. den Dunnen, *et al.*, "Calling on a million minds for community annotation in WikiProteins," *Genome biology,* vol. 9, p. R89, 2008.

[78] A. R. Pico, T. Kelder, M. P. Van Iersel, K. Hanspers, B. R. Conklin, and C. Evelo, "WikiPathways: pathway editing for the people," *PLoS biology,* vol. 6, p. e184, 2008.

[79] R. Hoffmann, "A wiki for the life sciences where authorship matters," *Nature genetics,* vol. 40, pp. 1047-1051, 2008.

[80] Y. Wang, W. Hu, Y. Wu, and G. Cao, "SmartPhoto: A Resource-Aware Crowdsourcing Approach for Image Sensing with Smartphones," in *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*, 2014, pp. 113-122.

[81] X. Guo, E. C. Chan, C. Liu, K. Wu, S. Liu, and L. M. Ni, "ShopProfiler: Profiling shops with crowdsourcing data," in *INFOCOM, 2014 Proceedings IEEE*, 2014, pp. 1240-1248.

[82] A. Artikis, M. Weidlich, F. Schnitzler, I. Boutsis, T. Liebig, N. Piatkowski*, et al.*, "Heterogeneous Stream Processing and Crowdsourcing for Urban Traffic Management," in *EDBT*, 2014, pp. 712-723.

[83] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 791-800.

[84] "Social media "likes" healthcare: From marketing to social business," PwC Health Research Institute.

[85] *Sarasota Memorial Hospital.* Available: https://twitter.com/SMHCS [Accessed July 2015]

[86] *Rush University Medical Center.* Available: https://twitter.com/RushMedical [Accessed July 2015]

https://www.facebook.com/RushUniversityMedicalCenter [Accessed July 2015]

[87] *Detroit's Henry Ford Hospital.* Available: http://www.henryford.com,

https://twitter.com/thehenryford [Accessed July 2015]

[88] *Doximity*. Available: http://www.doximity.com/ [Accessed July 2015]

[89] *Sermo*. Available: http://www.sermo.com/ [Accessed July 2015]

[90] (2013). *AWS Case Study: SmugMug*. Available: http://aws.amazon.com/solutions/case-studies/smugmug/ [Accessed July 2015]

[91] S. Gurumurthi, A. Sivasubramaniam, and V. K. Natarajan, *Disk drive roadmap from the thermal perspective: A case for dynamic thermal management* vol. 33: IEEE Computer Society, 2005.

[92] S. H. Charap, P.-L. Lu, and Y. He, "Thermal stability of recorded information at high densities," *Magnetics, IEEE Transactions on,* vol. 33, pp. 978-983, 1997.

[93] Amazon. (2012). *Amazon S3 - The First Trillion Objects*. Available: http://aws.typepad.com/aws/2012/06/amazon-s3-the-first-trillion-objects.html [Accessed July 2015]

[94] N. Gohring. *Amazon's S3 Down for Several Hours*. Available: http://www.pcworld.com/article/142549/article.html [Accessed July 2015]

[95] J. Brodkin. (2008). *Outage hits Amazon S3 storage service*. Available: http://www.networkworld.com/news/2008/021508-amazon.html [Accessed July 2015]

[96] G. Wang and T. E. Ng, "The impact of virtualization on network performance of amazon ec2 data center," in *INFOCOM, 2010 Proceedings IEEE*, 2010, pp. 1-9.

[97] T. Kind, "RAMDISK Benchmarks," *University of California,* 2011.

[98] Rackspace. *Cloud Files*. Available: http://www.rackspace.co.uk [Accessed July 2015]

[99] J. Barr. (2011). Available: http://aws.typepad.com/aws/2011/10/amazon-s3-566-billion-objects-370000-requestssecond-and-hiring.html [Accessed July 2015]

[100] *Amazon S3 - Two Trillion Objects, 1.1 Million Requests / Second.* Available: http://aws.amazon.com/blogs/aws/amazon-s3-two-trillion-objects-11-million-requests-second/ [Accessed July 2015]

[101] G. Infrastructure. (2009). *GoGrid Cloud Hosting*. Available: http://www.gogrid.com [Accessed July 2015]

[102] M. Yuriyama and T. Kushida, "Sensor-cloud infrastructure-physical sensor management with virtualized sensors on cloud computing," in *Network-Based Information Systems (NBiS), 2010 13th International Conference on*, 2010, pp. 1-8.

[103] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Commun. ACM,* vol. 51, pp. 107-113, 2008.

[104] Apache. *Hadoop MapReduce*. Available: http://hadoop.apache.org/mapreduce/ [Accessed July 2015]

[105] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears, "MapReduce Online," in *NSDI*, 2010, p. 20.

[106] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: a survey," *AcM sIGMoD Record,* vol. 40, pp. 11-20, 2012.

