Exploring Software Engineering Approaches to Developing Mobile Applications in Hybrid Cloud Computing Environments

A Research and Translation Center
Funded Research Project
GP-CCIS-2013-11-18

Final report
November 2, 2016.
Exploring Software Engineering Approaches to Developing Mobile Applications in Hybrid Cloud Computing Environments

Principal Investigator: Dr. Basit Qureshi
Co-Investigator: Dr. Fakhry Khellah

This project is supported by Research and Translation Center (RTC) under the research grant number GP-CCIS-2013-11-18. The investigators would like to express their gratitude and appreciation to the RTC. The investigators also extend thanks to Prince Sultan University for providing facilities and support to carry out this research
Abstract:

Cloud Computing has been at the forefront in recent development of software applications. Although the technology is being used in recent years with many benefits such as reducing the cost for maintaining servers whereas improving the availability, reliability and scalability of the systems, there have been wide spread privacy concerns since personal information of individuals can be exposed to unauthorized parties. To this end, researchers have proposed using hybrid cloud environment integrating the public and private cloud infrastructure where the privacy centered data is stored in private cloud while non-sensitive data is kept available in the public cloud. While working with newer technologies is challenging, researchers and students in academia face difficulties in developing software and testing it. A very few number of works present sound engineering mechanism for developing such software. This research discusses issues in developing mobile-hybrid cloud applications using agile software development methodologies. The project is executed in two phases, the first phase implements a public cloud services Sha-Mo-Cloud, leveraging public cloud services for storing data including images and videos captured from user’s mobile devices. Users can upload, access, manage and share personal content using web interface as well as a mobile application. By adopting the Agile Extreme Programming (XP) process, this work also reports the software engineering aspects for the development of ShaMoCloud. The software construction team previously developed applications using Team Software Process (TSP). The work presented in this part compares and contrasts the experiences of the ShaMoCloud development team with four similar works performed recently, and with previously developed TSP work. The second phase investigates the realization of hybrid cloud by integrating ShaMoCloud public services with a private cloud cluster. As part of this phase we investigate the suitability of using affordable low-energy single board computers for building a Low-Cost Cloud Computing Cluster (LoC4). We provide design and deployment details for the LoC4 cluster and extensively tested it for performance using popular benchmarks. We investigate deployment of Hadoop on the cluster and provide configuration details for improving performance using DFSIO and TeraSort Hadoop benchmarks. Although the performance of the cluster is comparatively slow to a small scale data-center,
nevertheless the low cost and affordable platform provides excellent opportunities for academic research in cloud computing.

The project had significant impact on teaching and learning at Prince Sultan University through the integration of a graduate course SE501 Software Development Processes and an undergraduate course SE409 Cloud Computing. A Master in Software Engineering student successfully completed his thesis work as part of this project. At the time of writing this report this 9-month project has resulted in a published peer-reviewed conference paper in ANT2015 United Kingdom. One conference paper is submitted and currently under review. The authors are also extending their work to be submitted to a Scopus indexed journal.
Abstract (Arabic)

أصبح استخدام الحوسبة السحابية جزء من تقنيات هندسة البرمجيات الحديثة حيث يسهم في تقليل التكلفه وزيادة قابلية التوسعه ومدى الثقه.

الاتجاه الحالي في هذه التقنيه يقوم على وحفظ البيانات غير الحساسه على السحابه العامه وحفظ البيانات الحساسه على السحابه الخاصه. نظرا لأن هذه التقنيه حديثه فالأزال هناك صعوبات في تطوير البرمجيات. الهدف من هذا البحث هو تطوير هيكلة لهندسة البرمجيات باستخدام تقنية الحوسبة السحابية على مرحلتين: المرحلة الأولى تطوير تطبيق لاستخدام السحابه العامه والمرحلة الثانيه نحو بناء سحابه خاصه واعادة تطوير التطبيق عليها.
Abstract: ........................................................................................................................................... ii
Abstract (Arabic).................................................................................................................................. iv
List of Figures ...................................................................................................................................... vii
List of Tables ...................................................................................................................................... ix
Commonly used abbreviations ......................................................................................................... x
1. Introduction ..................................................................................................................................... 1
   1.1 Problem Statement ....................................................................................................................... 2
   1.2 Research Methodology ............................................................................................................... 3
   1.3 Research Contributions ............................................................................................................. 6
2. Literature review .............................................................................................................................. 9
   2.1 Cloud Computing ....................................................................................................................... 9
      2.1.1 Service Models .................................................................................................................. 9
      2.1.2 Deployment Models ......................................................................................................... 11
   2.2 Mobile Cloud Computing ......................................................................................................... 13
   2.3 Agile Software Development ................................................................................................... 17
   2.4 Low Cost Cloud Computing Clusters ..................................................................................... 21
3. ShaMoCloud Service Architecture ............................................................................................... 25
   3.1 What is ShaMoCloud? .............................................................................................................. 25
   3.2 Requirement Analysis for ShaMoCloud Service ....................................................................... 25
   3.3 Architecture Details ................................................................................................................ 31
      3.3.1 User Service Layer .......................................................................................................... 31
      3.3.2 Cloud Services Layer ....................................................................................................... 34
      3.3.3 Data Layer ....................................................................................................................... 35
   3.4 Implementation Details ............................................................................................................ 36
      3.4.1 User Interface Layer ........................................................................................................ 36
      3.4.2 Setting up cloud web and data services .......................................................................... 41
4. Software Engineering aspects of ShaMoCloud ............................................................................. 46
   4.1 Project Planning ....................................................................................................................... 46
   4.2 Requirements and User Stories ............................................................................................... 47
   4.3 Application Development ....................................................................................................... 50
4.4 Application Testing ........................................................................................................ 50
4.5 Continuous Integration and Deployment ........................................................................ 55
5. Analysis of Development aspects and comparison with other works ......................... 57
  5.1 Issues for Mobile Cloud Development in Recent Works ....................................... 57
  5.2 Gains in Developing ShaMoCloud Service ............................................................... 59
  5.3 Issues with ShaMoCloud Service .............................................................................. 61
  5.4 Transition from TSP to Agile XP on Cloud ............................................................... 62
  5.5 Issues with ShaMoCloud Service .............................................................................. 63
  6.1 Motivation .................................................................................................................. 65
  6.2 The Single Board Computing Platforms .................................................................. 69
  6.3 The Low Cost Cloud Computing Cluster (LoC4) ..................................................... 71
    6.3.1 Design of LoC4 Cluster ....................................................................................... 71
    6.3.2 Hadoop Deployment ......................................................................................... 72
  6.4 Performance Evaluation ......................................................................................... 75
    6.4.1 Performance Benchmarking ............................................................................... 75
    6.4.2 Hadoop Benchmarks ....................................................................................... 81
7. Conclusions and Future work ....................................................................................... 84
  7.1 Concluding Remarks ............................................................................................... 84
  7.2 Future Work ............................................................................................................. 86
  7.3 Recommendations ................................................................................................... 87
References .......................................................................................................................... 89
Appendices .......................................................................................................................... 96
  Appendix A: ShaMoCloud Web Interfaces .................................................................. 96
  Appendix B: ShaMoCloud Mobile Application ............................................................ 100
List of Figures

Figure 2-1 Cloud Service Models [7] ........................................................................................................... 11
Figure 2-2 Cloud Deployment Models [34].................................................................................................. 13
Figure 2-3 Project execution stages during planning, development, and integration phases [19] .............. 18
Figure 2-4 Comparisons of the development time in days [16] ................................................................. 19
Figure 2-5 MADONA methodology [17] ..................................................................................................... 20
Figure 2-6 Cloud-based mobile software development process [18] ......................................................... 21
Figure 3-1 ShaMoCloud use case diagram ............................................................................................... 26
Figure 3-2 Add friend workflow ............................................................................................................... 27
Figure 3-3 Access control workflow ....................................................................................................... 28
Figure 3-4 Video services ......................................................................................................................... 29
Figure 3-5 ShaMoCloud Data Flow Diagram ............................................................................................ 30
Figure 3-6 ShaMoCloud architecture overview ....................................................................................... 32
Figure 3-7 Android application architecture and alignment with WCF .................................................. 33
Figure 3-8 ShaMoCloud web-based interface ........................................................................................... 36
Figure 3-9 Activity and Fragment relationship ......................................................................................... 37
Figure 3-10 Hosting Fragment in Activity source code ............................................................................. 38
Figure 3-11 Relationship between AsyncTask class, JSON parser class, and HTTP libraries ................. 39
Figure 3-12 Code for web service class with AsyncTask class default methods ....................................... 40
Figure 3-13 AWS management console ................................................................................................... 42
Figure 3-14 FileZilla software client ....................................................................................................... 43
Figure 3-15 ShaMoCloud Entity Relationship Diagram ........................................................................... 45
Figure 4-1 Fiddler web debugger example ............................................................................................... 52
Figure 4-2 Log steps in Android ............................................................................................................... 52
Figure 4-3 Android emulator .................................................................................................................... 53
Figure 6-1 A Hybrid Cloud Environment for a PHR System at HCP ..................................................... 68
Figure 6-2 (a) Raspberry Pi Model 3B (b) HardKernel Ordoid Xu-4 ........................................................... 70
Figure 6-3 The LoC4 Cluster with 6 Xu4 (Rack mounted) and 5 RPi 3B in cases ................................. 74
Figure 6-4 NetPIPE benchmark results for Xu4 and RPi clusters considering latencies with data size in terms of bytes on the x-axis. ...................................................................................... 77
Figure 6-5 NetPIPE benchmark results for Xu4 and RPi clusters considering bandwidth with data size in terms of bytes on the x-axis. ...................................................................................... 78
Figure 6-6 TestDFSIO Read and Write Time for RPi and Xu4 sub-clusters in LoC4 Cluster. .............. 80
Figure 6-7 TeraGen and TeraSort execution time for LoC4 cluster. .................................................... 82
Figure 9-1 ShaMoCloud home page ......................................................................................................... 96
Figure 9-2 ShaMoCloud Login & registration page .................................................................................. 97
Figure 9-3 Shared video page ................................................................................................................... 97
Figure 9-4 Manage video page ............................................................................................................... 98
Figure 9-5 Manage friend page ........................................................................................................... 98
Figure 9-6 New friends form page ........................................................................................................ 99
Figure 9-7 ShaMoCloud App home screen ............................................................................................ 100
Figure 9-8 Cloud video screen .............................................................................................................. 101
Figure 9-9 Pop up menu sample ............................................................................................................ 101
List of Tables

Table 4-1 ShaMoCloud user stories .......................................................... 47
Table 4-2 Agile XP process data for ShaMoCloud case study.......................... 50
Table 5-1 Comparison of recent research works, strengths, and weaknesses ............. 58
Table 5-2 Process time comparison between ShaMoCloud and DSDM ...................... 61
Table 5-3 TSP and XP process comparison .................................................. 62
Table 6-1 Comparison of Raspberry Pi 3B and HardKernel Odroid Computers ............ 70
Table 6-2 Hadoop configuration properties for LoC4 Cluster .............................. 75
Table 6-3 Average CPU Execution Time for nodes with n threads ......................... 76
Table 6-4 Read and Write Throughput (KB/sec) for individual devices in the clusters using FIO ................................................................. 77
Commonly used abbreviations

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AM</td>
<td>Agile Modeling</td>
</tr>
<tr>
<td>ASD</td>
<td>Adaptive Software Development</td>
</tr>
<tr>
<td>AVD</td>
<td>Android Virtual Device</td>
</tr>
<tr>
<td>AWS</td>
<td>Amazon Web Services</td>
</tr>
<tr>
<td>CC</td>
<td>Cloud Computing</td>
</tr>
<tr>
<td>CRC</td>
<td>Class Responsibility Collaborator</td>
</tr>
<tr>
<td>Crystal</td>
<td>Crystal Methods</td>
</tr>
<tr>
<td>CTQ</td>
<td>Cost Time Quality</td>
</tr>
<tr>
<td>DSDM</td>
<td>Dynamic Systems Development Method</td>
</tr>
<tr>
<td>EC2</td>
<td>Amazon Elastic Cloud Computing</td>
</tr>
<tr>
<td>ERD</td>
<td>Entity Relationship Diagram</td>
</tr>
<tr>
<td>FDD</td>
<td>Feature Driven Development</td>
</tr>
<tr>
<td>GAE</td>
<td>Google App Engine</td>
</tr>
<tr>
<td>HBase</td>
<td>Hadoop Data Base</td>
</tr>
<tr>
<td>HDFS</td>
<td>Hadoop Distributed File System</td>
</tr>
<tr>
<td>IaaS</td>
<td>Infrastructure-as-a-Service</td>
</tr>
<tr>
<td>IDE</td>
<td>Integrated development environment</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Server</td>
</tr>
<tr>
<td>ISD</td>
<td>Internet Speed Development</td>
</tr>
<tr>
<td>JSON</td>
<td>JavaScript Object Notation</td>
</tr>
<tr>
<td>LOC</td>
<td>Lines of code</td>
</tr>
<tr>
<td>LoC4</td>
<td>Low Cost Cloud Computing Cluster</td>
</tr>
<tr>
<td>MADONA</td>
<td>Methodology for Automatic Development of clOud-based busiNess Application</td>
</tr>
<tr>
<td>MCC</td>
<td>Mobile Cloud Computing</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OSS</td>
<td>Open Source Software Development</td>
</tr>
<tr>
<td>PaaS</td>
<td>Platform-as-a-Service</td>
</tr>
<tr>
<td>PP</td>
<td>Pragmatic Development</td>
</tr>
<tr>
<td>PSP</td>
<td>Personal Software Process</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service</td>
</tr>
<tr>
<td>REST</td>
<td>Representational State Transfer</td>
</tr>
<tr>
<td>RPi</td>
<td>Raspberry Pi Computer</td>
</tr>
<tr>
<td>RUP</td>
<td>IBM Rational Unified Process</td>
</tr>
<tr>
<td>SaaS</td>
<td>Software-as-a-Service</td>
</tr>
<tr>
<td>SAMI</td>
<td>Service-based Arbitrated Multi-tier Infrastructure</td>
</tr>
<tr>
<td>SBC</td>
<td>Single Board Computers</td>
</tr>
<tr>
<td>ShaMoCloud</td>
<td>Sharing-Mobile-Cloud</td>
</tr>
<tr>
<td>SOAP</td>
<td>Simple Object Access Protocol</td>
</tr>
<tr>
<td>TSP</td>
<td>Team Software Process</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>WCF</td>
<td>Windows Communication Foundation</td>
</tr>
<tr>
<td>XML</td>
<td>EXtensible Markup Language</td>
</tr>
<tr>
<td>XP</td>
<td>Extreme Programming</td>
</tr>
<tr>
<td>Xu4</td>
<td>HardKernel Odroid Xu4 computer</td>
</tr>
</tbody>
</table>
1. Introduction

In recent years there is a trend in migrating popular applications to the cloud with social networking applications such as Facebook, twitter etc. amongst many others. Many software giants have been offering their software and services in the cloud such as Google-docs [2], Gmail [3], MS-Office 360 [4] etc. When a Cloud is made available as a pay-as-you-go manner in the public domain, it is called a Public Cloud; the service being sold is Utility Computing [1]. Microsoft Azure [5], Google App Engine [6], Amazon EC2 [7] and recently IBM CloudOne [8] have been providing Software-as-a-Service (SaaS) and Platform-as-a-service (PaaS) for developers to build applications using corporate infrastructure. Cloud computing provides storage for large scale data on external servers allowing researchers and developers to easily access the data [48]. Mobile cloud computing utilizes the ever-increasing pool of computing resources available on mobile devices. In mobile cloud computing, mobile devices can rely on cloud computing and information storage resource to perform computationally intensive operations such as searching, data mining, and multimedia processing [21]. Various frameworks, middleware and applications have been developed in recent years providing storage and computation as cloud services on mobile devices [9, 13-14]. Authors in [11-12] provide a survey of recent advances in Mobile Cloud computing.

