Visualising Distributed File System Architecture using Mixed-Reality

MSc Research Project
Cloud Computing

Sarthak Sharma
x15047814

School of Computing
National College of Ireland

Supervisor: Mr. Victor Del Rosal

National College of Ireland
Project Submission Sheet – 2016/2017
School of Computing

Student Name:	Sarthak Sharma
Student ID:	x15047814
Programme:	Cloud Computing
Year:	2016
Module:	MSc Research Project
Lecturer:	Mr. Victor Del Rosal
Submission Due Date:	16/08/2017
Project Title:	Visualising Distributed File System Architecture using Mixed-Reality
Word Count:	7766

Signature:	
Date:	15th September 2017

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Visualising Distributed File System Architecture using Mixed-Reality

Sarthak Sharma
x15047814
MSc Research Project in Cloud Computing

15th September 2017

Abstract

Distributed File Systems (DFSs) have become a key component for cloud computing. However, a large-scale cloud computing cluster environment has to face many challenges due to file or data transfer and network overheads, which lead to the development of new distributed file systems, such as Hadoop Distributed File System (HDFS).

One of the key advantages of HDFS is the support for a fault-tolerant, scalable and reliable data storage. Due to this, HDFS is one of the most suitable DFSs to process MapReduce jobs. However, one of the major challenge with HDFS is that most of the techniques to analyse and identify the performance of HDFS are experimental, and hence, cannot be relied on.

Therefore, this research paper purposes a mixed-reality application, HoloHDFS, to understand and visualise the architecture of HDFS and its data processing with the help of mixed-reality and holograms. An application to analyse the resources of Hadoop cluster and provide a way to highlight the inefficient workload/data distribution in a distributed file system environment.

This paper describes the design and implementation of the application and discusses the encouraging results from the initial evaluation of the system, and further, discusses how the application can be extended and improvised in several ways.

1 Introduction

A Distributed File System (DFS) can be defined as a file system that gives the ability to multiple users to store and retrieve a file that is distributed across multiple nodes of a cluster with the help of a computer network to take advantage of data/file and storage resources sharing. With the evolution of cloud computing, the role of DFSs in a large-scale cloud computing environment has become even more vital and crucial. However, DFS comes with its own challenges in a cloud computing DFS environment involving scalable storage, data transfer bottlenecks and more. There has been a number of DFSs introduced to improve the performance and efficiency of DFSs in a large-scale cloud computing environment.

One such DFS is Hadoop Distributed File System (HDFS) that is one of the most important components of Apaches Hadoop framework. Apaches Hadoop framework allows the ability to perform large-scale distributed data processing. The HDFS is a fault-tolerant DFS that supports features, such as self-healing, reliable and scalable data storage on commodity hardware device. The HDFS is based on famous master-worker architecture where the Name Node acts as the master and Data Node acts as a worker. One of the major advantage of HDFS is perhaps its compatibility with MapReduce with the help of distributed storage and computing amongst nodes of large clusters. These nodes are capable of storing any format of data, such as text, binary or images, and can scale-up based on the number of requests while keeping the cost under control.

Undoubtedly, MapReduce has become one of the most critical technology in heterogeneous cluster and cloud environment. Proposed by Google in 2004, MapReduce is one of the most efficient and famous parallel computing framework for large-scale data processing. In general, MapReduce jobs include dividing of input files into a number of map tasks by the master node which then schedules and allocates the map and reduce tasks to the data nodes for parallel data processing.

However, every distributed system comes with new challenges and such is the case with HDFS. HDFS's purpose is to store huge files over a distributed environment and split the files into blocks which are bigger than the block size. In addition, multiple copies each of these blocks are stored on different data nodes to ensure data-locality. This technique is known as data replication which results into high availability of data.

Even though there has been a number of techniques available to analyse the performance of DFSs, especially HDFS, almost most of them are experiment-based which require to design the test and run them on DFSs to examine their performance. But the problem is, it is quite hard to design test for every possible real-life environment conditions, and hence, analysing the performance of DFSs is still a critical issue.

To resolve this challenge, this research paper proposes a mixed-reality based solution to visualise the DFS architecture using virtual objects, and thereby, providing a medium to identify the under-/over-utilised resources and interact with the resources using virtual touch.

1.1 Objectives and Research Question

With the evolution of wearable head mounted devices, such as Google Glass or Microsoft HoloLens, mixed-reality is becoming popular day-by-day. Mixed-reality allows us to merge the virtual objects with the real objects and enables us to interact with them as if we interact with the real objects.

Also, many researches have been performed to examine the impact of mixed-reality on the user experience as opposed to using 2D graphics on a 2D screen, and the results have shown how mixed-reality helps in providing an interactive and intuitive way of improving the user experience. Moreover, many current researchers are also focusing on using mixed-reality in education industry along with other industries, such as architecture visualisation.

This research paper will be focused around the following research question: "How does a mixed-reality application proves to be an effective and efficient medium to visualise the HDFS architecture and to analyse the performance impact of adding or removing resources from a Hadoop cluster?"

The major aim of this research will be on visualising HDFS environment using mixed-reality which can help in multiple scenarios and two of them are discussed below:

- For a user/developer, it will provide an effective medium to understand the HDFS architecture and the user will be able to visualise the concept of data-locality through live holograms representing the nodes in a cluster.

- For an administrator, it can provide the heat map to identify the under-/over-utilised resources/nodes. In addition, they should be able to analyse the impact of adding/removing resources to the Hadoop cluster.

The rest of the paper is divided into five more sections. The Section 2 starts with the discussion on importance of HDFS in cloud computing and the need or importance of visualising a DFS, such as HDFS, and then proceeds towards discussion the benefits of mixed-reality and mixed-reality head mounted devices. The Section 3 starts by providing the methodology to be used for this research along with supporting system architecture and other UML diagrams. Next, the paper discusses about the implementation of the purposed solution in section 4. Thereafter, provides detailed evaluation of the proposed solution in section 5, and thereby, concludes the research paper in section 6 along with future scope of work.

2 Related Work

2.1 Need for Visualising Distributed File System

2.1.1 Importance of Hadoop Distributed File System

An important component of the Hadoop framework is the fault-tolerant Hadoop Distributed File System (HDFS) which is based on Googles File System (GFS) (Ghemawat et al.; 2003) along with Java-based APIs. Hadoop framework provides parallel programming support distributed among different nodes of a distributed system with the help of MapReduce paradigm (Taylor; 2010).

Apache Hadoop is a widely used open-source implementation of MapReduce which was primarily developed by Yahoo. Hadoop is being used by Yahoo along with other wide variety of websites, such as Amazon and Last.fm, to process large-scale data on a daily-basis. Scientific data-intensive applications also take advantage from the Hadoop system in addition to data-intensive websites (Olston et al.; 2008).

Another computing area to take advantage of the Hadoop framework is the High Performance Computing (HPC) which is famous among various data analysis areas, such as bioinformatics (Taylor; 2010). HPC requires distributed computing, for which parallelisation is an important paradigm which is implemented through various programming APIs, such as Message Passing Interface (MPI) and the most popular one Hadoops MapReduce (Taylor; 2010).

Dean and Ghemawat (2004) from Google Inc. introduced and implemented highly scalable MapReduce in a large clustered environment at the beginning of 21st century. A proper implementation of MapReduce can process and compute many terabytes of data on a cluster with thousands of nodes (Dean and Ghemawat; 2004).

The above discussion proves that MapReduce is being used in DFSs from more than a decade now Dean and Ghemawat (2004), and simultaneously, has been implemented in many DFS frameworks, such as Apache's Hadoop framework which uses HDFS (Taylor; 2010). Though, Hadoop framework provides Job and Task trackers (Taylor; 2010), they aren't sufficient enough to visualise the processing of MapReduce within Hadoop framework and doesn't provide enough tools to analyse the performance or resource utilisation.

2.1.2 Current Approaches Available for Performance Analysis and Their Limitations

Wu et al. (2014) have explained how Distributed File Systems (DFSs) plays an important and crucial role in a large-scale cloud computing environment. Scalable storage, data transfer bottlenecks and performance unpredictability are few of the challenges faced by applications on these large-scale cloud computing DFSs. Many new DFSs, such as HDFS, GFS and MooseFS have been developed to avoid the aforementioned challenges.

Wu et al. (2014) have studied how the design-time performance can be used by system architects to measure and estimate the resource utilisation, throughput and timing behaviour of a DFS in advance to the deployment of the actual system. Another experimental technique to analyse performance of HDFS discovered by Plantenga et al. (2012) which involves analysing the information from Hadoop logs, machine/system metrics and job state after its completion.