[107] Y. Bu, B. Howe, M. Balazinska, and M. D. Ernst, "HaLoop: Efficient iterative data processing on large clusters," *Proceedings of the VLDB Endowment,* vol. 3, pp. 285-296, 2010.

[108] J. Zhang, D. Xiang, T. Li, and Y. Pan, "M2M: A simple Matlab-to-MapReduce translator for cloud computing," *Tsinghua Science and Technology,* vol. 18, pp. 1-9, 2013.

[109] I. MathWorks, *MATLAB: the language of technical computing. Desktop tools and development environment, version 7* vol. 9: MathWorks, 2005.

[110] *Octave*. Available: http://www.gnu.org/software/octave/ [Accessed July 2015]

[111] R. Gentleman, *R programming for bioinformatics*: CRC Press, 2008.

[112] R. A. Becker, J. M. Chambers, and A. R. Wilks, "The new S language," *Pacific Grove, Ca.: Wadsworth & Brooks, 1988,* vol. 1, 1988.

[113] J. M. Chambers and T. J. Hastie, *Statistical models in S*: CRC Press, Inc., 1991.

[114] R. A. Becker and J. M. Chambers, *S: an interactive environment for data analysis and graphics*: CRC Press, 1984.

[115] *The Julia Language*. Available: http://julialang.org/ [Accessed July 2015]

[116] R. Luengo-Fernandez, J. Leal, A. Gray, S. Petersen, and M. Rayner, "Cost of cardiovascular diseases in the United Kingdom," *Heart,* vol. 92, pp. 1384-1389, 2006.

[117] J. J. Oresko, Z. Jin, J. Cheng, S. Huang, Y. Sun, H. Duschl*, et al.*, "A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 14, pp. 734-740, 2010.

[118] E. Nemati, M. J. Deen, and T. Mondal, "A wireless wearable ECG sensor for long-term applications," *Communications Magazine, IEEE,* vol. 50, pp. 36-43, 2012.

[119] *Propeller Health*. Available: http://propellerhealth.com/ [Accessed July 2015]

[120] E. J. Topol, *The creative destruction of medicine: How the digital revolution will create better health care*: Basic Books, 2012.

[121] M. Bsoul, H. Minn, and L. Tamil, "Apnea MedAssist: real-time sleep apnea monitor using single-lead ECG," *Information Technology in Biomedicine, IEEE Transactions on,* vol. 15, pp. 416-427, 2011.

[122] T. Q. Le, C. Cheng, A. Sangasoongsong, and S. T. Bukkapatnam, "Prediction of sleep apnea episodes from a wireless wearable multisensor suite," in *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE*, 2013, pp. 152-155.

[123] S. Adibi, "Wireless-Based Sleep Technology for Monitoring Sleep Apnea," *Advanced Science, Engineering and Medicine,* vol. 6, pp. 111-113, 2014.

[124] R. A. Barkley, "Behavioral inhibition, sustained attention, and executive functions: constructing a unifying theory of ADHD," *Psychological bulletin,* vol. 121, p. 65, 1997.

[125] G. Weiss and L. T. Hechtman, *Hyperactive children grown up: ADHD in children, adolescents, and adults*: Guilford Press, 1993.

[126] R. C. Kessler, L. Adler, R. Barkley, J. Biederman, C. K. Conners, O. Demler*, et al.*, "The prevalence and correlates of adult ADHD in the United States: results from the National Comorbidity Survey Replication," *The American journal of psychiatry,* vol. 163, p. 716, 2006.

[127] R. A. Barkley, M. B. McMurray, C. S. Edelbrock, and K. Robbins, "Side effects of metlyiphenidate in children with attention deficit hyperactivity disorder: a systemic, placebo-controlled evaluation," *Pediatrics,* vol. 86, pp. 184-192, 1990.

[128] D. Efron, F. Jarman, and M. Barker, "Side effects of methylphenidate and dexamphetamine in children with attention deficit hyperactivity disorder: a double-blind, crossover trial," *Pediatrics,* vol. 100, pp. 662-666, 1997.

[129] L. Yang, W. Chao, G. Li, L. Chun-Hsiang, and G. Yike, "Wiki-Health: A Big Data Platform for Health Sensor Data Management," in *Cloud Computing Applications for Quality Health Care Delivery*, M. Anastasius and N. K. Anastasia, Eds., ed Hershey, PA, USA: IGI Global, 2014, pp. 59-77.

[130] J. F. Lubar, M. O. Swartwood, J. N. Swartwood, and P. H. O'Donnell, "Evaluation of the effectiveness of EEG neurofeedback training for ADHD in a clinical setting as measured by changes in TOVA scores, behavioral ratings, and WISC-R performance," *Biofeedback and Self-regulation,* vol. 20, pp. 83-99, 1995.