Although the technology is being used in recent years with many benefits such as reducing the cost for maintaining servers whereas improving the availability of the systems, there have been wide spread privacy concerns since personal information of individuals can be exposed to unauthorized parties. To this end, recently researchers have proposed several cloud-based secure systems using encryption where personal and sensitive data is encrypted and stored in cloud based storage. On the other hand, usually such privacy sensitive data are typically accessible by multiple data owners including data owners, clients, stakeholders. etc., complexity of key management in encrypted systems increases for the owners and users alike. To address these concerns researchers in [49, 52-53] address these issues using hybrid cloud environment integrating the public and private cloud
infrastructure where the encrypted data is stored in public cloud while the encryption keys and policies for multiple data owners are stored and executed in the private cloud.

Building hybrid cloud applications requires development and testing of software and cloud services on a privately owned cloud infrastructure. In academics, students and researchers need to develop middleware and test cloud based applications. While subscription to cloud service providers could be expensive for some, deployment and access to private cloud infrastructure could be costly. It is important for students in academic environments to be exposed to real cloud computing infrastructure, visiting a cloud service provider’s facilities is not enough for deeper understanding of underlying technologies. While universities and academic institutions provide hands-on experience in cloud based health informatics, essentially universities need to provide access to a suitable cloud computing infrastructure that can be used for experimentation, research, and teaching. Deploying the cloud infrastructure in universities could be a very costly effort.

1.1 Problem Statement
In this research we respond to two main questions

- Is it possible to develop Mobile hybrid cloud applications using Extreme Programming with students in an academic environment and deploy such applications using well known public service providers such as Amazon EC2?
- Investigate the possibility of deploying an affordable cloud computing cluster for testing hybrid cloud applications in academic environment.

To address the first question, the Sharing-Mobile-Cloud (ShaMoCloud) is developed as a case study to address the aforementioned aspects. A web interface is developed to interact with the cloud service. In addition, a mobile application is also developed that connects with the cloud service in order to provide support to mobile users. ShaMoCloud is developed in an academic setting with a team of software engineering students. Various aspects of software development are compared with results and recommendations in four recently completed works: Dynamic Systems Development Methods (DSDM) [16], Automatic Development of cloud-based business Application (MADONA) [17], CMPlayer [18], and work done by
Manuja and Manisha [19]. ShaMoCloud is successfully developed and deployed in Amazon EC2. A mobile application was also built using Android and was tested for functionality.

For the second question, we investigate the use of low cost low energy equipment such as Single Board Computers (SBC) [69] in construction and deployment of private cloud infrastructure. Due to the small size, low energy consumption and affordable costs, clusters composed of these computers can be deployed in a short period of time. We present design and architecture of LoC4 Cluster, deploy this cluster in compact area and install Hadoop as well as HBase for distributed processing and storage. We extensively study the performance of LoC4-clusters in terms of computation, storage, network throughput and power consumption.

1.2 Research Methodology

There are various research challenges in developing mobile applications for a hybrid cloud computing environment. To address these challenges, the research team considered two scenarios.

- Availability of content where privacy of data is not a big concern. We implement ShaMoCloud, a cloud-based service that provides content management for media files, including images and videos. ShaMoCloud is implemented in a public cloud environment for maximum accessibility and scalability. The case study is designed and developed for open accessibility from various types of client devices. A web-based interface provides users with access to the service through any web browser. In addition, an Android-based application was developed to connect to the service. Users need to register and set up an account with a two-step authentication process. Next, the users can upload media content, including images or videos, to their accounts. ShaMoCloud was developed and hosted in Amazon EC2 using a single instance for testing purposes.

- Availability of content where privacy of data is of importance. We study the requirements of a Cloud based Electronic Heath Record and Patient Health Records Systems. In these systems the privacy of large amounts of data is of much importance. Researchers have developed scenarios where either the data is encrypted and kept in
the public cloud where as the encryption keys are stored in private cloud, or the privacy sensitive data is kept in the private cloud. We consider the scenario where the personal and sensitive information would be stored in the private cloud whereas the rest of the content is kept in the public cloud.

To address the first scenario, we explored two aspects of mobile cloud application development.

- Design and implementation of a hybrid cloud environment to satisfy the requirements of a mobile cloud content management and sharing framework, preserving the privacy requirements.
- Study the various aspects of developing a mobile cloud application using extreme programming as an agile development methodology. Integrate usage of cloud technologies in software team development, integration and testing. Compare and contrast prior experiences of the team with Team Software Process Data with results obtained from this development.

The ShaMoCloud framework is developed as a case study to address the above mentioned two aspects. The framework is composed of a Cloud Service integrating public and private cloud. A web interface is developed to interact with the cloud service. Additionally, a mobile application is also developed that connects with the cloud service, providing support to the user on the go. Various aspects of software development are compared with results and recommendations in four recently completed projects, DSDM [28], MADONA [29], CMPlayer [30] and work done by Manuja et. al. [19].

The researchers trained a team of one graduate student enrolled in Master of Software Engineering at Prince Sultan University and four undergraduate students enrolled in Bachelors of Computer Science program. The students learned about the Team Software Process [47] and completed all the assignments given in the course work collecting software process data such as average Lines of Code written, number of defects per assignment, identification for types of defects introduced in code and time taken to remove defects. Using
the data collected from these assignment students learned how to introduce quality mechanisms in writing code and were able to predict future defects using linear regression which is a tool commonly used in TSP for defect prevention. Students tried various other assignments to advance their quality code writing capabilities and improve the quality of the prediction model. After a period of three months of training students were introduced to the Extreme Programming concept and given tasks to prepare for process migration from TSP to Extreme Programming.

To address the second scenario, the team extended the ShaMoCloud service to connect to a private network. ShaMoCloud is extended to implement a middleware projecting data from the Amazon EC2 host to a webserver in the private cloud. SOAP protocol based REST web services were used to provide access to the service. JSON parser was used for XML compatibility for interface. Due to the lack of availability of resources to test cloud applications in a data center at Prince Sultan University and delay in funding, the team investigated the use of low cost computers to build a Hadoop cluster for writing and testing applications. In this regard the team thoroughly evaluated single board computers in a cluster using various performance benchmarks. The undergraduate student’s team was used to construct the LoC4 cluster using Raspberry Pi Model 3B and Odroid Xu4 Computers. Later the researchers constructed an image using required platform and tools including configuration parameters for the Hadoop environment as well as testing binaries to be used for performance evaluation of various benchmarks. Once the image was prepared, students helped write the image to each node of the cluster. Hadoop configuration was tested and the experiments were prepared and subsequently executed on the cluster. The team studied the effect of various configurations and parameters for tweaking Hadoop performance.

In terms of project management, due to the drastic reduction in approved budget for the project as well as delays in payment of installments, the team decided to complete the project in two parts.

Part 1. Design and develop ShaMoCloud cloud service on Amazon EC2 to handle video upload/view/download requests. This cloud service will allow users to manage their content
such as storing and accessing their video postings later on. Design and develop a mobile application that would provide connectivity with the cloud service. A mobile user would be able to access his videos, post his videos and share his videos with other users. Extend the ShaMoCloud service as web-service to application running in the Private cloud.

Part 2. Investigate the use of Single board computers to build a low cost cloud computing cluster. Configure and deploy Hadoop and Hbase environments on this cluster and provide access to this cluster as a testbed for cloud computing research.

Both of these parts of the project were completed. The team looks forward to release of next three installments for the project budget, so as to re-compensate the team members for the purchase of all hardware and software equipment.

1.3 Research Contributions
The motivation behind this work was to study the application of Software Engineering principles in an academic environment while developing a mobile-hybrid cloud computing application as a case study. The goal was achieved by

1. Designing and developing a mobile application in order to explore the development aspects for such applications. We designed and developed a web and mobile application system called ShaMoCloud, which is a mobile–cloud based environment for content/media management. The design and implementation of a cloud environment was detailed with a focus on integrating the three tiers of the service, while preserving data security and privacy. Various aspects of development using XP as an Agile development methodology were discussed. The work was compared with similar works that highlighted the gains and challenges in development. Our experience showed that mobile cloud application development can be integrated with Agile development methodologies to reduce cost and time, and improve software quality.

2. We investigate the use of SBC in a low cost cloud computing cluster. We consider two kinds of popular platforms Raspberry Pi 3B and Odroid Xu-4 using ARM Cortex
Processors. The LoC4 cluster was deployed comprising of 11 SBCs interconnected in a network topology over a gigabit Ethernet. Hadoop was installed on the cluster with configuration to suit the SBC’s memory and storage requirements. Extensive performance evaluation was carried out for both platforms using various performance benchmarking tests for task execution time, I/O read/write and network performance. Further to this, we also conducted Hadoop benchmarks TestDFSIO and TeraSort for performance evaluation of the Hadoop deployment in the cluster. Results show that Hadoop with Hbase can be successfully deployed on a cluster built using low cost SBCs. In terms of compute power comparison with a traditional server, the limited onboard resources of these SBC yield very poor comparative performance. Nevertheless, the LOC4 Hadoop cluster deployment was successful and our experience provides excellent opportunities for academic research in cloud based health informatics.

The project had significant impact on teaching and learning at Prince Sultan University through the integration of a graduate course SE501 Software Development Processes and an undergraduate course SE409 Cloud Computing. A Master in Software Engineering student successfully completed his thesis work as part of this project.

This research has culminated in the following publications in international peer reviewed venues:


The remainder of this report is organized as follows.
Chapter 2 provides recent work in Mobile Cloud Computing, Mobile Cloud Software Development processes and Low cost Cloud Infrastructure development.

Chapter 3 provides details of the ShaMoCloud framework. Motivation, requirements analysis, design & architecture as well as implementation details for the framework are presented.

Chapter 4 presents details related to the software engineering aspects of ShaMoCloud development.

Chapter 5 provides analysis of development aspects, gains and issues encountered in comparison with similar works. A discussion on process migration from TSP to XP is also presented.

Chapter 6 details the design and architecture of the LoC4 Cloud Cluster. Details about Hadoop deployment, performance evaluation using popular benchmarks as well as Hadoop benchmarks are also provided.

The report concludes with recommendations and directions for future work, in Chapter 7.
2. Literature review

This section presents recent work in the area of mobile cloud computing and related software development models.

2.1 Cloud Computing

“Cloud computing refers to both the applications delivered as services over the Internet and the hardware and systems software in the datacenters that provide those services” [1]. From different perspectives, the cloud computing types can be divided into two groups:

2.1.1 Service Models

Service model refers to the services type that provided by cloud providers such as Amazon EC2 and Microsoft Azure etc. Primarily there are three kinds of service models, Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). Following provides brief details for each service model.

**Software as a Service (SaaS)** refers to delivery of software applications or services that run in cloud provider infrastructure. The consumer does not control or manage the resources like servers, operating system and storages. Users are provided access to application software and databases. Cloud providers manage the infrastructure and platforms that run the applications. SaaS is sometimes referred to as "on-demand software" and is usually priced on a pay-per-use basis or using a subscription fee.

In the SaaS model, cloud providers install and operate application software in the cloud and cloud users access the software from cloud clients. Cloud users do not manage the cloud infrastructure and platform where the application runs. This eliminates the need to install and run the application on the cloud user’s own computers, which simplifies maintenance and support. Cloud applications are different from other applications in their scalability—which can be achieved by cloning tasks onto multiple virtual machines at run-time to meet changing work demand. This process for load balancing in virtual machines is transparent to the cloud user, who sees only a single access point. Proponents claim SaaS allows a
business the potential to reduce IT operational costs by outsourcing hardware and software maintenance and support to the cloud provider. This enables the business to reallocate IT operations costs away from hardware/software spending and personnel expenses, towards meeting other goals. [31] In addition, with applications hosted centrally, updates can be released without the need for users to install new software. One drawback of SaaS is that the users’ data are stored on the cloud provider’s server. As a result, there could be unauthorized access to the data [32].

**Platform as a Service (PaaS)** refers to delivery of deployment services onto cloud provider infrastructure whereas the consumer can deploy certain applications with limitations on control on management of the resources. In the PaaS models, cloud providers deliver a computing platform, typically including operating system, programming language execution environment, database, and web server. Application developers can develop and run their software solutions on a cloud platform without the cost and complexity of buying and managing the underlying hardware and software layers. With some PaaS offers like Microsoft Azure and Google App Engine, the underlying computer and storage resources scale automatically to match application demand so that the cloud user does not have to allocate resources manually [31].

**Infrastructure as a Service (IaaS)** refers to delivery of infrastructure provisioning services to the consumer where the consumer can provision servers, operating system, storage and others fundamental resources [11-12].

Providers of IaaS offer computers physical or virtual machines – and other resources. A hypervisor, such as Xen¹, Oracle VirtualBox², KVM³, VMware ESX/ESXi⁴, or Hyper-V allows guests to run instances as virtual machines. Pools of hypervisors within the cloud operational support-system can support large numbers of virtual machines and the ability

---

¹ www.xen.com  
² www.virtualbox.org  
³ www.intel.com/virtualKVMgateway  
⁴ www.vmware.com/products/esxi-and-esx
to scale services up and down according to customers' varying requirements. IaaS-cloud providers supply these resources on-demand from their large pools installed in data centers. To deploy their applications, cloud users install operating-system images and their application software on the cloud infrastructure. In this model, the cloud user patches and maintains the operating systems and the application software. Cloud providers typically bill IaaS services on a utility computing basis: cost reflects the amount of resources allocated and consumed [7]. The following figure provides a summative overview of the service models [31].

![Cloud Service Models](image)

**Figure 2-1 Cloud Service Models [7]**

### 2.1.2 Deployment Models
Deployment Model refers to how cloud infrastructure owner host and offer the cloud computing services. These are public, private and hybrid cloud models [10].