Wu et al. (2014) mentioned some of the factors how design-time performance is of such importance. These are, 1) testing the performance of the system can be expensive as compared to analysing the performance of the system, 2) testing the system with all various kinds of configurations can be complex and infeasible and 3) analysing the system in advance to configuring and deploying the system can reduce the chances of redesign, reconfiguration and redeployment. On the other hand, Plantenga et al. (2012) extended their performance analysing method using the traditional Hadoop Job-Tracker analysis with the help of including additional data for analysis and using Web browser based visualisation tools.

Wu et al. (2014) proved that analysing and testing the running and deployed system can be expensive and time consuming. Thus, it can be said that there is a need for further research to develop tools that provide analysis of a running and deployment DFS. Moreover, eventhough Plantenga et al. (2012) provided interactive and visual tools, these tools only provide the analysis of the jobs that are processed and completed.

The number of proposed approaches to quantitatively and qualitatively measure the performance of a DFS and its nodes or components that analyse a system prior to its deployment and without the need of benchmark tools are limited (Wu et al.; 2014). A common approach to evaluate the performance of DFSs is to run a series of experiments on running DFSs; this implies that they are majorly focused on analysing the experiment results or drawing comparison between existing DFSs.

This subsection can be concluded by saying the basic approaches discussed in this section and many more based on experimenting by running the DFS are not reliable in every scenario and can take huge amount of time depending on the scale of DFS. Therefore, this research paper will focus on providing an interactive mixed-reality visualisation of HDFS to analyse and monitor the performance of the system by the means of a virtual heat map of the system. The following subsection discusses about mixed-reality and the challenges associated with mixed-reality to provide a realistic environment for visualising an interactive presentation of HDFS.

2.2 Using Mixed-Reality for Visualisation

2.2.1 Introducing Mixed-Reality and Mirror World

With the evolution of wearable devices supporting mixed-reality, such as Google Glass and Microsoft HoloLens, augmenting the physical world is now possible by creating a digital layer (a mirror world) (Ricci et al.; 2015). Such devices have seamlessly extended how people perceive the information provided by the digital layer and how they influence people's actions in the real as well as the virtual/augmented world.

Ricci et al. (2015) have discussed about a mirror world vision in which digital cities are modelled and coupled with the physical world and software services/agents together act as the inhabitants/societies of these digital cities. Mirroring is a crucial aspect for designing mixed-reality and extending the human computer interaction. Ricci et al. (2015) explains mirroring implementation as when there exists a digital entity for a physical object (an object that can be perceived, sensed and acted upon by humans in the real world) which can be perceived and acted upon by the software agents. Conversely, a digital entity that exists in a mirror world which can be perceived and acted upon by the software agents can have a physical appearance in the real world (which can be done using augmented/virtual reality) where the humans can perceive and act upon these entities.

2.2.2 Rendering Virtual Objects in Mixed-Reality Scenes

Now-a-days, mixed-reality is becoming one of the new, interactive and intuitive way to visualise the virtual objects in a real-world environment in multiple application domains, such as educational and entertainment systems, prototyping and architectural visualisations (Knecht et al.; 2012) (Kronander et al.; 2015). One of the long-lasting and important aspects of computer graphics is to generate and merge realistic images and the virtual objects into a mixed-reality scene to give an illusion of a real-world (Kronander et al.; 2015). As a result, there has been an increase in the development of several methods to capture the lighting conditions in real-world environment as well as to merge the virtual objects into mixed-reality scenes to make them look more realistic.

Knecht et al. (2012) have highlighted one of the major challenges that is commonly ignored in the mixed-reality systems. The mixed-reality systems used for visualisation of various application domains requires a realistic presentation of virtual object, however, this is commonly ignored due to the ignorance of real illumination conditions and the mutual shading effects among the real and virtual entities (Knecht et al.; 2012). On the other hand, Kronander et al. (2015) have highlighted some key factors to evaluate the method for capturing and rendering virtual objects in a mixed-reality scene. Some of these factors involve, illumination assumption of the captured scene, time and effort required to capture and process the scene, illumination and accuracy of the final reconstructed and rendered scene and the perceptual quality of rendered scene. In addition, Knecht et al. (2013) focused on supporting reflective and refractive objects in a mixed-reality scenario which involves the reflection or refraction of real objects in or through virtual objects and vice-versa.

Knecht et al. (2012) have introduced a rendering method by combining an Instant Radiosity algorithm along with Differential Rendering that presents a mixed-reality scenario at real-time frame rates with illuminating the global scenario. In other words, Knecht et al. (2012) provided a method to render a mixed-reality scenario where the real scene can be relighted and the virtual objects can be illuminated by a real or virtual source of light. Continuing with the Knecht et al. (2012) research, Knecht et al. (2013) extended the Differential Instant Radiosity (DIR) method presented in Knecht et al. (2012) which was unable to implement differential shading and illumination effects to the reflected or refracted objects in a mixed-reality scene. Knecht et al. (2013) extended the DIR method to add the differential effects to the objects in addition to global illumination and then processes the new colour information to the user end to simulate the reflective and refractive objects in a mixed-reality scene. Using the Microsoft Kinect sensor, Knecht et al. (2012) reconstructed a real scene and stimulated mutual shading effects among the real and virtual entities with promising results.

2.2.3 Gadgets to Interact with Mixed-Reality

With the advancement in computer graphics, the entertainment industry has become deeply dependent on the Computer-Generated Imagery (CGI) graphics for the products, such as video games and movies (Kade et al.; 2015). In addition, with the use of CGI graphics, the developers need to ensure the movement of CGI characters match the real-world physics. Kade et al. (2015) have experimented using dedicated vision-based motion capture technique as one of the most commonly used techniques to capture real-life actors, e.g. humans, motion for daily-life activities and for professional purposes. Similarly, Mateu et al. (2014) have also developed a tool for mixed-reality paradigm that can be used to create virtual and interactive content for the educational application.

Kade et al. (2015) pointed out the limitation of dedicated vision-based motion capture which can only be done in a studio as the actors involved require an acting environment which is generally prepared using wooden or metal props. Thus, to address this issue and eliminate the need of creating an external acting environment, Kade et al. (2015) developed a new interface that can be used to project and acting environment to give the same experience to the actors as in a motion capture studio, even outside the studio.

As per Kade et al. (2015) results, they have proved that a head-worn device can be used to explore a virtual environment that provides an immersive and interactive experience for the user. One of the major advantage of their prototype is that the device is itself mobile without any requirement for tethering wires or external projecting, tracking or computing systems. On the other hand, to interact with the virtual objects, (Mateu et al.; 2014) used a virtual world within a mixed-reality scene for simulating the virtual objects in a three-dimensional space. This is done by augmenting tangible and gestural virtual objects upon which gestures can be tracked or recognised, and hence, the interaction with the virtual world is established (Mateu et al.; 2014).

However, there are certain limitations in the head-worn device developed by Kade et al. (2015). Firstly, it still requires a screen to which it can project the virtual environment. Secondly, their prototype cant be used when it comes to interacting with the virtual environment. However, Kade et al. (2015) have proven that using a head-worn device to project the virtual environment provides a better and interactive way of visualising a virtual environment and it is cost effective and mobile solution to a motion capture studio.

Another important branch of mixed-reality that deals with incorporating the ability for the user to interact and touch virtual objects in a real scene is Visuo-Haptic Mixed Reality (VHMR) (Barbieri et al.; 2014). VHMR requires a see-through display that will project and mix the virtual objects with the real objects, and haptic devices that are required for providing the haptic stimuli when the users interact with the virtual objects. VHMR has been widely-used in many industrial areas, especially for virtual product prototyping which helps in reducing the product development cycle significantly (Ortega and Coquillart; 2005) and providing a collaborative experience in digital environment field (Knoerlein et al.; 2007).

Barbieri et al. (2014) focused on investigating the two critical problems that arise due to haptic interaction devices adopted in VHMR environments. These are, obtrusive problem due to haptic interaction devices and virtual tool misalignment problem. These problems hamper the user experience and impacts the illusion of the virtual world (Barbieri et al.; 2014). The obtrusion problem is a result of haptic devices physical presence in the mixed-reality environment (Barbieri et al.; 2014). These haptic devices occupy a large space and are present in the area where the interaction with the virtual objects is supposed to occur and hence, become obtrusive visual elements. As per the results collected and calculated by Barbieri et al. (2014), these obtrusive elements impact the user perception of the virtual objects negatively and reduces the user experience of the tasks involved.