[131] M. Arns, S. de Ridder, U. Strehl, M. Breteler, and A. Coenen, "Efficacy of neurofeedback treatment in ADHD: the effects on inattention, impulsivity and hyperactivity: a meta-analysis," *Clinical EEG and neuroscience,* vol. 40, pp. 180-189, 2009.

[132] Q. Wang, O. Sourina, and M. K. Nguyen, "Eeg-based" serious" games design for medical applications," in *Cyberworlds (CW), 2010 International Conference on*, 2010, pp. 270-276.

[133] M. Lansbergen, M. van Dongen-Boomsma, J. Buitelaar, and D. Slaats-Willemse, "ADHD and EEG-neurofeedback: a double-blind randomized placebo-controlled feasibility study," *Journal of Neural Transmission,* vol. 118, pp. 275-284, 2011.

[134] V. Mihajlovic, B. Grundlehner, R. Vullers, and J. Penders, "Wearable, Wireless EEG Solutions in Daily Life Applications: What are we missing?," 2014.

[135] I. S. Mackenzie, S. V. Morant, G. A. Bloomfield, T. M. MacDonald, and J. I. O'Riordan, "Changing face of multiple sclerosis in the United Kingdom 1990–2010. An incidence and prevalence study," *Journal of Neurology, Neurosurgery & Psychiatry,* vol. 84, pp. e2-e2, 2013.

[136] D. Gijbels, U. Dalgas, A. Romberg, V. de Groot, F. Bethoux, C. Vaney*, et al.*, "Which

walking capacity tests to use in multiple sclerosis? A multicentre study providing the basis for a core set," *Multiple Sclerosis Journal,* vol. 18, pp. 364-371, 2012.

[137] B. H. Dobkin and A. Dorsch, "The promise of mHealth daily activity monitoring and outcome assessments by wearable sensors," *Neurorehabilitation and Neural Repair,* vol. 25, pp. 788-798, 2011.

[138] L. Shammas, T. Zentek, B. von Haaren, S. Schlesinger, S. Hey, and A. Rashid, "Home-based system for physical activity monitoring in patients with multiple sclerosis (Pilot study)," *Biomedical engineering online,* vol. 13, p. 10, 2014.

[139] T. Square and B. Lane. (2013). *Enhanced Service-Facilitating timely diagnosis and support for people with dementia.* Available: http://www.hscic.gov.uk/article/3489/IAG-documents [Accessed July 2015]

[140] *Google Glass*. Available: http://www.google.com/glass [Accessed Jan 2014]

[141] *Narrative Clip*. Available: http://getnarrative.com/ [Accessed July 2015]

[142] *Autographer Camera*. Available: http://www.autographer.com/ [Accessed July 2015]

[143] B. Kikhia, J. Hallberg, K. Synnes, and Z. Sani, "Context-aware life-logging for persons with mild dementia," in *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, 2009, pp. 6183-6186.

[144] P. Wang and A. F. Smeaton, "Using visual lifelogs to automatically characterize everyday activities," *Information Sciences,* vol. 230, pp. 147-161, 5/1/ 2013.

[145] J. Hallberg, B. Kikhia, J. Bengtsson, S. Sävenstedt, and K. Synnes, "Reminiscence Processes Using Life-Log Entities for Persons with Mild Dementia," in *Proceedings of the first international workshop on reminiscence systems*, 2009, pp. 16-21.

[146] R. Cracknell, "The ageing population," House of Commons Library research2010.

[147] P. Scuffham, S. Chaplin, and R. Legood, "Incidence and costs of unintentional falls in older people in the United Kingdom," *Journal of epidemiology and community health,* vol. 57, pp. 740-744, 2003.

[148] S. Borland, "More than 500,000 elderly sent to A&E every year because of failures in care: Needless admissions among older people up 40% in five years," ed, 2013.

[149] A. Arcelus, M. H. Jones, R. Goubran, and F. Knoefel, "Integration of smart home

technologies in a health monitoring system for the elderly," in *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference on*, 2007, pp. 820-825.

[150] F. Portet, M. Vacher, C. Golanski, C. Roux, and B. Meillon, "Design and evaluation of a smart home voice interface for the elderly: acceptability and objection aspects," *Personal and Ubiquitous Computing,* vol. 17, pp. 127-144, 2013.

[151] J.-V. Lee, Y.-D. Chuah, and K. T. Chieng, "Smart Elderly Home Monitoring System with an Android Phone," *Int. J. Smart Home,* vol. 7, pp. 17-32, 2013.

[152] Y. Charlon, W. Bourennane, F. Bettahar, and E. Campo, "Activity monitoring system for elderly in a context of smart home," *IRBM,* vol. 34, pp. 60-63, 2013.