**Public Cloud model** refers to consumer usage of an organization's cloud infrastructure on a pay per use basis. A cloud is called a "public cloud" when the services are rendered over a network that is open for public use. Public cloud services may be free. Generally, public cloud service providers like Amazon AWS, Microsoft and Google own and operate the infrastructure at their data center and access is generally via the Internet.
**Private Cloud model** refers to internal deployment, management and operation of an organization’s own cloud infrastructure. Private cloud is cloud infrastructure operated solely for a single organization, whether managed internally or by a third-party, and hosted either internally or externally. Self-run data centers are generally capital intensive requiring organizations to pay for hardware and software acquisitions, upgrades and maintenance. Additionally, they have a significant physical footprint, requiring allocations of space, hardware, and environmental controls. These assets have to be refreshed periodically, resulting in additional capital expenditures [34]. They have attracted criticism because users "still have to buy, build, and manage them" and thus do not benefit from less hands-on management.

**Hybrid Cloud model** refers to an integration on both public and private cloud models within an organization. Gartner, Inc. defines a hybrid cloud service as a cloud computing service that is composed of some combination of private, public and community cloud services, from different service providers [34]. A hybrid cloud service crosses isolation and provider boundaries so that it can’t be simply put in one category of private, public, or community cloud service. It allows one to extend either the capacity or the capability of a cloud service, by aggregation, integration or customization with another cloud service.

Varied use cases for hybrid cloud composition exist. For example, an organization may store sensitive client data in house on a private cloud application, but interconnect that application to a business intelligence application provided on a public cloud as a software service [35]. Hybrid cloud adoption depends on a number of factors such as data security and compliance requirements, level of control needed over data, and the applications an organization uses. The following figure shows the illustrations of public, private and hybrid cloud deployment models.
2.2 Mobile Cloud Computing

Integrating mobile applications with cloud environments have recently become a popular approach in the industry. The term mobile cloud computing means to run an application such as Google’s Gmail for Mobile on a remote resource rich server (in this case, Google servers), while the mobile device acts like a thin client connecting over to the remote server through 3G [11].

Mobile devices are typically challenged in terms of processing power, storage capabilities and communication issues. Although mobile computing provides users with mobility that allows them to accomplish various tasks on the go, mobile cloud computing provides the benefits of both mobile device and cloud computing. In a mobile cloud-computing environment, mobile device’s data storage and data processing are outsourced to the cloud. This increases battery life, computation processing, sharing of information as well as running computation intensive applications on a resource constraint mobile device [12]. Papers [13-14] present examples of various applications developed in Mobile Cloud Computing environment.
Several approaches and techniques have been proposed to augment mobile devices by leveraging cloud computing.

In recent years, applications targeted at mobile devices have started becoming abundant in various categories such as entertainment, health, games, business, social networking, travel and news. The popularity of these is evident by browsing through mobile application download centers such as Apple's iTunes or Andriod market etc. Mobile computing provides a platform to the user when and where, user needs it, irrespective of user movement or location, thus supporting location independence [39]. Indeed, ‘mobility’ is one of the characteristics of a pervasive computing environment where the user is able to continue his work seamlessly regardless of his movement. However, with mobility come its inherent problems such as resource scarceness. Real time mobile applications that demand high levels of responsiveness, that in turn, demand intensive computing resources. Some mobile applications, such as location based social networking, process and make use of the phone’s various sensor data. However, extensive use of sensors, such as obtaining a GPS reading, is expensive in terms of energy and this limits the mobile phone in providing the user a better service through its embedded sensors. Furthermore, applications that require extensive processing – image processing for video games, speech synthesis, natural language processing, augmented reality, wearable computing all these demand high computational capacities thus restricting the developers in implementing applications for mobile phones.

The concept of offloading data and computation in cloud computing is used to address the inherent problems in mobile computing by using resource providers other than the mobile device itself to host the execution of mobile applications. Such an infrastructure where data storage and processing could happen outside the mobile device could be termed a ‘mobile cloud’ [44]. By exploiting the computing and storage capabilities of the mobile cloud, computer intensive applications can be executed on limited resource mobile devices.

Recently several attempts have been made in developing cloud services software to provide content to mobile users. A pioneering effort in developing mobile content as a service in the cloud was reported by Raatikainen et. Al in [44]. The proposed architecture maintains the content in the mobile device where it was first created. The resulting design leads to a mobile
device cloud that treats devices, together with the content and resources they host, as first-class cloud citizens. The prototype developed by authors is a basic attempt at developing a full scale service and does not qualify as a working software.

Researchers in [36] also developed a prototype that attempts to provide management tools for managing an enterprise fleet of mobile devices. The prototype connects mobile devices to a cloud service and provides a virtualized control of mobile devices. Berndt et. Al. in [37] devise a platform for providing mobile healthcare solutions as SaaS based cloud services. The proposed platform highlights the architecture of this system with no actual implementation of the system.

Danihelka and Kencl in [38] report a prototype build as a cloud service to provide 3D environments for collaboration between various entities using Microsoft Azure. The authors test the service with various complexities of the 3D model and report their results. Paper [40], reports a framework for synchronization of data between mobile devices and cloud storage. The authors devise a mechanism for propagation of data between nodes in a IaaS cloud service provider.

Authors in [39] comprehensively survey the mobile augmentation domain and presents taxonomy of Cloud based mobile augmentation approaches. The objectives of this study is to highlight the effects of remote resources on the quality and reliability of augmentation processes and discuss the challenges and opportunities of employing varied cloud-based resources in augmenting mobile devices. Authors propose architecture of the system with general discussion on challenges and future directions.

Paper [41] present an approach to standardization of function and requirement of mobile cloud services. Authors in [42] propose a framework to provide runtime support for the dynamic partitioning and execution of the application in the cloud. The proposed framework allows the dynamic partitioning for a single user but also supports the sharing of computation instances among multiple users in the cloud to achieve efficient utilization of the underlying cloud resources. Chieu et.al. [43] also design a similar prototype to share
resources using cloud computing. The proposed architecture has three tiers that support application services requiring manual configuration.

Yu and bin in [30] develop a cloud based application for music sharing among mobile devices. Users would be able to post music files to a cloud based repository which can be shared by subscribers to this service. The authors document details for the application including, design, architecture and implementation details. They compare the results against benchmarks defined by wireless communication standards such as 3G and WiFi. Unfortunately the paper does not address software engineering approaches to the development of these applications. No details are provided about the process, issues in design and development or post mortem analysis of the application.

Ferdiana in [45] reported developing a framework called Mobile Application Software Engineering Framework (MASEF). This framework addresses application development using agile practices and studies the adoption of these practices for four mobile applications. These applications are developed under four mobile platforms including Windows Phone, Android, Mobile Web and HTML 5. Although work reported documents successful adoption of agile practices with limited success in mobile computing environment, it fails to underscore the issues in mobile-cloud computing application development and data migration from a mobile device to cloud environment.

Despite all the recent advances in Mobile Cloud Computing, trust and privacy is still a major issue in MCC that hinder its vision. SAMI presented in [24] is an arbitrated multi-tier infrastructure model for mobile cloud computing leveraging Service Oriented Architecture. Authors in [25] present FocusDrive, a new mobile cloud data processing framework through trust management and private data isolation. Authors in [26] provide a trust model for offloading computation from mobile devices to a public cloud network. The main precursor for this research is improving mobile device energy efficiency. Zhang et. al. in [27] provide the concept of personal cloud network. They propose a novel approach of using hybrid cloud to distribute the computing among mobile devices, private cloud and public cloud. User data
is stored within the personal cloud framework where users have full, physical control. User can authorize public cloud services to access user-approved data.

2.3 Agile Software Development

Agile approaches are typically used in software development to help businesses respond to unpredictability [18]. It is a set of software development methods based on iterative and incremental development where customer should be involved throughout the development process [15]. Extreme Programming (XP) is by far the most popular of agile methods [20]. XP is flexible in a sense that early delivery of customer needed functionalities is acceptable reflecting changes late in life cycle. Developers get daily feedback when software is developed and tested in paired teams. All code is unit tested before release with acceptance tests running frequently.

Manuja and Manisha [19] discussed how to shift from Agile to Agile on Cloud, and discussed the challenges on both sides by developing a real-life application. Therefore, they provided Agile models for both Agile and Agile on Cloud. Furthermore, Manuja and Manisha applied the Scrum framework to align with CC ideally through AWS. Their focus was on addressing the CTQ challenges by implementing the Agile on Cloud model. The results provided for both models show indications of improvements with regards to CTQ. The Agile on Cloud model shows positive effects on cost and time dimension, whereas there is no effect on quality dimension because of security and privacy issues. Manuja and Manisha claimed that application is accessible on different devices (iOS, BlackBerry, Android, Win8) in addition to desktops and laptops. In contrast, the models figure 2.3 do not show the development and testing environment for these devices, and there appear to be no efforts related to developing/testing mobile applications.
Kalem et al. [16] described the relationship between Agile methods and CC. In addition, they described the benefits for the entire software development process. As a practical part of this paper, the researchers developed a warehouse management application by applying the DSDM Agile method. The cloud service provider for their project was the GAE platform. Kalem et al. compared development for Agile CC with the traditional Agile traditional method. Therefore, they developed a web application that does not use CC, and the same application using CC. The application had the same functionalities, but different environments. They employed ASP.NET, Microsoft Visual Studio as the Integrated Development Environment (IDE), and SQL Server for the normal application, and Eclipse when migrating to CC. Figure 2.4 shows a comparison of the development time in days for the applications.
The main goal of [17] was to make the MADONA methodology accessible to non-IT professionals. The primary goal of MADONA is to allow business stakeholders to develop automatically a cloud-business application simply by describing the business requirements via a web form. The MADONA methodology shows strength in the SaaS/PaaS/IaaS discovery phases, as presented in Figure 2.5. These phases are important because the organization needs to determine first whether the required services are available in the marketplace. They discover the SaaS service, if the required functions are not matched, the methodology proceeds with PaaS; if the required platforms are again not matched, the methodology continues with IaaS. After discovering all phases, and based on user needs, service ranking is calculated based on the coefficients associated to QoS attributes and assigned by the user based on priorities. The service with the highest rank is selected.

The authors claimed that MADONA is Agile because it allows business stakeholders to deploy another application if the one generated does not achieve the requirements. After examining the MADONA methodology, we can see that it offers automatic control service for stakeholders so that they can deploy another application if the original does not satisfy the requirements. To us, MADONA seems as service control, but not an Agile method. Agile includes iterative and incremental methods, so the process should pass through method phases i.e., from requirements, design, and development, to deployment.
Yu and Boskovic [18] attempted to understand the CC environment and how to apply it in the mobile software engineering field. They built a mobile application in streaming technology to explore the impact and effectiveness of CC in mobile software engineering. They provided the architecture, technologies, and process for the application they developed. Yu and Boskovic offered a brief description of the system architecture and design pattern. They adopted the three-tier client-server architecture and designed a pattern for the CMPlayer. They also specified the technologies used in each tier. In addition, they provided the software development process activity diagram that they applied. Furthermore, Yu and Boskovic discussed deployment into the GAE cloud, in addition to testing results, benchmarks, and performance.

In the development process presented in Figure 2.6, the research activities appear embedded in the process, but appear ignored in other processes. The processes highlighted in green are involved in cloud-based activities, either with GAE tools or through programming interfaces, which requires upfront research on various cloud-based infrastructures, platforms, and services.
2.4 Low Cost Cloud Computing Clusters

The Beowulf cluster created at Boise State University in 2013 [74] was perhaps the earliest attempt at creating a cluster consisting of multiple nodes of SBCs. It was built for collaboratively processing sensor data in a wireless sensor network. This cluster is composed of 32 Raspberry Pi Model B computers and offers an alternative in case if the main cluster is unavailable. This work documents the cluster construction process and provides information on the clusters performance and power consumption. The researchers present the compute performance of single RPI and an Intel Xeon III based server using MPI libraries running computation of the value of pi using Monte Carlo method. They first compare a single RPI against 32 RPIs organized in a cluster and report improvement of the speed up as well as decrease in the execution time. However when the RPI Cluster is compared to the Intel Xeon server, the Xeon server performs 30 times better in terms of execution time. They conclude that RPI Cluster provides platform for parallel processing in academic research and can be successfully constructed at a low cost.
The Bolzano Raspberry Pi cloud cluster experiment implemented a 300 node Pi cluster [51]. The main goal of this project was to study the process and challenges of building a Pi cluster on such a large scale. The researchers demonstrate how to setup and configure the hardware, the system and the software. It also presents how to monitor and maintain the system and utilize it as a cloud cluster. In their work, Abrahamsson et. al. presented applications of this cluster as a testbed for research in environment friendly, green computing. Furthermore, they also considered using this cluster to be deployed as a mobile data center. Although the focus of this work is on the design and deployment of the cluster using Raspberry Pi Computers, the work lacks detailed performance analysis of the cluster using popular performance benchmarks.

The Iridis-Pi project implemented a 64 node Raspberry Pi cluster [52]. Commonly known as the Lego super-computer, the work presents design and deployment of the raspberry pi cluster using lego blocks in a compact layout. They present a detailed analysis of performance in terms of execution time, network throughput as well as I/O read/write. The cluster compute performance is measured using the HPL Linpack benchmark which is popularly used to rank performance of super computers. The network performance was measured using a Message Passing Interface (MPI) to communicate between the Raspberry Pi’s. A detailed analysis of write speeds of various SD-Cards was also presented. Researchers argue that although the cluster cannot be used in conventional super computing environments due to its lacking performance, however the low cost, energy efficient, open source architecture, allows future academics and researchers to consider use of such clusters.

Tso et al. [53] built a small scale data center consisting of 56 RPi Model B boards. The Glasgow Raspberry Pi Cloud offers a cloud computing testbed including virtualization management tools. The primary purpose of this research was to build a low cost testbed for cloud computing resource management and virtualization research areas to overcome the limitations of simulation-based studies. The work compares the acquisition cost, electricity costs and cooling requirements of the cluster of single board computers with a testbed of 56 commodity hardware servers. Although the work presented provides a testbed for cloud
computing research, no further details are available on the performance comparison of this work.

Whitehorn [71] presented a Hadoop cluster composed of five Raspberry Pi Model B nodes. He noted that the performance of a Hadoop cluster is considered too slow due to limited memory capacity and poor data transfer rate. He concluded that this cluster is useful for academic research and educational purposes only.

In 2013, the developers of Cubieboards [72] single board computer presented a Hadoop cluster of Eight nodes. They compared the performance of Raspberry Pi Model A and Model B against the Cubieboard and concluded the Cubieboard is better suited for Hadoop deployment due to the faster CPU at 1 GHz as well as bigger main memory of 1 GB. The authors provide a complete step-by-step guide for deploying Hadoop on the cubie board platform for students and enthusiasts. They demonstrated use of wordcount program on a large 34 Gigabyte file obtained from Wikipedia. Although the demonstration shows deployment of the Hadoop cluster, the authors do not present any performance analysis results. Kaewkas and Srisuruk [73] at Suranaree University of Technology built a cluster of 22 Cubieboards running Hadoop and Apache Spark. They performed various tests studying the I/O performance and the power consumption of the cluster. They conclude that a 22 node cubie-board based cluster is enough to perform basic Big-Data operations with in an acceptable time.