This section provided an overview of the important factors that need to be considered while working with virtual objects in a mixed-reality scene. In addition, it proves how mixed-reality can improve the user experience using wireless, tethering free devices, such as a heard-worn mixed-reality device (Microsoft HoloLens) (Kade et al.; 2015), providing a way to interact with the virtual and real objects with virtual touch (Mateu et al.; 2014). However, issues like misalignment and obtrusive haptic device can also ruin the user experience (Barbieri et al.; 2014), and hence, these factors should be considered while choosing a platform/device to develop and project mixed-reality scenes.

2.2.4 Using Mixed-Reality in Education and Training Areas

As stated by Wegman (2000), "The cave automatic virtual environment is an immersive stereoscopic projection based virtual reality environment that has become, in many ways, the VR environment of choice." It provides a prominent level of presence of the virtual environment, however, the environment itself doesn't react or adapt as per the users actions (Nakevska et al.; 2017).

Nakevska et al. (2017) conducted an experimental study on interactive storytelling in a mixed-reality environment, which combines the virtual and real objects, information and their features, to provide an immersive experience to a dramatic storyline of Alices Adventures in Wonderland. Nakevska et al. (2017) call it as ALICE installation in which they investigated a study of immersiveness through interactive storytelling which was carried out with 41 participants, and the proves that immersiveness is not dependent on quality of stimuli, but in fact, it dependent on their time-density. On the other hand, Mateu et al. (2014) displayed with their results from their practice of 570 use cases using their tool how mixed-reality can be used in the education industry by simulating virtual worlds with real contexts.

Ke et al. (2016) discusses about the evolving Mixed-reality Integrated Learning Environment (MILE) and the need for more experimental research on MILE to improves its design, implementation and educational effectiveness. Ke et al. (2016) examined a MILE platform integrated with a Kinect sensor in a training environment for university teaching assistants. The results suggested how a MILE platform can enhance the immersiveness of the teaching experience and increases the user interactions.

Bertram et al. (2015) conducted a similar study using virtual training session, but their participants involved members of police force. Continuous training is an important part for the success of emergency service personnel, such as fire fighters and police. The solution suggested by Bertram et al. (2015) involves Virtual Training Environments (VTEs) which can make it easier to simulate these operations for training purposes without much complexity and in a cost-effective manner (Bertram et al.; 2015).

3 Methodology and Design Specification

Concluding from the discussion in the previous section, it is now clear that the mixed-reality technology can be used to represent the components of HDFS as holograms in a virtual world and to visualise the resource utilisation with the help of heat map. The main aim of this research paper is to develop a mixed-reality application for a mixed-reality projecting device, such as Microsoft HoloLens, and make it support virtual touch through which the users will be able to visualise the HDFS architecture as well as interact with its components and change its configuration.

3.1 Methodology to Visualise Resource Utilisation

In this research paper, to analyse the resource utilisation, CPU usage and the HDFS under-/over-replicated block report have been used. Though, the individual data node block replication report can be generated, but that turned out to be an additional overhead as this report will be generated each time the application makes a request to the master node. Therefore, to avoid the overhead, only the block replication report of the root directory of HDFS is calculated and sent to the mixed-reality application.

As per the HDFS architecture illustrated by Sheu et al. (2014), the proposed methodology to analyse and visualise the resource utilisation for HDFS is given below:

1. As soon as the Hadoop cluster starts, a log file is maintained which helps in keeping track of all the data nodes/workers in the cluster. Whenever a node is added or removed from the cluster, this log file is updated in the system.

2. Whenever a request is received from the application, the master node calculates the CPU usage of individual node in the cluster using the above mentioned log file.

3. Further, the master node generates the block replication report using the HDFS framework API and returns the under-/over-replication block percentage.

4. Once the resource utilisation report for individual node including the block replication for root HDFS directory is generated, a heat map can be generated using this metadata. Now, with the help of mixed-reality virtual object or holograms, the augmented data nodes can be highlighted based on their resource utilisation.

Apart from visualising the resource utilisation, this research paper also aims to provide a medium for the user to interact with the augmented HDFS architecture by the means of virtual touch using mixed-reality scenes. After generating the heat map, the user should be able to interact with the virtual objects, such as the cluster node, and view the individual node's description or manage the cluster nodes.

The first phase of this research focuses on visualising the architecture of the system and generate heat map for over utilised nodes. The second phase focuses on extending the features of the application and providing options to change the Hadoop cluster configuration, such as adding/removing nodes to reduce over-/under-replicated blocks count and starting/stopping a MapReduce job.

3.2 System Design and Architecture

While deciding the system design and architecture for the mixed-reality application, the aim was to design a product a product that is user-friendly and easy-to-use which doesn't require a user to have knowledge of Hadoop framework, mixed-reality technology or mixed-reality devices as prerequisites. In addition, the product should also be flexible enough to support advanced features, such as managing the Hadoop cluster configuration, for the expert users. Therefore, considering the requirements of this product, the two types of users have been distinguished as give below:

- Basic user - They should be able to visualise the virtual HDFS architecture without any prior knowledge and should be able to analyse the resource utilisation. From a developer's perspective, the basic features should be enough to understand the HDFS architecture.

- Advance user - In addition to the basic user feature, the advance users should be able to interact with the virtual objects and manipulate the Hadoop configuration. From an administrator's perspective, the advance features should allow them to analyse the under-/over-utilised resources and control the resource utilisation by managing the Hadoop cluster environment.

To design such a product with the aforementioned requirements, the system architecture of the application contains the following components:

- A mixed-reality projecting device (MR device) with integrated or external set of required tacking sensors.

- In case, the MR device doesn't include a computing unit, an optional external computing system will be required which connects with the MR device.

- A running Apache Hadoop cluster on a public cloud.

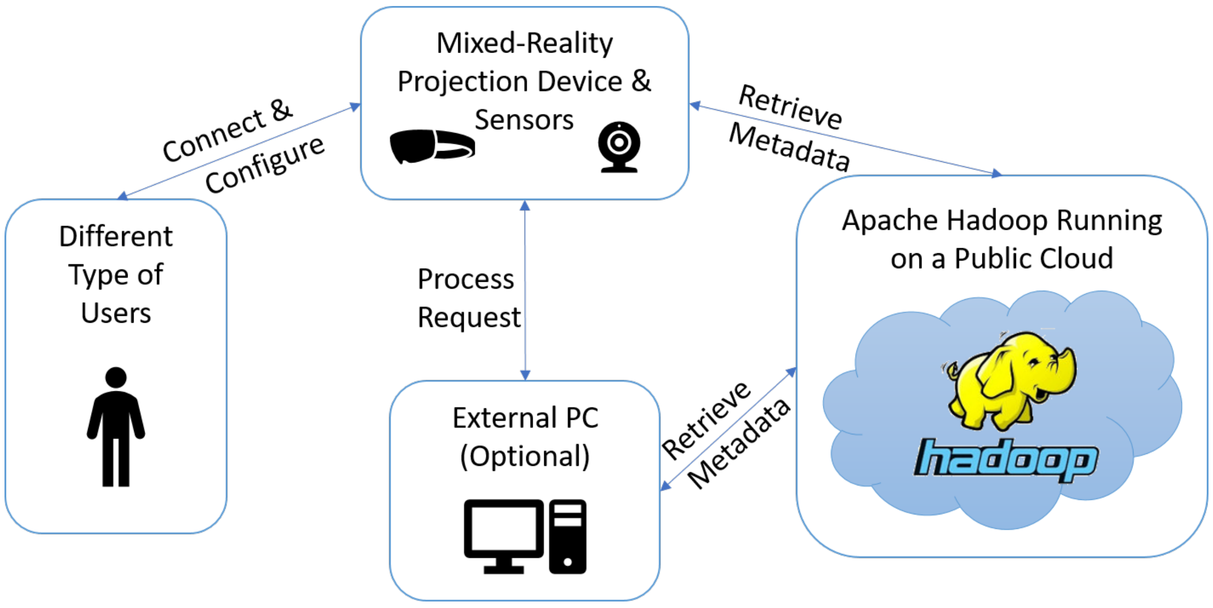


Figure 1: System Architecture Overview

The Figure 1 provides an overview of the system architecture. In addition, the Figure 2 provides an overview of the software architecture for this product.