[153] N. Ogawa, Y. Imai, H. Morita, and R. Nagai, "Genome-wide association study of coronary artery disease," *International journal of hypertension,* vol. 2010, 2010.

[154] A. Helgadottir, G. Thorleifsson, A. Manolescu, S. Gretarsdottir, T. Blondal, A. Jonasdottir*, et al.*, "A common variant on chromosome 9p21 affects the risk of myocardial infarction," *Science,* vol. 316, pp. 1491-1493, 2007.

[155] R. McPherson, A. Pertsemlidis, N. Kavaslar, A. Stewart, R. Roberts, D. R. Cox*, et al.*, "A common allele on chromosome 9 associated with coronary heart disease," *Science,* vol. 316, pp. 1488-1491, 2007.

[156] D. F. Gudbjartsson, U. S. Bjornsdottir, E. Halapi, A. Helgadottir, P. Sulem, G. M. Jonsdottir*, et al.*, "Sequence variants affecting eosinophil numbers associate with asthma and myocardial infarction," *Nature genetics,* vol. 41, pp. 342-347, 2009.

[157] J. Erdmann, A. Großhennig, P. S. Braund, I. R. König, C. Hengstenberg, A. S. Hall*, et al.*, "New susceptibility locus for coronary artery disease on chromosome 3q22. 3," *Nature genetics,* vol. 41, pp. 280-282, 2009.

[158] A. E. Guttmacher, A. L. McGuire, B. Ponder, and K. Stefánsson, "Personalized genomic information: preparing for the future of genetic medicine," *Nature Reviews Genetics,* vol. 11, pp. 161-165, 2010.

[159] E. Stern, A. Vacic, N. K. Rajan, J. M. Criscione, J. Park, B. R. Ilic*, et al.*, "Label-free biomarker detection from whole blood," *Nature nanotechnology,* vol. 5, pp. 138-142, 2009.

[160] B. Godin, J. H. Sakamoto, R. E. Serda, A. Grattoni, A. Bouamrani, and M. Ferrari, "Emerging applications of nanomedicine for the diagnosis and treatment of cardiovascular diseases," *Trends in pharmacological sciences,* vol. 31, pp. 199-205, 2010.

[161] R. Hasan, M. Winslett, and R. Sion, "Requirements of secure storage systems for healthcare records," in *Secure Data Management*, ed: Springer, 2007, pp. 174-180.

[162] C. f. D. Control and Prevention, "HIPAA privacy rule and public health. Guidance from CDC and the US Department of Health and Human Services," *MMWR. Morbidity and mortality weekly report,* vol. 52, p. 1, 2003.

[163] U. Congress, "American recovery and reinvestment act of 2009," *Public Law,* p. 111, 2009.

[164] S. R. Salbu, "European Union Data Privacy Directive and International Relations, The," *Vand. J. Transnat'l L.,* vol. 35, p. 655, 2002.

[165] P. Carey, *Data protection: a practical guide to UK and EU law*: Oxford University Press, Inc., 2009.

[166] Amazon. *Amazon Elastic MapReduce*. Available: http://aws.amazon.com/elasticmapreduce/ [Accessed July 2015]

[167] G. Singh, S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar*, et al.*, "A metadata catalog service for data intensive applications," 2003, pp. 33-33.

[168] A. W. Leung, M. Shao, T. Bisson, S. Pasupathy, and E. L. Miller, "Spyglass: fast, scalable metadata search for large-scale storage systems," presented at the Proccedings of the 7th conference on File and storage technologies, San Francisco, California, 2009.

[169] P. Carns, S. Lang, R. Ross, M. Vilayannur, J. Kunkel, and T. Ludwig, "Small-file access in parallel file systems," 2009, pp. 1-11.

[170] J. F. Gantz and C. Chute, "The diverse and exploding digital universe: An updated forecast of worldwide information growth through 2011," 2008.

[171] D. T. Meyer and W. J. Bolosky, "A study of practical deduplication," *ACM Transactions on Storage (TOS),* vol. 7, p. 14, 2012.

[172] L. DuBois, M. Amaldas, and E. Sheppard, "Key considerations as deduplication evolves into primary storage," *White Paper,* vol. 223310, 2011.

[173] Y.-K. Guo and L. Guo, "IC cloud: Enabling compositional cloud," *International Journal of Automation and Computing,* vol. 8, pp. 269-279, 2011.

[174] L. Guo, Y. Guo, and X. Tian, "IC cloud: a design space for composable cloud computing," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 394-401.

[175] P. Saab, "Scaling memcached at Facebook," *Facebook Engineering Note,* 2008.

[176] C. Do, "Youtube scalability," presented at the Youtube/Google, Inc, 2007.

[177] M. Bergsma, "Wikimedia architecture," *Wikimedia Foundation Inc,* 2007.