In 2016, C. Baun in [69] presented the design of a cluster geared towards academic research and student scientific projects. They argue for the case of physical representation of the cloud infrastructure to the students which may not be accessible in a public cloud domain. They built a 8-node Raspberry Pi Model 2B cluster and study the performance aspects including computation time, I/O reads and writes as well as Network throughput. The researchers conclude that although the performance of their cluster is below par to traditional super computer clusters, it is however suitable for academic research because it provides better efficiency per watt for energy consumption. They argue that the reliability of their cluster is better compared to single server systems.
The low cost aspect of a SBC makes it attractive for students as well as researchers in academic environments. As pointed out in the literature, it is possible to deploy a Hadoop cluster using SBCs such as Raspberry Pi computers. Although the Raspberry Pi computers are cheap and widely available, the limitations in terms of processing power, available on-board memory and reliance on SD-Cards for external storage with slow I/O operations, yield performance with much to be desired. Thanks to increased interest in SBCs, newer single board computers with better design and faster operations speeds are becoming available. It remains to be seen how the improved SBCs perform when deployed in Hadoop clusters. In chapter 6 of this report, we present a detailed study on design and deployment of Hadoop on two SBC based clusters using Raspberry Pi Model 2 B as well as HardKernel Odroid Model Xu-4. The Odroid Xu-4 is a SBC with faster processor, larger on-board memory and faster I/O storage.
3. ShaMoCloud Service Architecture

This chapter provides requirement analysis and architecture for ShaMoCloud Service followed by implementation details.

3.1 What is ShaMoCloud?

As part of this work, there is a desire to study applications of the Agile process for CC and MCC works. The XP process was nominated for application in this study for developing the CC and MCC works. ShaMoCloud is a mobile and web-based CC service selected as a case study. While developing the ShaMoCloud case study by adopting XP processes, all the development aspects throughout the entire lifecycle were discussed and studied. Therefore, our enthusiasm focuses on exploring the software engineering approaches required to develop mobile and web-based applications in CC environments by developing ShaMoCloud as a case study for this research.

Planning, requirements, design, implementation, testing, integration, and deployment aspects are covered in the thesis work. Because the team tracked the work data from the start, the team experiences when developing the applications provided here can serve as guidelines for other teams.

ShaMoCloud is a cloud-based service that hosts web-based and mobile web services on the cloud. On the other hand, the mobile application (client) side was developed and hosted on an Android OS to consume and invoke the web services running on the cloud side. The infrastructure, business logic, and database for ShaMoCloud run on the cloud service.

3.2 Requirement Analysis for ShaMoCloud Service

The ShaMoCloud service is a cloud-based service that provides content management for media files, including images and videos. Users can access the service from a web interface or mobile application. Content accessibility and privacy is of utmost importance. ShaMoCloud is implemented in a public cloud environment for maximum accessibility, scalability, and user privacy.

The case study is designed and developed for open accessibility from various types of client devices. A web-based interface provides users with access to the service through any web browser. In addition, an Android-based application was developed to connect to the
service. Users need to register and set up an account with a two-step authentication process. Then, the users can upload media content, including images or videos, to their accounts. The media content can be shared with friends of the user. Any user can define the friends who might benefit from sharing and managing the uploaded media content. Friends can also be invited to register to the service, and in turn, share their media. Sharing permissions can be revoked by the user. The service allows resource scalability by allowing users to increase the repository size as well as integrate with other services in the future. Figure 3.1 shows the use case diagram for ShaMoCloud that overviews the functionalities of the application. We present the work and data flows for the ShaMoCloud below:

![Figure 3-1 ShaMoCloud use case diagram](image-url)
Add Friend Workflow

The process for adding friends in ShaMoCloud is presented in Figure 3.2.

*Figure 3-2 Add friend workflow*
Access Control Workflow

The access control process refers to how the user logs in and has ability to use the ShaMoCloud service. Figure 3.3 presents the workflow for this process.

Figure 3-3 Access control workflow
Video Upload/Download Workflow

Users can manipulate the videos set by the service. Figure 3.4 illustrates how users can upload, download, share, and delete videos.

Figure 3-4 Video services
Data Flow Diagram (DFD)

Figure 3.5 shows the DFD diagram for the ShaMoCloud development.

**Figure 3-5 ShaMoCloud Data Flow Diagram**
3.3 Architecture Details

ShaMoCloud is essentially designed as a three-tier architecture. Figure 3.6 illustrates an overview of the architecture. The three tiers are named based on their utility. User Services provides an interface for the user on the web or mobile devices. The middle tier, Cloud Services, implements classes for communication, media sharing, content management, etc. The third tier provides a database for storage. The following sections present details for these three layers.

3.3.1 User Service Layer

At the user level, the cloud service is accessible over the web and as an application on a smartphone. The devices connect to the second tier through web services. Native code on the Android hardware is written to communicate with the middle tier. The Android mobile application was developed to provide the ShaMoCloud service for sharing media and content management. Figure 3.7 shows the architecture of the Android application with alignment of Windows Communication Foundation (WCF) services with the middle tier and third layer. The following lists the components used in developing the user applications.

- New classes are written to implement Restful Web services in JSON format in order to provide Android and web accessibility. The purpose is to manage JSON requests and responses of the web service before/after calling the web service. Within this class, HTTP libraries are called to manage the various service requests.
- The AsyncTask class [49] allows proper and easy use of the User Interface (UI) thread in Android. It allows performing background operations and publishing results on the UI thread without having to manipulate threads and/or handlers. The AsyncTask class is modified for access to web service calls in order to manage the UI thread.
- XML is used for common functionality and to provide an interface for the Android native code.
Figure 3-6 ShaMoCloud architecture overview
Figure 3-7 Android application architecture and alignment with WCF
3.3.2 Cloud Services Layer

The second tier implements cloud and web services in the public cloud environment. User selected media content is made available in the ShaMoCloud database. In addition, most classes that implement the video and user services run as part of the cloud and web services. The case study is a cloud-based solution because it is hosted in a public cloud service provider. ShaMoCloud is hosted at Amazon EC2, part of the AWS. It utilizes in addition to hardware resources, a web server, and SQL database. The following lists the major components of this tier:

- **Web Interface**: this component is developed in ASP.NET to provide web accessibility to the cloud service. It provides video content management and sharing functionalities to users.

- **WCF Services**: this component manages all WCF services. It serves as an intermediate service for smartphone applications, and implements:
  - A SOAP protocol to support Restful Web services.
  - Interface for format compatibility with XML and JSON.
  - Exchange of complex messages, including streams of binary data.

In light of the application requirements, two major WCF services are defined in the solution. Each of these services may contain multiple components defined as functions based on the architecture:

- **User Services**: provide the user with account management, accessibility, access control, friend requests, and friend management. User service contains all the functions related to the user as an entity.

- **Video Services**: set of classes developed to cater to all services related to media management. These services provide video uploading, playing, deleting, editing, and streaming. In addition, management of media content and user provisioning for all content is provided. Video service contains all the functions related to the media posted by the user. This includes listing of media, uploading media, sharing, deleting, and editing media content.
3.3.3 Data Layer

The third tier contains the ShaMoCloud database hosted on the SQL server within the Windows Server instance that contains all media content and data. There is no way of communicating with the database layer except through the second layer (the cloud service layer).

The database is scalable to fit application requirements in case the data are small or huge. The scalability of the database is managed by the cloud service provider. The developer simply has to create databases, schemas, and tables over the SQL server, Oracle, or similar. Then, the developer can create the user name and password required for the applications that will be manipulated with the database.

In the future, if the number of users, data, and media are huge, the cloud-computing providers can offer services that allow users to scale the resources necessary to fit the requirements of target workload.
3.4 Implementation Details

This section presents the implementation details and technical aspects for ShaMoCloud with respect to the architecture of the proposed solution.

3.4.1 User Interface Layer

Because the ShaMoCloud service is accessible through smartphones and is web-based, two main technologies were used to develop the service, as explained in the following sections.

A. Web-based Interface

The interface for the web-based side was developed with the ASP.NET framework. In addition, in order to accelerate UI design and development and the make front-end development faster and easier, we nominated the Bootstrap framework as the technology that covers HTML, CSS, and JavaScript. Figure 3.8 shows the ShaMoCloud web-based interface. Appendix A shows the remaining screens for the ShaMoCloud web interface.

![ShaMoCloud web-based interface](image)

**Figure 3-8 ShaMoCloud web-based interface**

B. Mobile Applications

The programming language used to develop ShaMoCloud is Java. We also developed a corresponding mobile application for the Android smartphone using Android Studio, which is the official IDE for Android application development. The mobile application contains a set of Activities and Fragments considered as Java classes from the Java programming
perspective, but from the Android perspective, this is different. In Android development, “Activities” refers to those components that provide screens with which users can interact in order to accomplish a task. A “Fragment” is a piece of the application’s UI or behavior that can be placed in an Activity. Therefore, one Activity can obtain more than one Fragment, or none. For the ShaMoCloud mobile application, there is an Activity called “ShamocloudDefaultActivity” that hosts, or obtains, more than one Fragment: “LocalVideoActivity,” “CloudVideoActivity,” and “SharedVideoFragment.” All these three Fragments are tabs because one of the benefits of Fragments is their support for the tab feature. One can combine multiple Fragments into a single Activity in order to build a multi-pane UI, and reuse a Fragment in multiple Activities. Figure 3.9 shows the relationship between Activity and Fragment, as well as the home screen for the ShaMoCloud application. Appendix B presents some other screens from the ShaMoCloud mobile application.

Figure 3-9 Activity and Fragment relationship
There are various methods for developing and including Fragments within the Activity class in the Android environment. Figure 3.10 shows a sample of code developed for ShaMoCloud in order to set up the tabs within one Activity. The advantage is that these tabs/Fragments can be hosted in another Activity.

```java
public class ShamocloudDefaultActivity extends ActionBarActivity {

    private void setupTabs() {
        ActionBar actionBar = getSupportActionBar();
        actionBar.setNavigationMode(ActionBar.NAVIGATION_MODE_TABS);
        actionBar.setDisplayShowTitleEnabled(true);

        Tab tab1 = actionBar
            .newTab()
            .setText(R.string.title_activity_cloud_video)
            .setIcon(R.drawable.abc_ab_share_pack_holo_dark)
            .setTabListener(new SupportFragmentTabListener<CloudVideoActivity>(R.id.fragment1, this, "first", CloudVideoActivity.class));
        actionBar.addTab(tab1);
        actionBar.selectTab(tab1);

        Tab tab2 = actionBar
            .newTab()
            .setText(R.string.title_activity_local_video)
            .setIcon(R.drawable.abc_ab_share_pack_holo_dark)
            .setTabListener(new SupportFragmentTabListener<LocalVideoActivity>(R.id.fragment1, this, "second", LocalVideoActivity.class));
        actionBar.addTab(tab2);
        // actionBar.selectTab(tab2);

        Tab tab3 = actionBar
            .newTab()
            .setText(R.string.title_activity_shared_video)
            .setIcon(R.drawable.abc_ab_share_pack_holo_dark)
            .setTabListener(new SupportFragmentTabListener<SharedVideoFragment>(R.id.fragment1, this, "third", SharedVideoFragment.class));
        actionBar.addTab(tab3);
    }
}
```

**Figure 3-10 Hosting Fragment in Activity source code**

For each web service call, the scenario in Figure 3.11 occurred. This scenario contains several steps depending on developer requirements. AsyncTask allows proper and easy use
of the UI thread. This class allows performing background operations and publishing results to the UI thread without having to manipulate threads and/or handlers. AsyncTask must be “subclassed” in order to be used. The subclass overrides at least one method (doInBackground(Params...)), and most often overrides a second one (onPostExecute(Result)).

In ShaMoCloud uses almost all three main functions of this class, and classifies the steps based on the call of the functions described after Figure 3.11.

**Figure 3-11 Relationship between AsyncTask class, JSON parser class, and HTTP libraries**

- **onPreExecute() function**
  
  This function is used to declare the variables and run on the UI thread before executing the doInBackground() function, and thus the main goal is to initiate the thread. Step 1 in Figure 3.11 represents this function.

- **doInBackground() function**
  
  This method executes computations on the background and sends the results to the onPostExecute() function. Steps 2 to 5 in Figure 3.11 represent this function.

- **onPostExecute function**
  
  This function runs on the UI thread after doInBackground(). Steps 6 to 9 in Figure 3.11 represent this function.

More functions are used for the thread target, but ShaMoCloud used only the aforementioned functions. Figure 3.12 emphasizes the previous explanation by
demonstrating one of the services developed in ShaMoCloud, and how to connect to the cloud service via the web service.

```java
public class CloudVideoActivity extends ListFragment implements View.OnClickListener {
    final static String URL_DeleteVideo_Local = "http://localhost:3936/WCFServices/RESTfulServices/VideoService.svc/DeleteVideo?"
    private static final String TAG_GetOwnerVideoListResult = "GetOwnerVideoListResult";
    JSONParser jParser = new JSONParser();
    TextView videoId;
    TextView videoName;
    String CloudVideoID;
    String UserID;

    private class DeleteVideoService extends AsyncTask<String, String, JSONObject> {
        String OwnerID;
        String VideoID;

        private ProgressDialog pDialog;

        public DeleteVideoService(String UserID, String CloudVideoID) {
            OwnerID = UserID;
            VideoID = CloudVideoID;
        }

        @Override
        protected void onPreExecute() {
            super.onPreExecute();
            pDialog = new ProgressDialog(getActivity());
            //pDialog.setMessage("Getting Data ...");
            pDialog.setIndeterminate(false);
            pDialog.setCancelable(true);
            pDialog.show();
        }

        @Override
        protected JSONObject doInBackground(String... args) {
            // Building Parameters
            List<NameValuePair> params = new ArrayList<NameValuePair>();
            params.add(new BasicNameValuePair("OwnerId", OwnerID));
            params.add(new BasicNameValuePair("VideoID", VideoID));
            Log.d("request!", "starting" + params);
            // getting product details by making HTTP request
            JSONObject json = jParser.makeHttpRequest(URL_DeleteVideo_Local, "POST", params);
            // check your log for json response
            Log.d("Delete attempt", json.toString());
            return json;
        }

        protected void onPostExecute(JSONObject json) {
            pDialog.dismiss();
            try {
                // Getting JSON Array
                String result = json.getString("DeleteVideoResult");
                Log.d("Share Video Result", result);
                if (result.startsWith("ERROR")) {
                    Toast toast = Toast.makeText(getActivity(), result, Toast.LENGTH_LONG);
                    toast.show();
                } else {
                    Toast toast = Toast.makeText(getActivity(), result, Toast.LENGTH_LONG);
                    toast.show();
                }
            } catch (JSONException e) {
                e.printStackTrace();
            }
        }
    }
}
```

*Figure 3-12 Code for web service class with AsyncTask class default methods*
3.4.2 Setting up cloud web and data services

The cloud layer hosts the web services and database server for ShaMoCloud.

\textit{A Cloud web service}

This is implemented through the following steps:

1. Launch Amazon EC2 instances.
2. Set up web server.
3. Set up FTP platform.

The following sections explain each of the previous steps in detail.