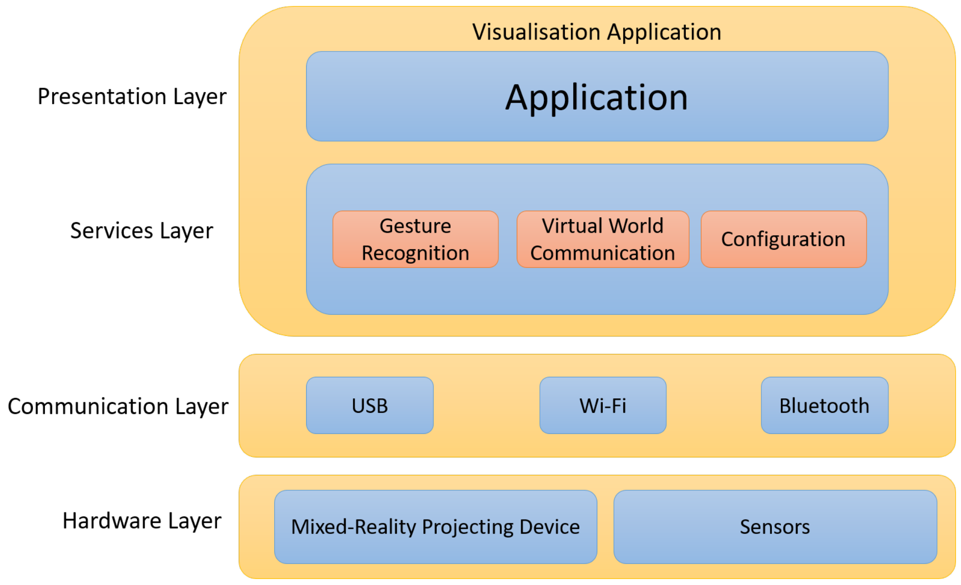


Figure 2: Software Architecture Overview

The different software architecture layers shown in the Figure 2 are defined in the following list:

- Hardware layer - consists of the required hardware required to run the application. This will be a mixed-reality device, such as Microsoft HoloLens.
- Communication layer - consists of communication devices that connects the MR device with the Hadoop cluster. For instance, USB port, Wi-Fi or Bluetooth. Often, these devices are integrated within the hardware device.
- Services layer - consists of various services including Gesture Recognition (responsible for interacting with virtual object), Virtual World Communication (responsible for sending and receiving data from Hadoop cluster to the product itself) and Configuration (responsible for configuring the Hadoop environment settings).
- Presentation layer - is the closest layer to the user. This layer is the actual mixed-reality application rendering and merging the virtual world with the real-world. This layer will be responsible for the user interaction with the virtual and real objects.

The use case diagram represented in Figure 3 depicts the different type of actions that can be performed by different types of users.

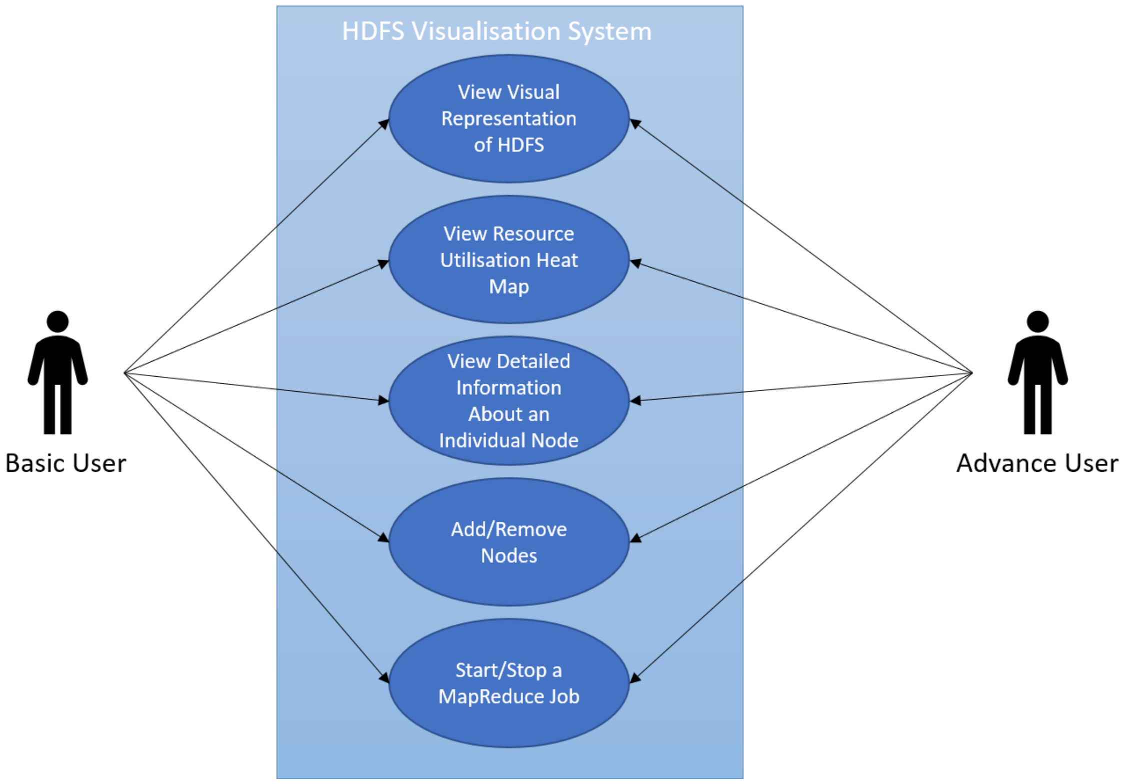

HDFS Visualisation System
View Visual Representation of HDFS
View Resource Utilisation Heat Map
View Detailed Information About an Individual Node
Add/Remove Nodes
Start/Stop a MapReduce Job
Basic User
Advance User

Figure 3: Use Case Diagram of the System

Based on the actions that can be performed by different types of users, the system can be formalised into the following two sequence diagrams.