[178] R. Admins, "reddit's May 2010," *State of the Servers" report," Reddit. com,* vol. 11, 2011.

[179] B. Fitzpatrick, "Distributed caching with memcached," *Linux J.,* vol. 2004, p. 5, 2004.

[180] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff*, et al.*, "The eucalyptus open-source cloud-computing system," 2009, pp. 124-131.

[181] Openstack. Available: http://openstack.org [Accessed July 2015]

[182] Y. Abe and G. Gibson, "pWalrus: Towards better integration of parallel file systems into cloud storage," 2010, pp. 1-7.

[183] J. Bresnahan, K. Keahey, T. Freeman, and D. LaBissoniere, "Cumulus: an open source storage cloud for science," *SC10 Poster,* 2010.

[184] *Objectivity/DB*. Available: http://www.objectivity.com [Accessed 09 July 2015]

[185] *VelocityDB*. Available: http://www.velocitydb.com [Accessed 09 July 2015]

[186] *EyeDB*. Available: http://www.eyedb.org [Accessed 09 July 2015]

[187] *ObjectStore*. Available: http://www.objectstore.com [Accessed 09 July 2015]

[188] Y. Tianming, J. Hong, F. Dan, N. Zhongying, Z. Ke, and W. Yaping, "DEBAR: A scalable high-performance de-duplication storage system for backup and archiving," in *Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium*

on, 2010, pp. 1-12.

[189] T. Yujuan, J. Hong, F. Dan, T. Lei, Y. Zhichao, and Z. Guohui, "SAM: A Semantic-Aware Multi-tiered Source De-duplication Framework for Cloud Backup," in *Parallel Processing (ICPP), 2010 39th International Conference on*, 2010, pp. 614-623.

[190] L. Chuanyi, L. Yingping, S. Chunhui, L. Guanlin, D. H. C. Du, and W. Dong-sheng, "ADMAD: Application-Driven Metadata Aware De-duplication Archival Storage System," in *Storage Network Architecture and Parallel I/Os, 2008. SNAPI '08. Fifth IEEE International Workshop on*, 2008, pp. 29-35.

[191] S. Quinlan and S. Dorward, "Venti: a new approach to archival storage," in *Proceedings of the FAST 2002 Conference on File and Storage Technologies*, 2002.

[192] L. L. You, K. T. Pollack, and D. D. Long, "Deep Store: An archival storage system architecture," in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, 2005, pp. 804-815.

[193] C. Dubnicki, L. Gryz, L. Heldt, M. Kaczmarczyk, W. Kilian, P. Strzelczak, *et al.*, "Hydrastor: A scalable secondary storage," in *Proccedings of the 7th conference on File and storage technologies*, 2009, pp. 197-210.

[194] W. Jiansheng, J. Hong, Z. Ke, and F. Dan, "MAD2: A scalable high-throughput exact deduplication approach for network backup services," in *Mass Storage Systems and Technologies (MSST), 2010 IEEE 26th Symposium on*, 2010, pp. 1-14.

[195] B. Schwartz, P. Zaitsev, and V. Tkachenko, *High Performance MySQL: Optimization, Backups, and Replication*: O'Reilly Media, Inc., 2012.

[196] D. Applegate, A. Archer, V. Gopalakrishnan, S. Lee, and K. Ramakrishnan, "Optimal content placement for a large-scale VoD system," in *Proceedings of the 6th International COnference*, 2010, p. 4.

[197] V. Valancius, N. Laoutaris, L. Massoulié, C. Diot, and P. Rodriguez, "Greening the internet with nano data centers," in *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, 2009, pp. 37-48.

[198] X. Zhou and C.-Z. Xu, "Efficient algorithms of video replication and placement on a cluster of streaming servers," *Journal of Network and Computer Applications,* vol. 30,

pp. 515-540, 2007.

[199] G. Zhang, L. Chiu, and L. Liu, "Adaptive Data Migration in Multi-tiered Storage Based Cloud Environment," in *Cloud Computing (CLOUD), 2010 IEEE 3rd International Conference on*, 2010, pp. 148-155.

[200] M. Eshel, R. L. Haskin, D. Hildebrand, M. Naik, F. B. Schmuck, and R. Tewari, "Panache: A Parallel File System Cache for Global File Access," in *FAST*, 2010, pp. 155-168.

[201] J.-z. Wang, P. Varman, and C.-s. Xie, "Optimizing storage performance in public cloud platforms," *Journal of Zhejiang University SCIENCE C,* vol. 12, pp. 951-964, 2011.

[202] D. Zhao and I. Raicu, "HyCache: a User-Level Caching Middleware for Distributed File Systems."

[203] J. C. McCullough, J. Dunagan, A. Wolman, and A. C. Snoeren, "Stout: An adaptive interface to scalable cloud storage," in *Proc. USENIX Annual Technical Conference*, 2010, p. 101.