1. Launch Amazon EC2 Instance

After subscription in AWS, Amazon grants credential details to users in order to allow them to log on to Management Console dashboard. Figure 3.13 shows the dashboard, which is a portal used by AWS users to launch and manage instances and other services, such as S3. For the ShaMoCloud service, the Amazon EC2 service is the option because it allows users to launch a virtual machine in the cloud.

We launched the \textit{Microsoft Windows Server 2008 R2 with SQL Server Standard} instance. The location of this instance is in Ireland because it is the closest Amazon data center to Saudi Arabia.
2. Set up Web Server

Because ShaMoCloud is based on the Windows Server instance and the ASP.NET programming language, the Internet Information Server (IIS) 7.0 is the web server used for ShaMoCloud. IIS hosts the website source code and the web services invoked by the mobile application.

The following is the public domain for ShaMoCloud: [http://ec2-54-76-29-61.eu-west-1.compute.amazonaws.com/](http://ec2-54-76-29-61.eu-west-1.compute.amazonaws.com/)

3. Set up FTP Platform

In order to transfer files between the local machine (PC) and the cloud instance, we need to set up software that supports the FTP protocol. FileZilla is one of the applications that support transferring files features. Such application consists of the FileZilla client installed in the local machine, and the FileZilla server installed in the cloud instance. Figure 3.14 shows the FileZilla software and how to transfer files directly to the cloud virtual machine.

Figure 3-13 AWS management console
B. Data Service Layer

The data layer acts as the database server for the ShaMoCloud service. The SQL server is installed on the Window Server instance. Then, the ShaMoCloud database is created with the credential details in order to allow the application to use the database. Subsequently, the Entity-Relationship Diagram (ERD) is created to present the relationships between the tables created. Finally, a script is written to execute the tables in the SQL server.

Figure 3.15 presents ERD for the current ShaMoCloud database. The following paragraphs illustrate the rules of each table in the ShaMoCloud service.

- **USERS Table:** responsible for the ShaMoCloud user details. Any user who logs on to ShaMoCloud can use the functions.
- **VIDEO Table:** stores all videos and their details.
- **TAGS Table:** used when users want to add tags to videos while uploading them. This table stores all the tags of each video.
- **SHARED_VIDEO Table**: utilized after a user shares a video with friends. Information regarding the video ID and the user-granted privileges to the video is stored in this table.

- **Friends_TBL Table**: all friend relationships are stored in this table.

- **REGISTERED_USER_FRIENDSHIP_REQUESTS Table**: when the user wants to add a friend, the system checks whether such friend is registered in ShaMoCloud. Therefore, if the friend is registered, the system sends a request to that friend by adding the request in this table.

- **UNREGISTERED_USER_FRIENDSHIP_REQUESTS Table**: when the user wants to add a friend, the system checks whether such friend is registered in ShaMoCloud. Therefore, if the friend is NOT registered, the system sends a request to that friend by adding the request in this table and sending an email notification to the friend.
Figure 3-15 ShaMoCloud Entity Relationship Diagram
4. Software Engineering aspects of ShaMoCloud

This chapter discusses a detailed study of the XP process applied as the software engineering process for the ShaMoCloud case study. In addition, it shows an overview of the software engineering aspects adapted through the process.

4.1 Project Planning

There are three levels in Agile planning [42] applied to the ShaMoCloud case study, as described in the following sections.

- First Level: Release Planning

A set of stories that contains the group of features to be released together. This is used to determine the number of required iterations. The output of this plan is a list of stories and completion date.

In Agile process, stories are used instead of requirement documents. The implementation time for each story receives one, two, or three-week estimates for "ideal development time."

The plan for releases is as follows:
1. Release (1): Set up cloud service and infrastructure.
4. Release (4): General enhancements

The releases were originally planned to be three, but we added one more for general enhancements.

- Second Level: Iteration\Sprint Plan

A set of development tasks and their estimates to complete these tasks by developer. This is used to assign tasks to developers. In the release plan, the customer should choose the user stories for this iteration. The implementation time for each task receives one, two, or three-day estimates for "ideal programming days."

- Third Level: Daily Planning
During daily meetings, the engineer attempts to outline the previous day’s accomplishments, those planned for the current day, and the problems that are causing delays.

### 4.2 Requirements and User Stories

User stories are employed for documenting the customer requirements as system features. The user stories are short, simple, and in a format of approximately three sentences written by customers using their own terminology without technical syntax.

The ShaMoCloud user stories are listed in Table 4.1. These stories have to be implemented from both the mobile application and website. Therefore, this means that each story should be considered as two stories.

<table>
<thead>
<tr>
<th>ID</th>
<th>User Story</th>
<th>Release Number</th>
<th>Delivery Date - Release (1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>As a normal user, I need an application to manage my videos and friends, so that I can access them from a web browser and mobile application. – Prepare cloud infrastructure.</td>
<td>Release 1</td>
<td>May 31, 2014</td>
</tr>
<tr>
<td>2</td>
<td>As a normal user, I can Register/Sign up, so that I can use available functionalities.</td>
<td>Release 2, Release 3</td>
<td>June 6, 2014</td>
</tr>
<tr>
<td>3</td>
<td>As a normal user, I want to log on to my account, so that I can access available functionalities.</td>
<td>Release 2, Release 3</td>
<td>June 8, 2014</td>
</tr>
<tr>
<td>4</td>
<td>As a normal user, I want to log out from my account.</td>
<td></td>
<td>June 8, 2014</td>
</tr>
</tbody>
</table>

**Table 4-1 ShaMoCloud user stories**

**Video Services**
|   | As a normal user, I want to see my list of videos, so that I can manage it. | June 17, 2014 | Local Video: Nov 07, 2014  
Cloud Video: Nov 28, 2014 |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>As a normal user, I want to upload my videos to my cloud account, so that I can store them in the cloud storage.</td>
<td>June 16, 2014</td>
<td>Nov 22, 2014</td>
</tr>
<tr>
<td>7</td>
<td>As a normal user, I want to download my videos from my cloud account, so that I can store them in my mobile device. (For mobile only)</td>
<td>Nov 30, 2014</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>As a normal user, I want to play my videos stored in my account, so that I can see the video. (For Website only)</td>
<td>June 19, 2014</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>As a normal user, I want to add tags to the video I uploaded to my account, so that I can filter and do searches later. (For Website only)</td>
<td>July 23, 2014</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>As a normal user, I want to delete my videos stored in my account, so that I can manage the video.</td>
<td>June 26, 2014</td>
<td>Dec 06, 2014</td>
</tr>
</tbody>
</table>

**Sharing Services**

|   | As a normal user, I want to share videos with friends, so that friends can also see video. | Aug 19, 2014 | Dec 06, 2014 |
|   | As a normal user, I want to un-share my videos, so that I can manage them. | Aug 19, 2014 | Dec 06, 2014 |
| 13 | As a normal user, I want to see a list of videos friends have shared with me, so that I can manage them. | Aug 19, 2014 | Dec 06, 2014 |

**Friendship Services**

<p>|   | As a normal user, I want to invite friends, so that I can share my videos. | Sept 06, 2014 | Dec 12, 2014 |</p>
<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Date</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>As a normal user, I want to accept invitation request, so that I can share my videos.</td>
<td>Sept 20, 2014</td>
<td>Dec 15, 2014</td>
</tr>
<tr>
<td>16</td>
<td>As a normal user, I want to ignore invitation requests.</td>
<td>Sept 20, 2014</td>
<td>Dec 18, 2014</td>
</tr>
<tr>
<td>17</td>
<td>As a normal user, I want to delete friends from my friends list.</td>
<td>Sept 20, 2014</td>
<td>Dec 20, 2014</td>
</tr>
<tr>
<td>18</td>
<td>As a normal user, I want to see a list of my friends, so that I can manage them.</td>
<td>Sept 06, 2014</td>
<td>Dec 10, 2014</td>
</tr>
</tbody>
</table>
4.3 Application Development

After completing the iteration planning, we can start assigning tasks to developers. The developer must track time and defects while developing a task. Table 4.2 presents the tracked data for the ShaMoCloud case study.

<table>
<thead>
<tr>
<th>Phases/Tasks</th>
<th>Agile XP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of code</td>
<td>7,186 LOC</td>
</tr>
<tr>
<td>Defects</td>
<td>Code 9 defects</td>
</tr>
<tr>
<td>Planning</td>
<td>3 days</td>
</tr>
<tr>
<td>Requirement</td>
<td></td>
</tr>
<tr>
<td>Design</td>
<td>1 day</td>
</tr>
<tr>
<td>Code &amp; Test</td>
<td>35 days</td>
</tr>
<tr>
<td>Research Effort</td>
<td>25 Days</td>
</tr>
<tr>
<td>Total days</td>
<td><strong>64 days</strong></td>
</tr>
</tbody>
</table>

Once the tasks have developed and tested by the developer, he/she pushes the changes to the ShaMoCloud repository and migrates the latest changes from the development to the cloud environment. The client can then start the acceptance test, and provide feedback during the weekly meetings. Subsequently, the next iteration/sprint starts with both new user stories and defects determined by the client. Thus, the developer fixes bugs and develops the new user story.

4.4 Application Testing

For the ShaMoCloud, the types of test applied are described in the following sections.

1. Integration Test

This refers to the test module, component, and web services that integrate the cloud with the client (mobile application). In general, when the web service and function are developed on the cloud side, they are tested with any client test tools, such as SoapUI\(^5\) or Fiddler\(^6\) in order to check the connectivity with the cloud and the results. Then, the developer from the

---

\(^5\) http://www.soapui.org/
\(^6\) http://www.telerik.com/fiddler
client side (the Android developer) can start development in order to integrate the mobile application with the cloud. The approach for the integration test adapted for the ShaMoCloud is as detailed below:

1.1 Test integration between the cloud functions and database (cloud layer and data layer).

1.2 Test exposed web services via the client test tools, such as Fiddler. When the web service is developed in cloud, it is tested through such tool, as shown in Figure 4.1.

1.3 Validate whether the data presented in UI or passed from UI is correct, and that it is as received from the cloud web service. To perform this type of test, there are log steps to help developers add code after/before critical lines so that the developer can monitor the runtime logs through the Android logging system “logcat” responsible for collecting and viewing the system debug output. The developers can monitor requests, web service response by adding steps, as shown in Figure 4.2.

1.4 Finally, validate all integration scenarios starting from UI and ending in the cloud or database layer.

2. Functional Test

This refers to the activities that validate the service functions, mobile web API, and UI. The functional test is managed by the development team, and does not include the client. In ShaMoCloud, after finishing the integration testing the development team test the whole scenario as per user story to check the functionality of the service before proceed to client acceptance test.

3. Acceptance Test

One of the goals of the user stories is to be prepared for the user acceptance test scenarios. Each story might have one or more acceptance tests. Such tests are a black box test, and thus the user enters information into system, and the output should be as expected in the written acceptance test. The thesis supervisor acted as the client for this work, and therefore, he performed the acceptance test. The development team met with the supervisor once or twice a week as development continued. At each meeting, feedback and the testing results were reported in order to fix bugs or proceed with the next function or iteration.
4. Interoperability Testing

This refers to those activities that check system interoperability across different devices, platforms, browsers, and wireless networks.
In order to perform the aforementioned types of testing, two approaches were followed for the ShaMoCloud. Such approaches are according to [50]:

1. Emulation-based Testing

   This testing involves the use of a mobile device emulator (called device simulator) that creates a virtual machine version of the mobile device on a local or personal computer. For Android mobile development, there is a tool called the Android AVD that provides a graphical UI where developers can create and manage the Android Virtual Devices (AVDs) required by the Android Emulator. Figure 4.3 shows an AVD running on a personal computer.

![Figure 4-3 Android emulator](image)

2. Device-based Testing

   The device-based testing approach requires setting up a testing laboratory and purchasing real mobile devices. With regard to the ShaMoCloud, application testing was performed with two real mobile devices.
Figure 4.4 Screen captured from the ShaMoCloud Android Application
4.5 Continuous Integration and Deployment

Deployment and version control are important aspects that need to be managed and planned early. All the source code for the ShaMoCloud service is stored in one repository to allow the team to commit and rollback such code, as well as track changes made to the code. Whenever a developer finishes a development task, he/she can commit the changes, and they are stored in the repository. Figure 4.5 shows the architecture for the deployment and version control framework.

![Figure 4.5 Changes deployment and source code management](image)

The following sections emphasize the framework adapted for ShaMoCloud:

1. **Set up Git System**

   Git is an open source and free version control system. It has a repository on a personal computer to store and commit source code from a programming editor to repository, such as Android Studio to Git system. When a developer finishes the tasks, he/she can push the code to a cloud account, such as Github or Bitbucket. For our case, we use the Bitbucket service.

2. **Set up Bitbucket**

   The ShaMoCloud team created an account in the Bitbucket service and launched a repository for ShaMoCloud. Bitbucket offers a source code repository that can be integrated with a code editor, such as Android Studio, Eclipse, or Visual Studio. Figure 4.6 shows the administrator page for the version control when the source code is pushed from the developer’s computer.
Figure 4.6 Bitbucket administrator page
5. Analysis of Development aspects and comparison with other works

This chapter’s focus is on the software engineering aspects of developing mobile cloud applications. To this end, the comparisons between the software development practices in four recent research publications are provided in the following sections.

5.1 Issues for Mobile Cloud Development in Recent Works

Kalem et al. [16] described the relationship between Agile development methods and CC. They first developed a warehouse management system. Later, the same application was developed and deployed as a cloud service in the GAE platform. The researchers discussed and contrasted both development approaches and concluded that cloud-based Agile development reduces cost in terms of development time. A criticism of their work is in the development environment, which changed completely when migrating from one project to another. It is not clear what effect the development environment would have had on the suggested approach. Another consideration is the learning curve for development. The same team that built the earlier application developed the cloud-based application, after already experiencing the difficulties involved in providing solutions. In addition, the web application was tested and deployed on web browsers accessible from smartphones, but not specifically on mobile applications that execute native code.

The authors in [17] provided MADONA, which is accessible to non-IT professionals. The primary goal of MADONA is to allow business stakeholders to develop automatically a cloud-business application simply by describing the business requirements via a web form. MADONA determines the business needs, and service ranking is calculated based on the coefficients associated to QoS attributes as assigned by the user based on his/her priorities. The researchers presented the benefits of using the Agile approach for developing MADONA. The approach used for selecting user requirements is Agile, but essentially, the development approach follows the traditional waterfall process, which may not be suitable for mobile cloud applications with quickly changing requirements.

Yu and Boskovic [18] discussed integration of the CC environment with mobile software engineering. They built a cloud service for streaming media content, including videos that could be played on the CMPlayer, a mobile cloud video-playing application. The cloud service
was built on the GAE framework. The authors concluded by providing recommendations and changes to the traditional software development processes. The authors faced challenges researching development solutions. Considerable time management was required for finding resources or developing additional resources for mobile applications.