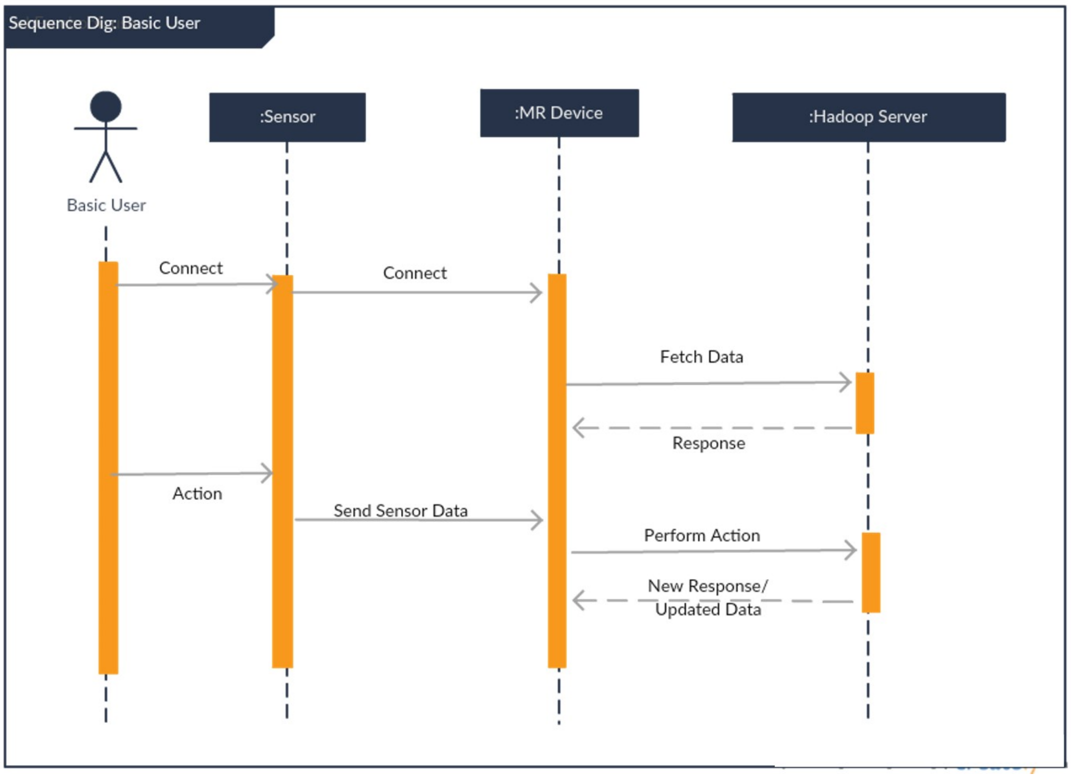


Figure 4: Sequence Diagram for Basic Users

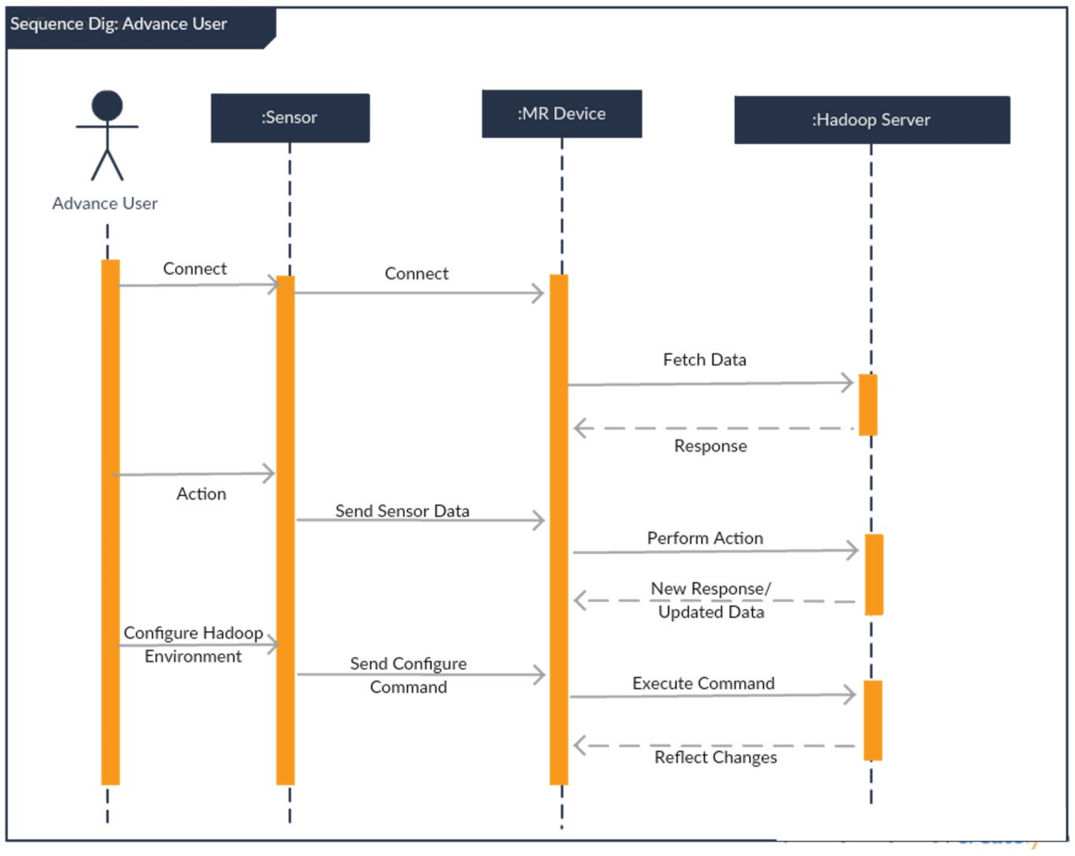


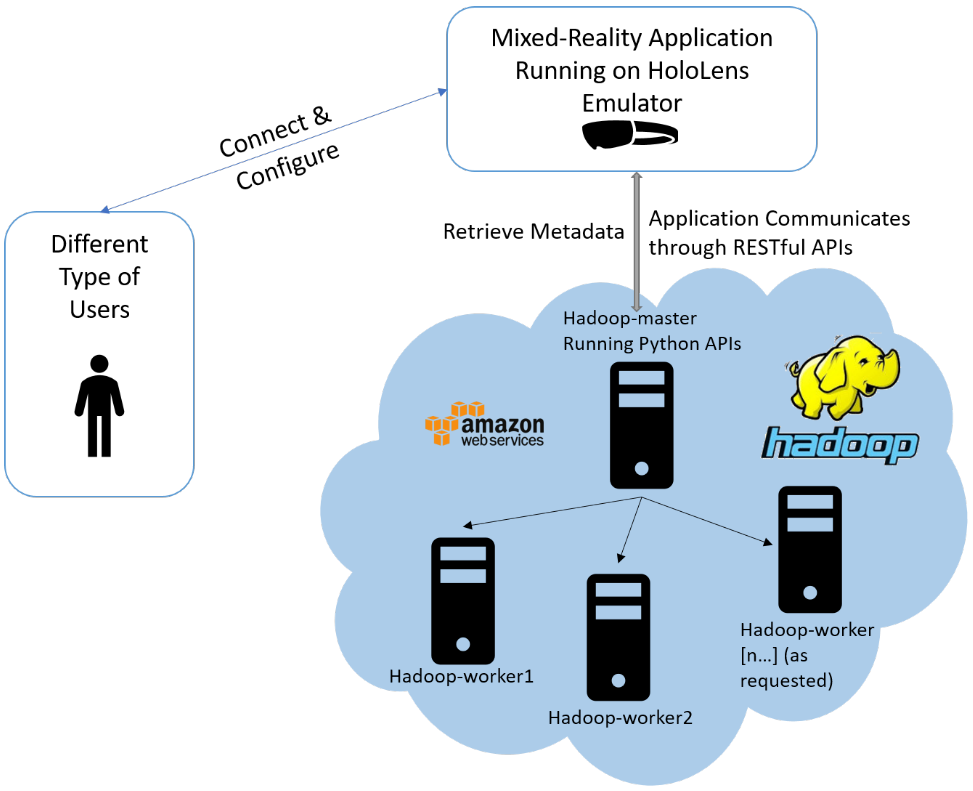
Figure 5: Sequence Diagram for Advance Users

4 Implementation

The mixed-reality application developed in this research paper is named HoloHDFS for easier reference. The HoloHDFS application architecture consists of the following layers:

- A Hadoop cluster running on Amazon Web Services (AWS) public cloud.
- A Representational State Transfer (RESTful) API running on the Master node/NameNode exposing the various functionalities, such as add/remove an instance to/from the Hadoop cluster, to the mixed-reality application.
- A mixed-reality application for Microsoft HoloLens running on HoloLens emulator for Microsoft HoloLens OS.

The Figure 6 illustrates the overall architecture of this mixed-reality application.



Mixed-Reality Application Running on HoloLens Emulator
Connect & Configure
Different Type of Users
Retrieve Metadata
Application Communicates through RESTful APIs
Hadoop-master Running Python APIs
amazon webservices
hadoop
Hadoop-worker1
Hadoop-worker2
Hadoop-worker [n...] (as requested)

Figure 6: The HoloHDFS Overall Architecture

The HoloHDFS application is developed, tested and running on a Microsoft HoloLens emulator and is fully compatible with an actual HoloLens device. As soon as the application is started, it connects with the RESTful API running on master node which returns the Hadoop cluster configuration details to the application in JavaScript Object Notation (JSON) format. The application then reads the data and creates Holograms as per the number of workers/data nodes. The application keeps the connection open with the API so that the application gives a live visualisation of the cluster. This way, in case, a node is added/removed to the cluster externally or by another user, the application will be able to reflect the changes instantly.

In addition, the API also helps in executing the commands for adding/removing a node or start/stop a sample WordCount MapReduce job. Users are able to interact with the holograms with the help of virtual touch. There's a side menu to add an instance and to start/stop the sample MapReduce job. In addition, a pop-up menu opens up when a user performs an air taps on the hologram which provides them details about the node and an option to remove the instance. However, for security purposes, the user cannot delete the master node and the two default worker nodes.

Moreover, the application gets live performance data of the cluster nodes through the APIs to highlight under-/over-utilised resources. At present, only the CPU usage and block replica details are being used to identify whether a node is being under-/over-utilised or if a block has been the under-/over-replicated. If the CPU utilisation for a node is over 20%, the application will highlight that particular node. The block replication is displayed in a panel all the time right next to the master node. The user can then tap on the respective node and view the health status. The CPU usage will be displayed for every node, however, instead of displaying the individual block replication report, the block replication data for the master directory is displayed only.