[204] N. Tran, M. K. Aguilera, and M. Balakrishnan, "Online migration for geo-distributed storage systems," in *USENIX ATC*, 2011.

[205] B. Dong, Q. Zheng, F. Tian, K.-M. Chao, R. Ma, and R. Anane, "An optimized approach for storing and accessing small files on cloud storage," *Journal of Network and Computer Applications,* 2012.

[206] J. M. Tirado, D. Higuero, F. Isaila, and J. Carretero, "Predictive data grouping and placement for cloud-based elastic server infrastructures," in *Cluster, Cloud and Grid Computing (CCGrid), 2011 11th IEEE/ACM International Symposium on*, 2011, pp. 285-294.

[207] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and implementation of the Sun network filesystem," 1985, pp. 119-130.

[208] P. H. Carns, W. B. Ligon III, R. B. Ross, and R. Thakur, "PVFS: A parallel file system for Linux clusters," 2000, pp. 28-28.

[209] H. Gilbert and H. Handschuh, "Security analysis of SHA-256 and sisters," in *Selected Areas in Cryptography*, 2004, pp. 175-193.

[210] J. Barr. *Use Your own Encryption Keys with S3's Service-Side Encryption*. Available: https://aws.amazon.com/blogs/aws/s3-encryption-with-your-keys [Accessed July 2015]

[211] J. Barr. *Amazon S3 Server Side Encryption for Data at Rest*. Available: https://aws.amazon.com/blogs/aws/new-amazon-s3-server-side-encryption [Accessed July 2015]

[212] J. Daemen and V. Rijmen, *The design of Rijndael: AES-the advanced encryption standard*: Springer, 2002.

[213] D. Mituzas. *Page view statistics for Wikimedia projects*. Available: http://dumps.wikimedia.org/other/pagecounts-raw/ [Accessed July 2015]

[214] *Amazon CloudFront*. Available: http://aws.amazon.com/cloudfront [Accessed July 2015]

[215] *CloudFlare*. Available: https://www.cloudflare.com/plans [Accessed July 2015]

[216] D. Borthakur, "Hadoop avatarnode high availability," ed, 2010.

[217] L. Doclo, "Clustering Tomcat Servers with High Availability and Disaster Fallback," ed, 2011.

[218] Mulesoft, "Tomcat Clustering - A Step By Step Guide," ed.

[219] JetS3t. *JetS3t*. Available: http://jets3t.s3.amazonaws.com [Accessed July 2015]

[220] S. L. Garfinkel, "An evaluation of amazon's grid computing services: EC2, S3, and SQS," 2007.

[221] J. O'Loughlin and L. Gillam, "Towards Performance Prediction for Public Infrastructure Clouds: An EC2 Case Study," in *Cloud Computing Technology and Science (CloudCom), 2013 IEEE 5th International Conference on*, 2013, pp. 475-480.

[222] J. Fulmer, "Siege HTTP regression testing and benchmarking utility," *URL http://www. joedog. org/JoeDog/Siege*.

[223] F. B. Schmuck and R. L. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *FAST*, 2002, p. 19.

[224] E. Cohen, "Surgeons Send'Tweets' from Operating Room," in *CNN Technology*, ed, 2009.

[225] Y. Li, L. Guo, and Y. Guo, "CACSS: Towards a Generic Cloud Storage Service," presented at the CLOSER, 2012.

[226] Y. Li, L. Guo, and Y. Guo, "An Efficient and Performance-Aware Big Data Storage System," in *Cloud Computing and Services Science*, ed: Springer, 2013, pp. 102-116.

[227] N. Leavitt, "Will NoSQL databases live up to their promise?," *Computer,* vol. 43, pp. 12-14, 2010.

[228] I. T. Varley, A. Aziz, C.-s. A. Aziz, and D. Miranker, "No relation: The mixed blessings of non-relational databases," 2009.

[229] M. Stonebraker, "SQL databases v. NoSQL databases," *Communications of the ACM,* vol. 53, pp. 10-11, 2010.

[230] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Record,* vol. 39, pp. 12-27, 2011.

[231] D. Carstoiu, A. Cernian, and A. Olteanu, "Hadoop Hbase-0.20.2 performance evaluation," in *New Trends in Information Science and Service Science (NISS), 2010 4th International Conference on*, 2010, pp. 84-87.

[232] A. Khetrapal and V. Ganesh, "HBase and Hypertable for large scale distributed storage systems," *Dept. of Computer Science, Purdue University,* 2006.

[233] R. C. Team, "R: A language and environment for statistical computing," 2012.

[234] A. Vukotic and J. Goodwill, *Apache Tomcat 7*: Springer, 2011.