Manuja and Manisha [19] discussed migrating from Agile to Agile on Cloud, and presented the challenges involved by developing a real-life application. They proposed a modified Scrum framework aligned with CC in an ideal way using AWS. They focused on addressing the CTQ challenges by implementing Agile on Cloud model. Their results compared traditional Scrum with the proposed Agile on Cloud model, which showed indications of improvement with regard to CTQ. The Agile on Cloud model showed positive effects in terms of cost and time dimension, but there was no effect on the quality dimension because of security and privacy issues. Furthermore, the model suggested was not tested for building mobile applications. It was assumed that all devices would access a web interface. No consideration was given to UI limitations when developing for mobile devices. Table 5.1 lists a summary of comparison for these research works.

**Table 5.1 Comparison of recent research works, strengths, and weaknesses**

<table>
<thead>
<tr>
<th>Paper</th>
<th>Methodologies</th>
<th>Technologies Used</th>
<th>Strengths</th>
<th>Weaknesses</th>
<th>Testing Environments</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSDM [16]</td>
<td>Agile—DSDM</td>
<td>For Communication: Skype, Google Docs, and Google Forms For Development: Without cloud: ASP.NET, SQL Server, Microsoft Visual Studio With cloud: GAE with Eclipse</td>
<td>Proposed method was applied to two projects and a comparison was made. Results clearly showed reduction in development time, testing, and deployment after using suggested approach.</td>
<td>Development environment changed completely when migrating from one project to another. Unclear what effect development environment would have on suggested approach.</td>
<td>Web application made accessible over public cloud. System was tested and deployed.</td>
</tr>
<tr>
<td>MADONA [17]</td>
<td>Agile—MADONA</td>
<td>For Requirements: RIVAL—Requirements VocAbU-Lary for describing business</td>
<td>Focused on stakeholder to develop a cloud-based business</td>
<td>Approach used for selecting user requirements is Agile, but essentially.</td>
<td>Web Application deployed over Amazon cloud.</td>
</tr>
<tr>
<td>Model</td>
<td>Description</td>
<td>For Development:</td>
<td>Cloud Provider:</td>
<td>Cloud Provider:</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------------------</td>
<td>----------------</td>
<td>----------------</td>
<td></td>
</tr>
<tr>
<td>CMPlayer [18]</td>
<td>Lightweight version of waterfall model</td>
<td>Android Application, Java development in Eclipse</td>
<td>GAE (PaaS)</td>
<td>Amazon EC2</td>
<td></td>
</tr>
<tr>
<td>ShaMoCloud</td>
<td>Agile XP</td>
<td>Android application, Java development in Android Studio, Website ASP.NET in Visual Studio, SQL—source code version control system</td>
<td>GAE (PaaS)</td>
<td>Amazon EC2</td>
<td></td>
</tr>
</tbody>
</table>

### 5.2 Gains in Developing ShaMoCloud Service

The team that developed ShaMoCloud had prior development experience using TSP [47]. TSP is a quality-focused process where defects are identified and prevented at an early stage of development. The stark contrast in software engineering practices when migrating from the TSP development approach to XP is clear. It can be argued that TSP is not suitable for a project with fast changing requirements, such as development projects in the MCC domain. The following describes the advantages and disadvantages of developing ShaMoCloud using the XP approach.

- The public cloud domain was used for the common storage area. Scalability and elasticity issues were managed easily. A configuration management system supported by the common cloud-based storage space was used by the distributed teams to share code and project updates. In addition, the web server and MS SQL database were utilized as part of the AWS EC2 instances.
• Version control for various codes was managed easily. CC facilitated solutions for managing teams, such as Trello⁷, and for source code version control, such as Bitbucket⁸ and Github⁹. Project setup was much speedier compared with TSP implementation. The shared development environment facilitated sharing and updating code for various classes and libraries online.

• Team and customer communication was managed easily using Skype for distant teams. The client was made aware of all aspects of development, and he provided input frequently as required by the process.

• There were no issues in testing and integration. Team members in the shared space shared all reports. The customer had access to all reports and provided input occasionally on various issues.

• Trust and privacy concerns are a big issue in the public cloud, and these were addressed by ShaMoCloud work. The proposed three-tier architecture allows users to manage the privacy of their media content since the data layer cannot be accessible except through cloud layer. In addition, the credentials details has given only for applications hosted on cloud layer so there is no channel to connect to data layer except by cloud layer with given credentials.

• The overall cost of development in terms of time and quality was much less compared with the team’s earlier data using TSP. The team also benefited from a shared environment to reduce the number of defects, thus improving software quality.

The ShaMoCloud work was completed in four releases. Table 5.2 shows the development effort in terms of days for all phases of the ShaMoCloud development. The team determined that the DSDM work [28] is the closest in terms of requirements and size compared with ShaMoCloud. It can be seen that most phases of work development were comparable with the DSDM work.

---

⁷ https://trello.com/
⁸ https://bitbucket.org/
⁹ https://github.com/
Table 5-2 Process time comparison between ShaMoCloud and DSDM

<table>
<thead>
<tr>
<th>Phases</th>
<th>DSDM</th>
<th>Phases</th>
<th>ShaMoCloud</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirement and Planning</td>
<td>3 days</td>
<td>Requirement and Planning</td>
<td>3 days</td>
</tr>
<tr>
<td>Testing Design and Development</td>
<td>32 days</td>
<td>Design, Coding, and Development</td>
<td>35 days</td>
</tr>
<tr>
<td>System Testing and Deployment</td>
<td>12 days</td>
<td>Integration test, functional test, acceptance test, interoperability test, and deployment</td>
<td>7 days</td>
</tr>
<tr>
<td>Research Effort</td>
<td>25 days</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

5.3 Issues with ShaMoCloud Service

There is quick-paced development using XP compared with TSP. As anticipated, Agile processes accelerate development. The following describes the issues encountered with the ShaMoCloud development case study.

- Frequently updating mobile and cloud libraries: at the start of the ShaMoCloud case study, the Android API version level was 19, but by the time the development was completed, Android API level 21 was released. The latest update required changes to the application we built. Such frequent changes provide new functionalities, whereas deprecating libraries and calls consequently require frequent updates to the applications.

- Research effort in developing: the team dedicated considerable time to researching open solutions. There are extremely few libraries available for development in the mobile cloud environment. Most libraries are available as open source, which typically requires modification and integration with existing code. Actually if the ShaMoCloud development team has prior experiences in the mobile and cloud computing areas when they started the development, the research effort will be less than the actual current effort since the team has no prior experiences in these areas.
- Project environment safety: during development, the team used tools in the public domain that can expose code, built data, configuration files, etc., thus leading to significant risks in terms of security and project confidentiality. From enterprise perspectives, the organizations try to avoid the source code version control public services by host local products inside the organization instead of use for example GitHub or Bitbucket.

5.4 Transition from TSP to Agile XP on Cloud

The ShaMoCloud team was involved in another work that applied the TSP process. In TSP, all the data were tracked and logged, similar to the Agile XP process. Table 5.3 presents data for both Agile and TSP processes. Both works were developed in an academic environment.

<table>
<thead>
<tr>
<th>Phases/Tasks</th>
<th>TSP</th>
<th>ShaMoCloud Agile XP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lines of Code</td>
<td>735 LOC</td>
<td>7,186 LOC</td>
</tr>
<tr>
<td>SRS</td>
<td>20</td>
<td>-</td>
</tr>
<tr>
<td>SDS</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>Test Plan</td>
<td>15</td>
<td>-</td>
</tr>
<tr>
<td>Code</td>
<td>14</td>
<td>9</td>
</tr>
<tr>
<td>Launch &amp; Strategy</td>
<td>8 days</td>
<td>3 days</td>
</tr>
<tr>
<td>Planning</td>
<td>8 days</td>
<td>3 days</td>
</tr>
<tr>
<td>Requirement</td>
<td>4 days</td>
<td>3 days</td>
</tr>
<tr>
<td>Design</td>
<td>3 days</td>
<td>1 day</td>
</tr>
<tr>
<td>Code &amp; Test</td>
<td>4 days</td>
<td>35 days</td>
</tr>
<tr>
<td>Research Effort</td>
<td>-</td>
<td>25 Days</td>
</tr>
<tr>
<td>Inspection</td>
<td>4 days</td>
<td>-</td>
</tr>
<tr>
<td>Total days</td>
<td>31 days</td>
<td>64 days</td>
</tr>
</tbody>
</table>

It was a difficult to compare between two different works adopting two different development processes. Since the ShaMoCloud development team was a part of TSP development team and the measurements data for both works and for same team, we can
compare between these different works by using LOC, number of defects and time to complete work.

Therefore, as noticed in the table 5.3, the size of XP work is greater than TSP work by ten times approximately. If total days of TSP is increased to be 64 days which is equal XP total days, TSP will deliver around 1500 LOC. That shows that the XP can deliver 7186 LOC/64 days whereas TSP can deliver 1500 LOC/64 days. This proves the fact that the Agile XP process can accelerate development and deliver product to customer faster than TSP.

Regarding the number of defects for TSP and XP, it shows that the TSP tries to find defects as can as possible even in the documentation which means the TSP focuses on quality more than Agile XP process.

5.5 Issues with ShaMoCloud Service

As anticipated, agile processes accelerate development. Following lists the issues with ShaMoCloud Development project phase 2.

- Frequently updating mobile and cloud libraries: At the start of ShaMoCloud project, the Android API level was 19 but by the time the Phase 2 was completed Android API level 21 was released. The latest update required changes to be made to the application that was built. These frequent changes provide new functionalities whereas deprecating libraries and calls consequently requiring frequent updates to the applications.

- Research Effort in developing: The team spent considerable time researching for open solutions. There are very few libraries available for development in mobile cloud environment. Most libraries are available as open source which typically require modification and integration with existing code.

- Data Privacy: Although ShaMoCloud addressed the issue of data privacy using private cloud for secure and data categorized by the user as private, the scalability issue on the private cloud layer would require investment. In order to solve this issue, the team is considering the concept of personal clouds presented in [27] where users would access their personal data from the public cloud using a virtualized device working as an overlay on their personal hardware.
• Project Environment Safety: During development, the team used tools in public domain which may expose code, built data, configuration files etc leading to huge risk in terms of security and project confidentiality.

Despite the above mentioned issues, the team is confident that the overall cost in terms of software size, time and quality was reduce by using cloud based extreme software development while working on the ShaMoCloud project.

In academics, students and researchers need to develop middleware and test cloud based healthcare applications. While subscription to cloud service providers could be expensive for some, deployment and access to private cloud infrastructure could be costly. It is important for students in academic environments to be exposed to real cloud computing infrastructure, visiting a cloud service provider's facilities is not enough for deeper understanding of underlying technologies. While universities and academic institutions provide hands-on experience in cloud based health informatics, essentially universities need to provide access to a suitable cloud computing infrastructure that can be used for experimentation, research, and teaching. Deploying the cloud infrastructure in universities could be a very costly effort. In this chapter we detail Low Cost Cloud Computing Cluster (LoC4) based on low cost and energy efficient Single board computers. The LoC4 cluster is extensively tested for performance using popular benchmarks. We investigate deployment of Hadoop on the cluster and provide configuration details for improving performance using DFSIO and TeraSort Hadoop benchmarks. Although the performance of the cluster is comparatively slow to a small scale data-center, nevertheless the low cost and affordable platform provides excellent opportunities for academic research in cloud based health informatics.

6.1 Motivation

Cloud computing provides storage for large scale data on external servers allowing researchers and developers to easily access the data [48]. Although the technology is being used in recent years with many benefits such as reducing the cost for maintaining servers whereas improving the availability of the systems, there have been wide spread privacy concerns for keeping personal data in the public domain.

In this chapter we consider a scenario for cloud based application where the privacy of users information as well as content is of much importance. We consider use of hybrid cloud environment which couples publicly available content on public cloud service provider's
infrastructure as well as more privacy sensitive information residing on a local data center. A very good case for such as applications is in the deployment of Personal Health Records (PHR) and Electronic Health Records (EHR) of patients and clients / healthcare providers in the Cloud. Healthcare applications providers and researchers face major challenges in integrating and effectively analyzing healthcare information. Electronic health record (EHR) and personal health record (PHR) systems are widely available and use different technologies and standards [46-47]. A majority of proprietary EHR and PHR systems implement a client server architecture where data is stored in local hardware and software using databases. The variety and size of medical health records data makes it difficult for researchers to accurately and easily integrate data from various sources. Traditionally healthcare providers have been using proprietary software for management of PHR information typically requiring large data centers for storage and retrieval of information [50]. Due to the high cost of building and maintaining specialized data centers many healthcare providers have been outsourcing PHR systems to third part cloud service providers such as Microsoft Health Vault, Google Health, General Electric’s Centricity Patient Online, eClinicalWorks and many more.

Since personal healthcare information can be exposed to unauthorized parties. Due to the sensitivity of personal information such as psychological diseases, fertility data, surgical procedures and emotional disorders among many others, the PHRs could potentially become targets of various malicious behaviors. The cloud service providers also may have significant commercial interest in harvesting and sharing patients’ PHR data with pharmaceutical advertisement companies, research institutions and/or insurance companies [49, 51]. To this end, recently researchers have proposed several cloud-based secure systems, especially cloud-based PHR systems using encryption where PHR data is encrypted and stored in cloud based storage. On the other hand, PHR are typically accessible by multiple data owners including patients, doctors, insurance companies etc., complexity of key management in encrypted PHR systems increases for the owners and users alike. To address these issues researchers in [49, 52-53] address these issues using hybrid cloud environment integrating the public and private cloud infrastructure where the encrypted data is stored in public cloud while the encryption keys and policies for multiple data owners are stored and executed in
the private cloud. Figure 6.1 shows framework for hybrid cloud deployment of a typical PHR/EHR systems for a Healthcare provider.

In academics, students and researchers need to develop middleware and test cloud based healthcare applications. While subscription to cloud service providers could be expensive for some, deployment and access to private cloud infrastructure could be costly. It is important for students in academic environments to be exposed to real cloud computing infrastructure, visiting a cloud service provider’s facilities is not enough for deeper understanding of underlying technologies. While universities and academic institutions provide hands-on experience in cloud based health informatics, essentially universities need to provide access to a suitable cloud computing infrastructure that can be used for experimentation, research, and teaching. Deploying the cloud infrastructure in universities could be a very costly effort. Most universities do not reveal the actual costs of establishing and running a datacenter, however knowledge from public domain reveals the cost of Ukko Cloud Computing Cluster at University of Helsinki was reported to be over 1 million Euros [58]. Expedient, a private cloud data center construction organization for small businesses provides installation cost of a tier III data center with 10 racks to be upwards of 1 million US Dollars [60]. Investment of this scale for many universities and educational institutions is out of reach.