4.1 Deploying a Hadoop Cluster

As this mixed-reality application helps in visualising and analysing the Hadoop cluster, a deployed and running cluster was required. This was achieved by setting up Hadoop cluster using AWS public cloud. The configuration of the Hadoop Cluster is given in the Table 1:

Hadoop Cluster Configuration				
Instance Name	AWS EC2 Type	RAM (in GB)	EBS Storage Capacity (in GB)	Operating System
hadoop-master	t2.micro (1 vCPU)	1	1	Ubuntu 16.04 LTS
hadoop-worker1	t2.micro (1 vCPU)	1	1	Ubuntu 16.04 LTS
hadoop-worker2	t2.micro (1 vCPU)	1	1	Ubuntu 16.04 LTS

Table 1: Hadoop Cluster Configuration

In addition to this, an image of the "haddoop-worker1" instance has been created and stored in AWS Elastic Block Store (EBS) storage which will be used to launch a new instance. Thereafter, the newly launched instance will be added to the Hadoop cluster whenever the "Add Instance" request is received from the application. This makes the process of launching and adding an instance easier and quicker as only the image id will be required to launch a new instance.

In addition, these nodes are communicating with each other through each other's private IP for security purposes. However, since the RESTFul API will be hosted on the master node, the public IP has been enabled on the master node along with access to the public traffic.

The following two subsections discusses about how an instance is added/removed and how the cluster's configuration/metadata is stored and managed in log files using various shell scripts

4.1.1 Adding/Removing Instances

To add/remove an AWS instance to/from the running Hadoop cluster, AWS Command Line Interface (CLI) has been used. AWS CLI provides the ability to manage and work with AWS services from the Bash shell of Linux operating system. To use AWS CLI, Python version 2.6.5 or higher is required. AWS CLI can be installed using the“pip install awscli” command from the Linux terminal.

Next, a shell script with the name “launchAWS.sh” has been stored in the home directory of the default user to launch a new AWS EC2 instance. A new instance is launched using the image id of the stored image in AWS EBS storage and then it’s hostname is added to the required files among the cluster to make it an active worker/data node for the Hadoop cluster.

Another shell script with the name “terminateAWS.sh” has been stored in the home directory to destroy the AWS EC2 instance and add it’s hostname name in the dfs.exclude file on the master node which tells the Hadoop cluster to decommission the mentioned instance. Now, whenever a request is received from the application, the respective shell script is run and the result is sent back to the application.

4.1.2 Creating and Managing Logs

A log file is created and stored in a safe directory which is exposed to the application through the RESTful API. This log file is created for the first time when the cluster is started and it is maintained throughout the life of the Hadoop Cluster. Whenever a new instance is added to the cluster or an instance is removed from the cluster, its entry is being stored in the log file along with its CPU utilisation which is updated every second to give live status of the nodes in the cluster. These log files are stored in a CSV format for easy reading and manipulation of data.

4.2 Creating RESTful API

Since the HoloHDFS application needs to communicate with the Hadoop cluster, RESTful APIs were developed using Python 2.7 framework along with a Python microframework named Flask. Flask is a micro Web framework which is written in Python and one of its features among others is to help create RESTful APIs. The reasons for choosing RESTful APIs using Python and Flask for the communication between the master node and the application is given below:

- RESTful Web services help in providing interoperability between different types of systems over the Internet. Since, the HoloHDFS application is running on a completely different environment, an interoperable environment was required to expose the data from Linux shell to the application. This was easily achieved by using Python along with Flask.

- Python is a lightweight development framework that allows to read and write CSV files. In addition, it also allows access to the Bash shell from within the Python script.

- Python also allows you to convert the data into JSON format which is one of the lightweight and platform independent medium to transfer the data. Further, with the help of Flask, various Python functions were converted to HTTP GET requests which, in turn, call the required shell scripts or perform required actions.

The RESTful API developed in Python exposes the following resources:

- http://hadoop-masterIP:5000/workers - This API exposes the list of master and its workers along with other necessary data. When executed, it gets the data about the workers from the log file, fetches the CPU usage and block replication report, converts it into JSON and then transfers the data as response.
- http://hadoop-masterIP:5000/launch - This API launches an AWS EC2 instance, adds it to the cluster, starts the required services on the newly added node and returns the updated metadata to the application.
- http://hadoop-masterIP:5000/destroy?ip=0.0.0.0 - This API accepts the private IP address of the node to be destroyed as an input in the form of query string. When executed, the API adds the information about the node to be destroyed in the dfs.exclude file which gives the command to the cluster to decommission the node, and thereafter, deletes the AWS EC2 instance and refreshes the Hadoop cluster nodes.
- http://hadoop-masterIP:5000/startjob - This API starts a sample Word Count MapReduce job on a file stored in the /user/hololens/input directory under the HDFS root directory.
- http://hadoop-masterIP:5000/stopjob - This API stops all the MapReduce jobs running on the master node.

4.3 Developing the Mixed-Reality Application

To develop the mixed-reality application, Microsoft HoloLens platform was selected. To develop an application for the HoloLens, the following frameworks/tools are required:

- Visual Studio 2017 along with Windows 10 SDK - This IDE allows a developer to develop Universal Windows Platform applications and to run, deploy or debug an application on various environments including Microsoft HoloLens emulator.
- Microsoft HoloLens Emulator - This allows to run a HoloLens mixed-reality application in an emulated environment using the Hyper-V virtualisation technology. The emulator runs the Windows Holographic OS which is the same OS as on Microsoft HoloLens.
- Unity version 5.5 or higher - Unity is a game development engine that allows to develop application for various platforms along with Universal Windows Platform. In addition, it also allows to develop application for virtual- and mixed-reality devices. It allows to write scripts using C# that makes it easier to invoke RESTful APIs which is one of the requirements for this project.

The Figure 7 and Figure 8 illustrates the user interface of the HoloHDFS application running on a HoloLens emulator. The Figure 7 depicts the default UI that a user will see when they start the application. It shows all the running nodes of the Hadoop cluster as holograms. Whereas, Figure 8 depicts the UI a user will see after performing a virtual touch on a node, and from there, they can either remove the instance or go back to main screen.

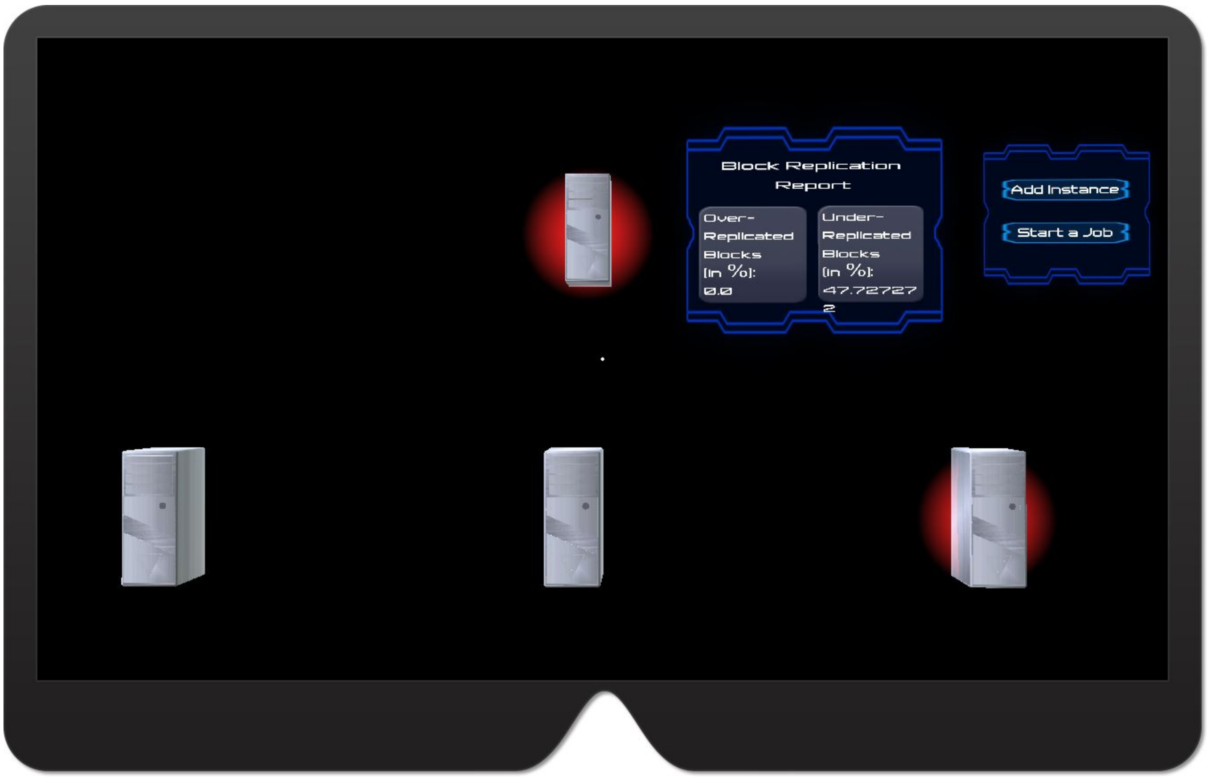
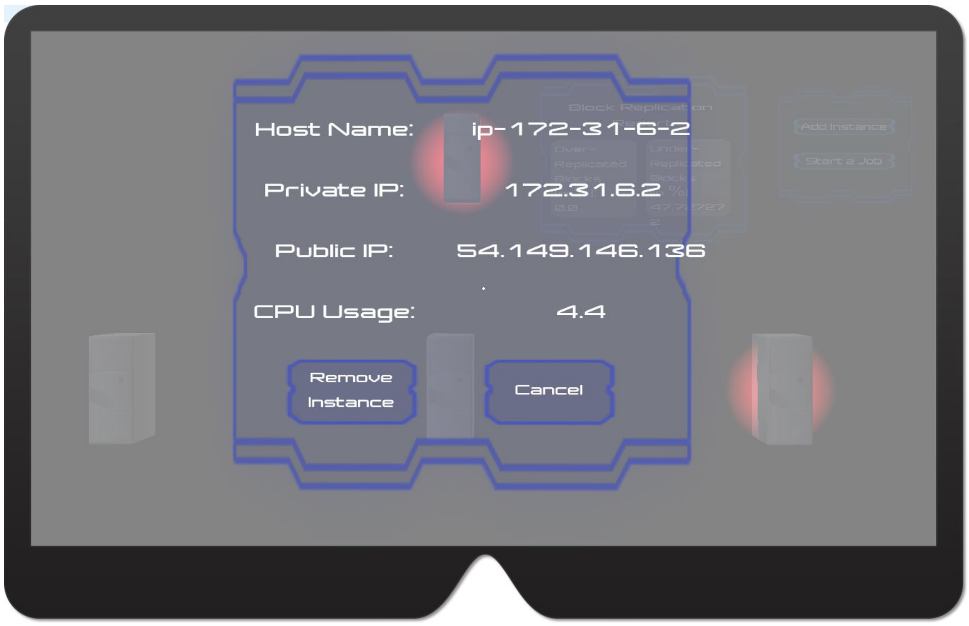