[235] *Quartz Scheduler*. Available: http://quartz-scheduler.org/ [Accessed July 2015]

[236] *ApacheBench*. Available: http://httpd.apache.org/docs/2.4/programs/ab.html [Accessed July 2015]

[237] A. L. Goldberger, L. A. Amaral, L. Glass, J. M. Hausdorff, P. C. Ivanov, R. G. Mark, *et al.*, "Physiobank, physiotoolkit, and physionet components of a new research resource for complex physiologic signals," *Circulation,* vol. 101, pp. e215-e220, 2000.

[238] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *Engineering in Medicine and Biology Magazine, IEEE,* vol. 20, pp. 45-50, 2001.

[239] J. Pan and W. J. Tompkins, "A real-time QRS detection algorithm," *Biomedical Engineering, IEEE Transactions on,* pp. 230-236, 1985.

[240] D. Benitez, P. Gaydecki, A. Zaidi, and A. Fitzpatrick, "The use of the Hilbert transform in ECG signal analysis," *Computers in biology and medicine,* vol. 31, pp. 399-406, 2001.

[241] P. De Chazal, M. O'Dwyer, and R. B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *Biomedical Engineering, IEEE Transactions on,* vol. 51, pp. 1196-1206, 2004.

[242] T. Ince, S. Kiranyaz, and M. Gabbouj, "A generic and robust system for automated patient-specific classification of ECG signals," *Biomedical Engineering, IEEE Transactions on,* vol. 56, pp. 1415-1426, 2009.

[243] C. Ye, M. T. Coimbra, and B. V. Kumar, "Arrhythmia detection and classification using morphological and dynamic features of ECG signals," in *Engineering in Medicine and Biology Society (EMBC), 2010 Annual International Conference of the IEEE*, 2010, pp. 1918-1921.

[244] M. R. Homaeinezhad, S. Atyabi, E. Tavakkoli, H. N. Toosi, A. Ghaffari, and R. Ebrahimpour, "ECG arrhythmia recognition via a neuro-SVM–KNN hybrid classifier with virtual QRS image-based geometrical features," *Expert Systems with Applications,* vol. 39, pp. 2047-2058, 2012.

[245] Z. Dokur and T. Ölmez, "ECG beat classification by a novel hybrid neural network," *Computer methods and programs in biomedicine,* vol. 66, pp. 167-181, 2001.

[246] M. Engin, "ECG beat classification using neuro-fuzzy network," *Pattern Recognition Letters,* vol. 25, pp. 1715-1722, 2004.

[247] P. Li, K. L. Chan, S. Fu, and S. M. Krishnan, "An abnormal ecg beat detection approach for long-term monitoring of heart patients based on hybrid kernel machine ensemble," in *Multiple Classifier Systems*, ed: Springer, 2005, pp. 346-355.

[248] B. Babusiak and M. Gala, "Detection of Abnormalities in ECG," in *Information Technologies in Biomedicine*, ed: Springer, 2012, pp. 161-171.

[249] M. C. Chuah and F. Fu, "ECG anomaly detection via time series analysis," in *Frontiers of High Performance Computing and Networking ISPA 2007 Workshops*, 2007, pp. 123-135.

[250] A. Lourenço, H. Silva, and C. Carreiras, "Outlier detection in non-intrusive ECG biometric system," in *Image Analysis and Recognition*, ed: Springer, 2013, pp. 43-52.

[251] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural computation,* vol. 13, pp. 1443-1471, 2001.

[252] C.-C. Chang and C.-J. Lin, "LIBSVM: a library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST),* vol. 2, p. 27, 2011.

[253] B. Scholkopf and A. J. Smola, *Learning with kernels: support vector machines, regularization, optimization, and beyond*: MIT press, 2001.

[254] S. Haykin and N. Network, "A comprehensive foundation," *Neural Networks,* vol. 2, 2004.

[255] J. R. Quinlan, "Induction of decision trees," *Machine learning,* vol. 1, pp. 81-106, 1986.

[256] T. M. Cover and P. E. Hart, "Nearest neighbor pattern classification," *Information Theory, IEEE Transactions on,* vol. 13, pp. 21-27, 1967.

[257] J. A. Hartigan and M. A. Wong, "Algorithm AS 136: A k-means clustering algorithm," *Applied statistics,* pp. 100-108, 1979.

[258] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*, ed: Springer, 2009, pp. 659-663.

[259] A. Krogh, B. Larsson, G. Von Heijne, and E. L. Sonnhammer, "Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes," *Journal of molecular biology,* vol. 305, pp. 567-580, 2001.

[260] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences,* vol. 55, pp. 119-139, 1997.

[261] L. Breiman, "Bagging predictors," *Machine learning,* vol. 24, pp. 123-140, 1996.

[262] J. R. Quinlan, "Bagging, boosting, and C4. 5," in *AAAI/IAAI, Vol. 1*, 1996, pp. 725-730.