In order to build a low cost effective cloud computing cluster researchers have investigated the use of single board computers (SBC). SBCs are inexpensive (cost: 40-80 US$), with onboard processors, RAM, Storage and Networking capabilities in addition of multitude of other features typically found in a personal computer. These computers are small sized and are capable of running a wide range of platforms such as Linux, Microsoft Windows, etc. In recent past, researchers have built clusters for high performance computing research using SBCs [61-64] and provide performance evaluation for compute task efficiency, memory utilization, network latency etc.
In this chapter we introduce LoC4, an affordable and low cost cloud computing cluster for health informatics research in universities and academic institutions. We provide architecture and design of the LoC4 cluster using Raspberry Pi Model 3B as well as HardKernel Odroid Xu-4 Single Board Computers. Both of these SBCs use ARM 32 based architecture for onboard processors and are equipped with removable memory /storage modules. We install these SBCs in a compact configuration for easy deployment and testing. The cluster deployment is tested using popular benchmark for performance evaluation of computation time, memory utilization, Network and I/O performance. Further to this we install the open source Hadoop platform on the LoC4 cluster tweaking various parameters for effective deployment and utilization of clusters resources. We also develop an application using Apache Hbase for storage of data in the NoSQL database. This application provides a middle ware implementation for connecting the LoC4 Cluster with public cloud domain in Amazon EC2. Our experience provides useful insights for students and researchers alike in building a low cost and affordable cloud computing cluster and providing a platform for developing and testing of Healthcare applications using Low cost cloud computing infrastructure.
6.2 The Single Board Computing Platforms

A single-board computer (SBC) is a complete computer built on a single circuit board [64]. A SBC incorporates microprocessor(s), memory, I/O as well as host of other features required by a functional computer. Single-board computers were designed to be used as demonstration systems for educational purposes. Recent advances in SoC Design has led to development of Advanced RISC Machine (ARM), a family of Reduced Instruction Set Computing (RISC) architectures for 32 bit and 64 bit computer processors. Currently ARM Cortex cores architecture is popular and widely used in smartphones, single board computers etc. Their use in educational environment has allowed non-profit organizations such as Raspberry Pi Foundation to encourage students at school levels to learn about programming and applications development. A wide range of programming languages are supported by these platforms. In this work we investigate use of Raspberry Pi Model 3B as well as HardKernel Odroid Xu-4 single board computers.

The Raspberry Pi Foundation [65], is a non profit organization that developed a credit card-sized single-board computer called Raspberry Pi (RPi) [66]. The first generation RPi was released in February 2012. Since then newer models such as A+, B+ and later the second generation Model 2B were released improving the previous development platform by increased processor speed, larger onboard memory size as well as newly added features. The latest RPi Model 3B was released in February 2016. Figure 6.2(a) shows the RPi Model 3B. Table 6.1 provides details of various on board features. The RPi Model 3B was released with upgraded feature set including 1.2 GHz 64-bit ARM-Cortex A53 quad core processor, 512 KB shared L2 cache with improved task threading and instruction set. Benchmarks showed the Raspberry Pi 3 to be approximately 80% faster than the Raspberry Pi 2 in parallelized tasks [67]. RPI is compatible with various platforms such as Microsoft Windows 10 IoT Core, Ubuntu MATE, RISC OS etc. Other operating systems are also available for RPI including Android, Arch Linux ARM, Firebox OS, Google Chromium, Fedora Plus and Unix. The RPI foundation provides a Debian based Linux distribution for download called Raspbian. It promotes Python and Scratch as their main programming languages with support for many other programming languages.
Odroid is a series of single board computers and tablet computers created by HardKernel Co. Ltd. [68], an open source company located in South Korea. Two models of these computers in classes C and XU are available. ODROID-XU-4 [69] is a newer generation of single board computers offered by HardKernel. Xu-4 board can run various flavors of Linux, including

![Figure 6-2 (a) Raspberry Pi Model 3B (b) HardKernel Odroid Xu-4](image)

<table>
<thead>
<tr>
<th>Table 6-1 Comparison of Raspberry Pi 3B and HardKernel Odroid Computers</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Raspberry Pi Model 3B</strong> [67]</td>
</tr>
<tr>
<td><strong>Processor (CPU)</strong></td>
</tr>
<tr>
<td><strong>GPU</strong></td>
</tr>
<tr>
<td><strong>Onboard RAM</strong></td>
</tr>
<tr>
<td><strong>Ethernet / Network</strong></td>
</tr>
<tr>
<td><strong>Storage</strong></td>
</tr>
<tr>
<td><strong>Audio/Video</strong></td>
</tr>
<tr>
<td><strong>Power Consumption</strong></td>
</tr>
<tr>
<td><strong>USB Ports</strong></td>
</tr>
<tr>
<td><strong>Released</strong></td>
</tr>
<tr>
<td><strong>Price at release</strong></td>
</tr>
</tbody>
</table>

(US$)
Ubuntu 15.04, Ubuntu MATE etc. XU-4 uses Samsung Exynos5 Quad core ARM Cortex™-A15 Quad 2Ghz and Cortex™-A7 Quad 1.3GHz CPUs with 2Gbyte LPDDR3 RAM at 933MHz. The Mali-T628 MP6 GPU supports OpenGL 3.0 with 1080p resolution via standard HDMI connector. Two USB 3.0 ports as well as a USB 2.0 port allows faster communication with attached devices. The board demands 4.0 amps power supply with power consumption of 2.5 Watts (idle) and 4.5 Watts (under load). By implementing the eMMC 5.0, the ODROID C1 and XU-4 boasts improved I/O transfer speeds over Class 10 SD card flash memories.

In summary, in terms of price, Odroid XU-4 priced at $79, is expensive compared to the $35 Raspberry Pi 3B, nevertheless the improved processing power, although demanding more power, provides tradeoff with improved performance, task execution time as well as better I/O read and write operations.

6.3 The Low Cost Cloud Computing Cluster (LoC4)

This section presents the architecture and configuration of the LoC4 cluster deployed in this experimental study.

6.3.1 Design of LoC4 Cluster

The LoC4 cluster is composed 11 SBCs of while five RPi Model 3B and six Odorid Xu-4 computers are interconnected with power supplies, network cables, Storage modules, connectors and cases. All the Raspberry Pi computers are equipped with 16 GB Class-10 SD cards for primary bootable storage. The Odroid Xu-4 devices are equipped with 32GB eMMCv5.0 modules. The Ordoid Xu-4s are housed in a compact layout racks using M2/M3 spacers, nuts and screws. The Raspberry Pi 3B’s are housed in cases and connected to the rest of the cluster. Currently each Raspberry Pi computer is individually supplied by 2.5Amp power supply; each Odroid Xu-4 computer is supplied by a 4.0Amp power supply that provides ample power for running each node.

Each SBC’s network interface is connected with a Cat6e Ethernet cable through the Rj-45 Ethernet connector. All Ethernet cables connect to a 16-port Cisco switch which in turn is connected to the university network equipment. The size of the cluster can be easily scaled.
by introducing a core switch that connects to the 16-port switch. Additional nodes can be included in the network configuration scaling the size of the cluster.

For Hadoop deployment, one of the Xu-4 SBC is used as the master node whereas the rest of the nodes server as slaves nodes. We compare the performance of Xu-4 sub cluster (composed of five Xu-4 nodes) with RPi sub cluster also composed of five RPi 3B nodes. Figure 3 shows the layout of the LoC4 cluster. Figure 3 shows the LoC4 cluster.

The purchase cost for all equipment for the LoC4 Cluster was $1179.

### 6.3.2 Hadoop Deployment

Apache Hadoop is an open source framework that provides distributed processing of large amounts of data in a datacenter. The Hadoop framework scales well for thousands of machines allowing processing of peta-bytes of data. Hadoop uses the map/reduce programming model for Big Data processing over multiple nodes. The map/reduce model is composed of two steps, the map step performs filtering and sorting of data, the reduce step provide further processing of data from map step usually summarizing the outcomes. Hadoop 2 introduced Yet Another Resource Negotiator (YARN) as a new resource management layer allowing for better resource management and monitoring. A basic installation of Hadoop consists of 1) HDFS, The Hadoop Distributed File System that keeps track of files distribution and replication; 2) YARN, provides dispatch, monitoring and execution of all map/reduce tasks in slave nodes and 3) JobHistoryServer keeps track of the completed jobs and maintains their logs.

Hadoop version 2.6.2 was installed due to availability of YARN daemon which improves the performance of the map-reduce jobs in the cluster. To optimize the performance of these Clusters, yarn-site.xml and Mapred-site.xml were configured with 512 MB of resource size allocation. The primary reason for this setting is the limited amount of RAM available in the RPi Model 3B. The Raspbian operating system as well as the shared GPU memory bus consume over 200MB or RAM out of a total of 1GB. The default container size on the Hadoop Distributed File System (HDFS) is 128 MB. Each SBC node was assigned a static IPv4 address.
based on the configuration and all slave nodes were registered in the Master node. Table 6.2 provides detail of important configuration properties for the Hadoop environment. For YARN Resource manager, we allocated up to 4 cores which means up to 4 containers can execute per node (one container per core). The replication factor for HDFS is 2 which means only two copies of each block would be kept on the file system. The LoC4 Cluster was tested extensively for performance using two sets of benchmarks. Further details are provided in the next section.
Figure 6-3 The LoC4 Cluster with 6 Xu4 (Rack mounted) and 5 RPi 3B in cases
Table 6-2 Hadoop configuration properties for LoC4 Cluster

<table>
<thead>
<tr>
<th>mapred-site.xml Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>yarn.app.mapreduce.am.resource.mb</td>
<td>512</td>
</tr>
<tr>
<td>mapreduce.map.cpu.vcores</td>
<td>1</td>
</tr>
<tr>
<td>mapreduce.reduce.cpu.vcores</td>
<td>1</td>
</tr>
<tr>
<td>mapreduce.map.memory.mb</td>
<td>512</td>
</tr>
<tr>
<td>mapreduce.reduce.memory.mb</td>
<td>512</td>
</tr>
<tr>
<td>mapreduce.input.fileinputformat.split.minsize</td>
<td>8 MB</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>YARN-site.xml Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>yarn.nodemanager.resource.memory-mb</td>
<td>1024</td>
</tr>
<tr>
<td>yarn.nodemanager.resource.cpu-vcores</td>
<td>1</td>
</tr>
<tr>
<td>yarn.scheduler.minimum-allocation-mb</td>
<td>256</td>
</tr>
<tr>
<td>yarn.scheduler.maximum-allocation-mb</td>
<td>1024</td>
</tr>
<tr>
<td>yarn.scheduler.minimum-allocation-vcores</td>
<td>1</td>
</tr>
<tr>
<td>yarn.scheduler.maximum-allocation-vcores</td>
<td>4</td>
</tr>
<tr>
<td>yarn.nodemanager.vmem-pmem-ratio</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>hdfs-site.xml Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>dfs.replication</td>
<td>2</td>
</tr>
</tbody>
</table>

6.4 Performance Evaluation

In this section, we present a performance evaluation study of LoC4 cluster in terms of processing speed, storage read/write and networking. The objective of these studies is to highlight the intrinsic capabilities of SBCs as well as to showcase the performance of these SBC when deployed as a cluster. We use popular benchmarking techniques widely used in the literature to highlight the capabilities of these SBC.

6.4.1 Performance Benchmarking

A. CPU Execution Times
The benchmark suite Sysbench\textsuperscript{10} was used to measure the CPU performance. Sysbench provides benchmarking capabilities for Linux and supports testing CPU, memory, File I/O, mutex performance in clusters. We execute the Sysbench benchmark\textsuperscript{11}, testing each number up to value 10,000 if it is a prime number for \( n \) number of threads. Since each computer has a quad core processor we run the sysbench cpu test for 1, 2, 4, 8 and 16 threads. We measure the performance for this benchmark test for Raspberry Pi Model 2B, Odroid Xu-4 as well as a Intel i7 3.0 GHz 4th Generation personal computer (for comparison purposes).

The CPU execution times scale well with increased number of threads. Sysbench test runs with \( n=2 \) and \( n=4 \) threads significantly improves the execution times performance for all processors by 50%. With \( n=8 \) and \( n=16 \) threads the test results yields almost similar execution times with little improvement in performance. The execution times for Odroid Xu-4 are 6 times better as compared to Raspberry Pi Model 3B. The increased number of threads (larger than 8) does not provide gain in performance of Odroid Xu-4 over Raspberry Pi, furthermore, the execution time for Raspberry Pi is further extended with larger \( n \). Table 3 shows the average CPU execution time for nodes with \( n \) threads.

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|c|c|}
\hline
 & CPU Cores & Clock Rate GHz & 1 & 2 & 4 & 8 & 16 \\
\hline
Raspberry Pi 3B & 4 & 1.2 & 382.2 & 192.07 & 96.31 & 98.69 & 99.73 \\
Odroid Xu-4 & 8 & 2.0 and 1.4 & 83.38 & 41.68 & 25.336 & 17.66 & 18.02 \\
Intel i7 4th Gen & 4 & 3.0 & 8.508 & 4.272 & 2.228 & 2.27 & 2.23 \\
\hline
\end{tabular}
\caption{Average CPU Execution Time for nodes with \( n \) threads}
\end{table}

\textsuperscript{10} https://wiki.gentoo.org/wiki/Sysbench

\textsuperscript{11} Using \texttt{sysbench --test=cpu --cpu-max-prime=10000 --num-threads=n run}
Table 6-4 Read and Write Throughput (KB/sec) for individual devices in the clusters using FIO’.

<table>
<thead>
<tr>
<th>Device Description</th>
<th>Read Throughput (KB/second)</th>
<th>Write Throughput (KB/second)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Buffered</td>
<td>Non-Buffered</td>
</tr>
<tr>
<td>Raspberry Pi 3B with 16 GB Class 10 SanDisk SD Card</td>
<td>3,688</td>
<td>2,341</td>
</tr>
<tr>
<td>Odroid Xu-4 with 32GB eMMCv5.0 Module</td>
<td>9,197</td>
<td>6,883</td>
</tr>
</tbody>
</table>

Figure 6-4 NetPIPE benchmark results for Xu4 and RPi clusters considering latencies with data size in terms of bytes on the x-axis.
Figure 6-5 NetPIPE benchmark results for Xu4 and RPi clusters considering bandwidth with data size in terms of bytes on the x-axis.

B. Storage Performance

Poor storage read/write performance can be a bottleneck in clusters. Compared to a server machines, an SBC is handicapped in terms of availability of limited storage options. SBCs are typically restricted to external storage connected through the USB interface with bootable flash disks or SD-Cards are primary storage devices. In our experimentation, the Raspberry Pi's were equipped with 16GB SanDisk Class 10 SD Cards, whereas the XU-4 devices were equipped with 32GB eMMC memory cards. Both of these memory cards were loaded with bootable Linux distributions.

FIO \(^\text{12}\) was used for benchmarking of random read and write with various block sizes. We consider the throughput with 8 threads each working with a file of size 512MB with a total 4GB of data. These parameters were set specifically to avoid buffering and caching in RAM issues which is managed by the underlying operating systems that can distort the results; i.e. the data size (4GB) selected is larger than the onboard RAM available on these devices. Table

\(^\text{12}\) https://www.openhub.net/p/fio
4 shows the comparison of buffered and non-buffered random read and write from both devices with block size 8KB.