Block Replication Report
Over-Replicated Blocks [in %]: 0.0
Under-Replicated Blocks [in %]: 47.727272
Add Instance
Start a Job

Figure 7: The HoloHDFS Main Screen



Host Name: ip-172-31-6-2

Private IP: 172.31.6.2

Public IP: 54.149.146.136

CPU Usage: 4.4

Figure 8: The HoloHDFS Menu Screen

5 Evaluation

To evaluate the implementation of the HoloHDFS application developed as part of this paper, the application was deployed on the HoloLens emulator. Afterwards, the application's functionalities and the performance were tested using two different scenarios. The Table 2 describes the configuration for both the evaluations.

Component	Machine Name
Hadoop Cluster	AWS EC2 Instances (hadoop-master, hadoop-worker1, n...)
RESTful API	hadoop-master
HoloHDFS App	HoloLens Emulator on a Supported Machine

Table 2: Testing Configuration

5.1 Functional Evaluation

In the first phase of evaluation, the following functionalities of the application were tested:

- Testing whether the application can request to add an instance to the cluster and whether the newly added instance appears on the screen dynamically.
- Testing whether a user can perform a virtual touch on any of the instances represented as holograms and view the details about that individual node.
- Testing whether a user can remove an individual instance using virtual touch.
- Testing whether a sample MapReduce job can be started and stopped from the application.
- Testing whether a node is highlighted with red color if the CPU utilisation goes above 20%.

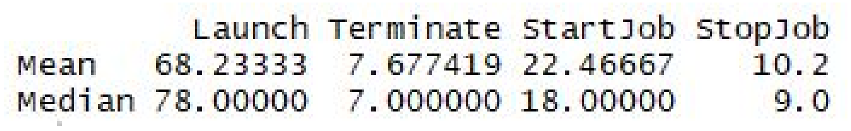
For the above tests, the required actions were performed from the running application and the results were logged using the shell respective script into CSV files. The above mentioned actions were performed for 30 times to make sure the application is performing as expected.

As per the expectations, the functionalities performed well, even when multiple commands were sent at the same time. However, a variation can be seen in the response time when multiple requests are made simultaneously. These variations and the results are discussed in the next evaluation section.

5.2 Performance Evaluation

In the second phase of evaluation, the time required to add and remove an instance was logged and analysed. While logging the time required to add/remove an instance, the start time is considered as the time when the request was received by the API and the end time is considered as the time when the request is finished processing. This task was performed for 30 times so that there's enough data to calculate the average response time. Similarly, the response time for the starting or stopping the MapReduce job was recorded, however, this task was performed for 15 times only.

The Figure 9 provides the mean and median response time in seconds for various tasks calculated using the data collected in the second phase of evaluation.



```
         Launch Terminate StartJob StopJob
Mean   68.23333  7.677419 22.46667    10.2
Median 78.00000  7.000000 18.00000     9.0
```

Figure 9: The Mean and Median Response Time (in Seconds) for Various Tasks

The Figure 10 provides an illustration of response time for various tasks with the help of histograms. In addition, the figure also includes a curve within each histogram which depicts the median response time for each task.

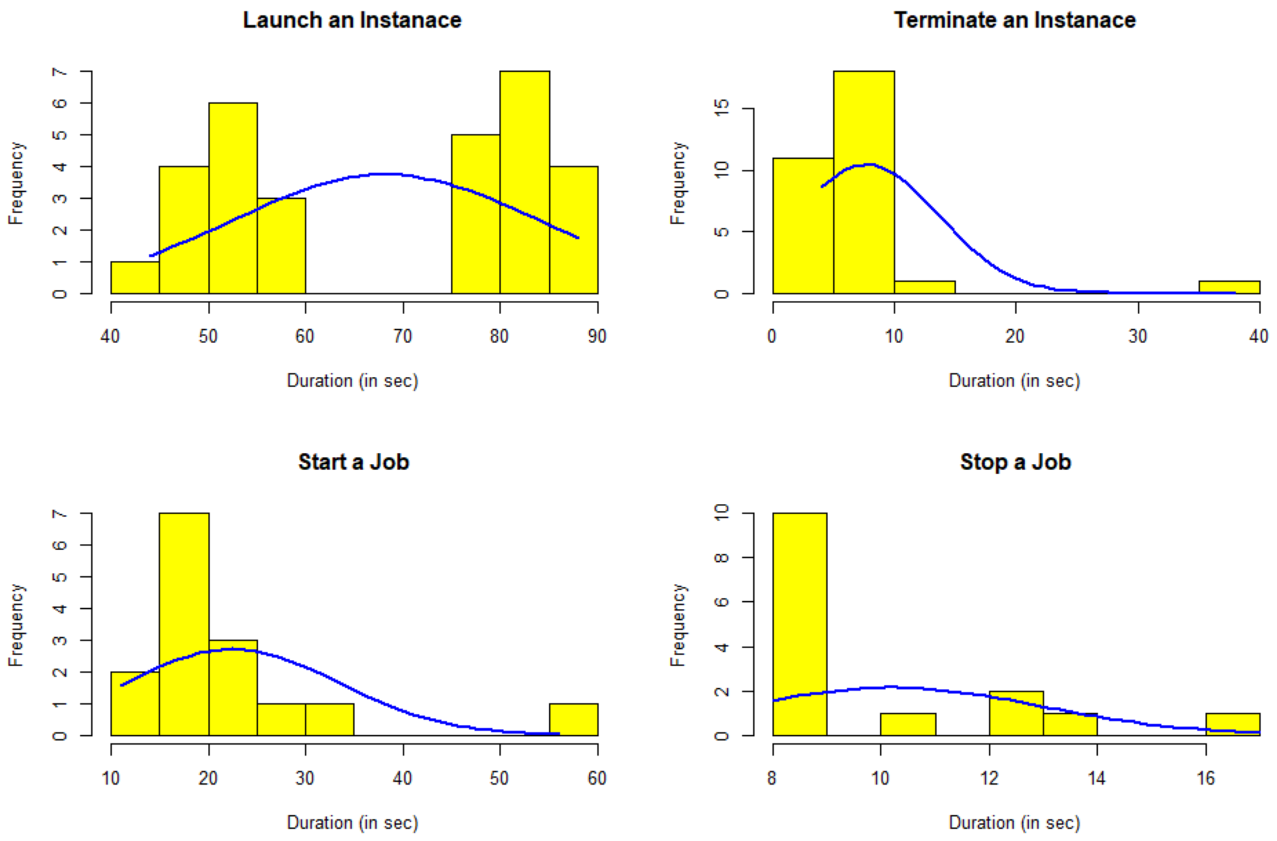


Figure 10: Response Time (in Seconds) for Various Tasks

5.3 Discussion

During the first phase of evaluation, it was noted that while adding an instance, the application displays the instance as a hologram within the application as soon as the instance is launched on AWS cloud. However, adding the newly added instance to the cluster takes little longer. Moreover, a user doesn't have to wait for the previous task to complete to perform same or another task, but the response time may vary. For example, after sending a request to add an instance, the user can make another request to delete another instance, start a MapReduce job or add another instance.