[263] A. Bouchama and J. P. Knochel, "Heat stroke," *New England Journal of Medicine,* vol. 346, pp. 1978-1988, 2002.

[264] H.-C. Gunga, M. Sandsund, R. E. Reinertsen, F. Sattler, and J. Koch, "A non-invasive device to continuously determine heat strain in humans," *Journal of Thermal Biology,* vol. 33, pp. 297-307, 2008.

[265] J. Stephant, A. Charara, and D. Meizel, "Virtual sensor: application to vehicle sideslip angle and transversal forces," *Industrial Electronics, IEEE Transactions on,* vol. 51, pp. 278-289, 2004.

[266] A. Kusiak, H. Zheng, and Z. Zhang, "Virtual wind speed sensor for wind turbines," *Journal of Energy Engineering,* vol. 137, pp. 59-69, 2010.

[267] N. Raveendranathan, S. Galzarano, V. Loseu, R. Gravina, R. Giannantonio, M. Sgroi*, et al.*, "From modeling to implementation of virtual sensors in body sensor networks," *Sensors Journal, IEEE,* vol. 12, pp. 583-593, 2012.

[268] J. H. van Beek, F. Supandi, A. K. Gavai, A. A. de Graaf, T. W. Binsl, and H. Hettling, "Simulating the physiology of athletes during endurance sports events: modelling human energy conversion and metabolism," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences,* vol. 369, pp. 4295-4315, 2011.

[269] J. Reith, H. Jorgensen, P. Pedersen, H. Nakamaya, L. Jeppesen, T. Olsen*, et al.*, "Body temperature in acute stroke: relation to stroke severity, infarct size, mortality, and outcome," *The Lancet,* vol. 347, pp. 422-425, 1996.

[270] J. A. Stolwijk, *A mathematical model of physiological temperature regulation in man* vol. 1855: National Aeronautics and Space Administration, 1971.

[271] G. Havenith, "Interaction of clothing and thermoregulation," *Exogenous Dermatology,* vol. 1, pp. 221-230, 2003.

[272] L. Keytel, J. Goedecke, T. Noakes, H. Hiiloskorpi, R. Laukkanen, L. Van Der Merwe*, et al.*, "Prediction of energy expenditure from heart rate monitoring during submaximal exercise," *Journal of Sports Sciences,* vol. 23, pp. 289-297, 2005.

[273] M. D. Mifflin, S. St Jeor, L. A. Hill, B. J. Scott, S. A. Daugherty, and Y. Koh, "A new predictive equation for resting energy expenditure in healthy individuals," *The American journal of clinical nutrition,* vol. 51, pp. 241-247, 1990.

[274] *OpenWeatherMap*. Available: http://www.openweathermap.org [Accessed July 2015]

[275] *Weather Underground.* Available: http://www.wunderground.com [Accessed July 2015]

[276] *Google Map*. Available: http://maps.google.com [Accessed July 2015]

[277] M. Haklay and P. Weber, "Openstreetmap: User-generated street maps," *Pervasive Computing, IEEE,* vol. 7, pp. 12-18, 2008.

[278] *BBC Weather*. Available: http://www.bbc.co.uk/weather [Accessed 2013]

[279] P. L. Houtekamer and H. L. Mitchell, "Data assimilation using an ensemble Kalman filter technique," *Monthly Weather Review,* vol. 126, pp. 796-811, 1998.

[280] O. Pearson, M. Busse, R. Van Deursen, and C. Wiles, "Quantification of walking mobility in neurological disorders," *Qjm,* vol. 97, pp. 463-475, 2004.

[281] R. W. Motl and B. M. Sandroff, "Objective monitoring of physical activity behavior in multiple sclerosis," *Physical Therapy Reviews,* vol. 15, pp. 204-211, 2010.

[282] R. Motl, "Physical activity and its measurement and determinants in multiple sclerosis," *Minerva medica,* vol. 99, pp. 157-165, 2008.

[283] B. M. Sandroff, R. W. Motl, L. A. Pilutti, Y. C. Learmonth, I. Ensari, D. Dlugonski, *et al.*, "Accuracy of StepWatch™ and ActiGraph Accelerometers for Measuring Steps Taken among Persons with Multiple Sclerosis," *PloS one,* vol. 9, p. e93511, 2014.

[284] F. Agrafioti, D. Hatzinakos, and A. K. Anderson, "ECG pattern analysis for emotion detection," *Affective Computing, IEEE Transactions on,* vol. 3, pp. 102-115, 2012.

[285] Y. Xu, G. Liu, M. Hao, W. Wen, and X. Huang, "Analysis of affective ECG signals toward emotion recognition," *Journal of Electronics (China),* vol. 27, pp. 8-14, 2010.