As can be seen from Table 6.4, the read throughput (buffered) of Odroid with eMMC memory is at least twice as fast as the Class 10 SDCard on the Raspberry Pi whereas the non-buffered read is more than three times better. Similarly for buffered write operations, Odroid Xu-4 with eMMC module throughput is more than twice better when compared to the Class 10 SDCard in Raspberry Pi.

C. Network Performance
When data are being processed in a cluster, servers need to transfer data with a certain amount of network bandwidth for the data to be delivered quickly and processed efficiently. The network performance was measured using the popular Linux based command line tool iperf 13v3.13 with the NetPIPE 14benchmark version 3.7.2. Through various sets of runs, iperf states the network throughput to be 82-88Mbits per second for the RPi as well as the Xu-4 SBCs. The NPtcp, NetPIPE benchmark using TCP protocol, involves running transmitter and receiver on two nodes in the cluster. In our experimentation, we executed the receiver on the LoC4 master node (Xu-4 device) with 1000 KB as maximum transmission buffer size for a period of 240 milliseconds. The transmitter was executed on the individual SBCs one by one. As can be seen from Figure 6.5, the network latency for all clusters with small payload is almost similar. As the payload increases we observe slight increase in network latency between the three clusters. On the other hand we observe a spike in throughput at message size 1000 bytes, this indicates that the smaller a message is, the more is the transfer time dominated by the communication layer overhead. For larger messages, the communication rate becomes bandwidth limited by a component in the communication subsystem that may include the data rate at the network link, utilization of the communication medium at the time or the traffic on the network switch. Contrasting the performance of Xu-4 and RPi SBCs we note the visible difference in throughput between the two, this is due to the poor overall Ethernet performance of the Raspberry Pi probably caused by design.

---

13 https://iperf.fr/
14 http://bitspjoule.org/netpipe/
Figure 6-6 TestDFSIO Read and Write Time for RPi and Xu4 sub-clusters in LoC4 Cluster.
6.4.2 Hadoop Benchmarks

Hadoop provides various benchmarks for performance of the Hadoop cluster. Among these, in this study we utilize TestDFSIO and TeraSort to study the performance of Execution time for tasks and read / write speeds of various tasks.

A. The TestDFSIO Benchmark

TestDFSIO [70] is a HDFS benchmark included in all major Hadoop distributions. TestDFSIO is designed to stress test the storage I/O (read and write) capabilities of a Hadoop cluster. TestDFSIO creates n mappers for n number of files to be created and read subsequently in parallel. The reduce tasks collect and summarize the performance values. The test provides I/O performance information by writing a set of files of a fixed size to HDFS and subsequently reading these files while measuring Average I/O rate (MB/second), throughput (MB/second) and execution time (seconds) for the job. Since TestDFSIO requires files to be written first before they can be read, we run experiments to write 10 files of varying sizes for each experiment on all clusters. Figure 6.6 shows the execution time of the TestDFSIO write benchmark with 10 files of sizes 5, 10 and 20 GB. We observe that the execution time for the RPI sub-cluster increases by 50% for file sizes 5GB and larger. The results for RPi sub-cluster is correlating with Xu-4 sub-cluster although the execution time is less than 50% comparatively.

The read test reads the output files written to the HDFS by the previous test and observes execution time, throughput and average I/O rate. We measure the results using 10 files of sizes 5, 10 and 20 GB. We observe that the read performance of Xu-4 sub-cluster is on average 1.89 times better compared to RPi devices. The performance degrades as the file size increases for LoC4 cluster. In contrast to these results we observe that the read throughput for RPi sub-cluster decreases significantly.

B. TeraSort Benchmark

TeraSort combines testing the HDFS and MapReduce layers of a Hadoop cluster and consists of three MapReduce programs, TeraGen, TeraSort and TeraValidate. TeraGen is typically used to generate large amounts of data blocks, consequently, TeraGen is a write intensive
I/O benchmark. TeraSort generates set of sample keys by sampling the input data generated by TeraGen before the job is submitted and writes the list of keys into HDFS. The input and output format, which are used by all three MapReduce programs, reads and writes the text files in the correct format. The TeraSort benchmark is CPU bound during the map phase and I/O bound during the reduce phase. TeraValidate verifies that the output of the TeraSort is globally sorted.

![Graph of TeraGen and TeraSort execution time for LoC4 cluster.](image)

Figure 6-7 TeraGen and TeraSort execution time for LoC4 cluster.

We run TeraGen, and TeraSort on LoC4 cluster with 400MB, 800MB, 1,600MB and 3,200MB data sizes. We observe the job execution time for each run for comparison and analyze the
performance on the cluster. Since TeraGen is I/O intensive, the write speeds of the memories/storage in corresponding nodes in the clusters plays a major role in degrading the overall job completion time. The average job completion times for RPi and Xu-4 sub-clusters can be seen in figure 6.7. The completion time for TeraGen on Xu-4 and RPi is correlating for various data sizes with Xu-4 being 3 times faster compared to RPi. We compute the execution time for TeraSort. For all experiments, we use the same number of map and reduce tasks on each cluster. The TeraSort benchmark is CPU bound during the map phase, i.e. reading input files and sorting tasks are carried out whereas it is I/O bound during the reduce phase, i.e. writing output files in the HDFS. We observed that 33-40% of job completion time occurred in map phase while 49% or more time was spend in reduce tasks overall for the majority of TeraSort jobs run on the cluster. We observe that as the data size increases the execution time also increases for RPi. Figure 6.7 also shows the comparison of the ratio of execution time degradations for RPi against Xu-4 sub-cluster. For smaller data size of 400MB the Xu-4 is 3 times faster whereas for larger data size of 3.2 GB, the Xu-4 is 3.4 times faster.
7. Conclusions and Future work

7.1 Concluding Remarks

The motivation behind this work was to document experiences in process migration from Team Software Process to Extreme Programming in an academic environment. The goal was achieved by designing and developing a ShaMoCloud framework that integrates public and private cloud infrastructures for easy accessibility and scalability for users. To this end we designed and developed web and mobile application to implement and test the ShaMoCloud framework for content/media management. The design and implementation of a cloud environment was detailed with a focus on integrating the three tiers of the service, while preserving data security and privacy. Various aspects of development using XP as an Agile development methodology were discussed. The work was compared with similar works that highlighted the gains and challenges in development. Our experience showed that mobile–hybrid cloud application development can be integrated with Agile development methodologies to reduce cost and time, and improve software quality. Furthermore, the accelerated pace of development using Agile methods is suitable to the fast changing pace of newer mobile and cloud technologies. Our results showed positive indications to applying XP for both mobile and web applications integrated with the public cloud.

The work was built in an academic environment, and adopted the XP process to fit user requirements. The nomination of the XP process was based on the needs of the programming effort required for such mobile and Cloud Computing works. The research efforts were included as part of the work implicitly, in addition to the training on new technologies, because the team was not familiar with mobile applications and Cloud Computing technologies. Four releases were launched to complete the work as applications for the framework. Each application were developed by native programming languages, so that the web application was based on ASP.NET, and the mobile application was based on the Android environment. The team attempted to adapt the XP process to fit needs, and thus not all practices of the XP process were applied in the work. The time, defects, and LOC were tracked during development in order to calculate complexity and perform a comparison
study with the TSP process. The testing tasks were considered within the development tasks in order to ensure that the developer tested the code immediately after development. The integration test, functional test, and interoperability were completed during development, whereas the acceptance test was executed by the client after each iteration/sprint. Contiguous integration and version control helped the team perform tasks properly by pushing changes to the source code repository in the cloud service. This provided the team with confidence to perform changes to the source code because backup existed in the cloud repository, and developers were able to roll back code changes.

The comparison analysis with recent similar works focused on three main points. First, we considered the development process types for those works. The work by Manuja and Manisha and DSDM applied Agile methodologies, whereas the remaining works applied the waterfall process. Second, the CC environment for all was public CC, and thus the work by Manuja and Manisha and MADONA were based on the Amazon EC2 service, whereas DSDM and CMPlayer were based on GAE. Third, the applications environment was web-based for all, with the exception of CMPlayer, which was developed for mobile devices. The proposed solution ShaMoCloud was developed to serve for both web and mobile applications by adapting the Agile XP process based on the Amazon EC2 public cloud provider. From the comparisons between Agile XP and TSP processes, we found that there is a stark contrast between software engineering practices when migrating from the TSP development approach to XP because we delivered 7180 LOC/64 days by followed XP but with TSP we delivered 1500 LOC/64 days. It can be argued that TSP is not suitable for a project with fast changing requirements, such as development projects in the Mobile-Hybrid Cloud Computing domain, whereas XP shows positive indications for the MCC domain.

In this work we also investigate the use of SBC in a low cost cloud computing cluster. We consider two kinds of popular platforms Raspberry Pi 3B and Odroid Xu-4 using ARM Cortex Processors. The LoC4 cluster was deployed comprising of 11 SBCs interconnected in a network topology over a gigabit Ethernet. Hadoop was installed on the cluster with configuration to suit the SBC’s memory and storage requirements. Extensive performance evaluation was carried out for both platforms using various performance benchmarking
tests for task execution time, I/O read/write and network performance. Further to this, we also conducted Hadoop benchmarks TestDFSIO and TeraSort for performance evaluation of the Hadoop deployment in the cluster. Results show that Hadoop with Hbase can be successfully deployed on a cluster built using low cost SBCs. In terms of compute power comparison with a traditional server, the limited onboard resources of these SBC yield very poor comparative performance. Nevertheless, the LoC4 Hadoop cluster deployment was successful and our experience provides excellent opportunities for academic research in cloud based health informatics. The LoC4 cluster is made available for students to test their applications in the SE409 Cloud Computing course taught at Prince Sultan University.

### 7.2 Future Work

Since this project encompassed various overlapping technologies and coupling of software engineering processes in development for these technologies, there are various areas open for further research.

Deployment of Hadoop cluster was done using Single Board computers which lack compute power and are much slower compared to regular Personal computers. Nonetheless, the environment provides testbed for building and testing cloud applications. The research in this area opens up horizons in green computing where energy aware applications can be built for energy hungry data centers to reduce the impact on the environment. The use of Single Board Computers is increasing whereas the cost of building these computers is decreasing, the coupling of these effects makes it possible to increase the compute power on these computers so they can be used in building cloud computing infrastructure.

Currently, although cloud service providers provide SaaS, PaaS and IaaS services to users, this does not satisfy enterprises where personal and sensitive data of individuals, companies and governments are kept in public space due to trust, privacy, and security issues. While large companies such as Microsoft, Google etc provide cloud based software such as Microsoft Health Vault, Google Health, General Electric’s Centricity Patient Online, eClinicalWorks and many more, although these systems are good options for small to
medium sized enterprises, they still pose the privacy challenges. The cloud service providers may have significant commercial interest in harvesting and sharing data with commercial enterprises and research institutions. This requires establishment of Hybrid Cloud computing infrastructure where privacy sensitive data is kept on premises of the enterprise whereas the other content is placed on public cloud. While ShaMoCloud provided a prototype of such as system, much work and further investigation is needed in this area.

Development of mobile applications for different environments, such as iOS, Windows Phone, or Blackberry, and also for tablets or iPads could be challenging. ShaMoCloud provided web-service service oriented architecture which can be ported and accessed via web browsers. Much work is being done in this area and new libraries are continuously being developed.

### 7.3 Recommendations

As we worked on this research project, we observed that there are various promising areas for further research in both cloud computing services as well as mobile development platforms. Considering these advances and their limitations we provide the following recommendations to the Software Engineering community at large.

Exploiting the vast scale of benefits of public cloud services considering the widening availability and support from cloud services providers as well as the enormous applications of scalability and elasticity features, almost all major cloud service providers if not already utilizing the global data centers, are developing data centers around world resulting in increased connectivity efficiency. This enormous amounts of available resources would help in accessibility and availability of cloud service easily anywhere with an acceptable level of quality.

Adapting the professional tools and products in development environment will help a Software development team in academic environment to manage the tasks easily. Version control of source code is an important issue; It can be adapted into practice using various
development environments available in the Cloud so that the developers can have full control over the source code. There are a lots of open source and free solutions that help in managing the team activities and task assignment using software such as Trello etc. While Open source tools are freely available and were widely used in this project, professional tools provide better quality and management.

Testing is an important phase in any kind of adopted Software Process. We recommend using extreme programming (XP) for development of Mobile applications and integrating them with Cloud Computing technologies in a global development environment. In general, XP supports testing at three levels including: testing of cloud services, testing of mobile applications, and integration testing.

While developing ShaMoCloud, we wrote a native code of web services for the mobile side by calling HTTP libraries. We discovered towards the end of our implementation that there are promising new and open source solutions addressing creation of API clients such as Retrofit\(^\text{15}\) and Gson\(^\text{16}\). We recommend use of these tools to students and professionals developing applications.

\(^{15}\) http://square.github.io/retrofit/

\(^{16}\) https://code.google.com/p/google-gson/
References


C. J. Wang, X. L. Xu, D. Y. Shi and W. L. Lin, "An Efficient Cloud-Based Personal Health Records System Using Attribute-Based Encryption and Anonymous Multi-receiver Identity-


[59] S Toor, L Osmani, P Eerola, O Kraemer, T Lindén, S Tarkoma and J White, “A scalable infrastructure for CMS data analysis based on OpenStack Cloud and Gluster file system”,

93
Expedient Data Center Cost Etimator, Available Online: https://www.expedient.com/data-center-build-vs-buy-calculator/


Gareth Halfacree, Raspberry Pi review: http://www.bit-tech.net/hardware/pcs/2012/04/16/raspberry-pi-review/


Appendices

Appendix A: ShaMoCloud Web Interfaces

1. Screenshot for ShaMoCloud home page: user can select video from local PC and upload it to cloud account with Tags.

Figure 0-1 ShaMoCloud home page
2. Screenshot for ShaMoCloud login & registration page: user can log on from this page to access ShaMoCloud functionalities, and sign up to the cloud service.

![ShaMoCloud Login & registration page](image)

*Figure 0-2 ShaMoCloud Login & registration page*

3. Shared video page: user can see all the details of videos shared with friends. In addition, user can unshare videos.

![Shared video page](image)

*Figure 0-3 Shared video page*
4. Manage videos page: user can access uploaded video in order to delete it or share it with friends.

![Figure 0-4 Manage video page](image)

5. Manage friends page: user can see list of friends and add new friends from this page.

![Figure 0-5 Manage friend page](image)
6. New friend form page: user can send request invitation to friends, regardless of whether friend is registered.

![New friends form page](image)

*Figure 0-6 New friends form page*
Appendix B: ShaMoCloud Mobile Application

1. ShaMoCloud home screen: default screen that appears once user logs on to mobile app successfully. Contains the tabs Cloud Videos and Device Videos. Cloud Videos shows a list of videos stored in the cloud service, whereas Device Videos shows a list of videos stored in the device so that the user can select the desired video and upload it to the cloud.

![ShaMoCloud App home screen](image)

*Figure 0-7 ShaMoCloud App home screen*

2. Cloud Videos screen: user can see all videos stored in the cloud and manage them as shown.
3. Once user clicks a certain video, the application displays pop-up menu. After clicking a list item, the system runs the function.