Though, the RESTful APIs can accept multiple requests from single or multiple users at the same moment of time, in the current version of the APIs can only process one request at a time. This is because the APIs aren't asynchronous in nature, that is, the APIs can't perform parallel tasks, but this doesn't result in lost/failed requests. To resolve this issue, the RESTful APIs can be modified to accept and process requests from single or multiple user(s) simultaneously. This will also improve the response time for processing a request and reduce the waiting time for the users.

One of the main aim to perform these evaluations was to confirm that there are no failed requests and the HoloHDFS application and the APIs running on master node keep communicating with each other, even during simultaneous requests from the application. This aim was confirmed from the results which showed all the functionalities work fine as there was no empty/nil value in the log files. All the entries in the log files had a start time and end time for each request. The APIs log the time they start processing a request into separate log files for each type of requests and save it as the start time. Whereas, the end time in the log files is the time just before the API returns the response to the calling application. Having no empty or nil values in the log files prove that each request received by the APIs successfully finished its processing and returned the response to each request.

During the second phase of evaluation, it was found that response time for adding an instance reduces if the requests are made immediately after the previous request has finished processing. That is, if a user makes simultaneous requests to add an instance, the response time for these requests seems to be improved than individual requests. One of the reasons behind this sudden drop in response time is CPU usage. If the CPU utilisation is high, the response time reduces, and if the CPU is idle, the response time increases. This is due to using a small tier AWS EC2 instances. Using high tier EC2 instances or instances with better computing capabilities should avoid this kind behaviour and should give a normalised distribution response time curve for adding an instance.

However, the same is not true for the other tasks, terminating an instance or starting/stopping MapReduce jobs. While evaluating the application, no such anomaly was noticed in the response time curves for the remaining tasks. This is because these tasks are not resource expensive as adding an instance. For adding an instance, the API has to call AWS CLI to launch an EC2 instance and wait until the instance has been launched and started. Furthermore, the API has to execute multiple shell commands on the cluster nodes, which makes this task time consuming. But, the same is not true for the other tasks. The rest of the tasks need to execute a single command on the master node only, and hence, doesn't require excessive computing resources or connecting to other nodes in the cluster.

From the distribution for response time for launching an instance in the Figure 10, it can be said the response time for the same task can be improved. The following list provides some of the possible solutions to improve the response time and avoid the anomaly in the distribution curve:

- The RESTful APIs should be running on a separate node which will only be responsible to handle the API requests.
- Use a powerful or higher tier AWS EC2 instance for the master node and for the RESTful API node, if running on different nodes.
- Improve the RESTful APIs to include caching so that the API doesn't have to read the log files every time they receive a request.
- Improve the RESTful APIs to handle the request asynchronously and process the requests in parallel.

To conclude the evaluation section, it can be said this application proves to be a useful application to not just understand the Hadoop cluster better, but also to perform various tasks on Hadoop cluster efficiently and with ease of use.

6 Conclusion and Future Work

This paper proposed a mixed-reality application, HoloHDFS, that allows a user to visualise the architecture of HDFS, analyse and provide a heat map for under-/over-utilised resources using CPU usage of each node and block replication metadata. In addition, the evaluation results proved that the application performs all the purposed functionalities as expected and can perform simultaneous requests and provides 0% request error rate. The application is supported for Microsoft HoloLens devices and can run simultaneously on multiple such devices.

Even though, all the requests received by the application were processed by the master node during the evaluation, there's a variation in the distribution of the response time for adding an instance to the Hadoop cluster. This variation could be a result of using lower AWS EC2 type instance as master node and because of running the API on the same node as master node. Hence, it can be said there is scope of improvement in the response time for the requests made from the application to the RESTful APIs.

Furthermore, the features of HoloHDFS application can be extended to include visualisation of live MapReduce jobs and analyse the live performance. In addition, the application hasn't been tested with real-life users. Therefore, as part of future work, the interface of the application can be further enhanced for better user-experience and the application can be tested for system usability to prove the impact of mixed-reality application in the education and training industry.

References

Barbieri, L., Bruno, F., Cosco, F. and Muzzupappa, M. (2014). Effects of device obtrusion and tool-hand misalignment on user performance and stiffness perception in visuo-haptic mixed reality, *International Journal of Human-Computer Studies* **72**(12): 846 – 859.

Bertram, J., Moskaliuk, J. and Cress, U. (2015). Virtual training: Making reality work?, *Computers in Human Behavior* **43**: 284 – 292.

Dean, J. and Ghemawat, S. (2004). Mapreduce: Simplified data processing on large clusters, *Proceedings of the 6th Conference on Symposium on Opearting Systems Design & Implementation - Volume 6* **6**: 10–10.

Ghemawat, S., Gobioff, H. and Leung, S.-T. (2003). The google file system, *SIGOPS Oper. Syst. Rev.* **37**(5): 29–43.

Kade, D., Akşit, K., Ürey, H. and Özcan, O. (2015). Head-mounted mixed reality projection display for games production and entertainment, *Personal Ubiquitous Computing* **19**(3-4): 509–521.

Ke, F., Lee, S. and Xu, X. (2016). Teaching training in a mixed-reality integrated learning environment, *Computers in Human Behavior* **62**: 212 – 220.

Knecht, M., Traxler, C., Mattausch, O. and Wimmer, M. (2012). Reciprocal shading for mixed reality, *Computers and Graphics* **36**(7): 846 – 856. Augmented RealityComputer Graphics in China.

Knecht, M., Traxler, C., Winklhofer, C. and Wimmer, M. (2013). Reflective and refractive objects for mixed reality, *IEEE Transactions on Visualization and Computer Graphics* **19**(4): 576–582.

Knoerlein, B., Székely, G. and Harders, M. (2007). Visuo-haptic collaborative augmented reality ping-pong, *Proceedings of the International Conference on Advances in Computer Entertainment Technology* pp. 91–94.

Kronander, J., Banterle, F., Gardner, A., Miandji, E. and Unger, J. (2015). Photorealistic rendering of mixed reality scenes., *Computer Graphics Forum* **34**(2): 643 – 665.

Mateu, J., Lasala, M. J. and Alamn, X. (2014). Virtualtouch: A tool for developing mixed reality educational applications and an example of use for inclusive education., *International Journal of Human-Computer Interaction* **30**(10): 815 – 828.

Nakevska, M., van der Sanden, A., Funk, M., Hu, J. and Rauterberg, M. (2017). Interactive storytelling in a mixed reality environment: The effects of interactivity on user experiences, *Entertainment Computing* .

Olston, C., Reed, B., Srivastava, U., Kumar, R. and Tomkins, A. (2008). Pig latin: A not-so-foreign language for data processing, *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, ACM, New York, NY, USA, pp. 1099–1110.

Ortega, M. and Coquillart, S. (2005). Prop-based haptic interaction with co-location and immersion: an automotive application, pp. 6 pp.–.

Plantenga, T. D., Choe, Y. R. and Yoshimura, A. (2012). Using performance measurements to improve mapreduce algorithms, *Procedia Computer Science* **9**: 1920 – 1929.

Ricci, A., Piunti, M., Tummolini, L. and Castelfranchi, C. (2015). The mirror world: Preparing for mixed-reality living, *IEEE Pervasive Computing* **14**(2): 60–63.

Sheu, R.-K., Yuan, S.-M., Lo, W.-T. and Ku, C.-I. (2014). Design and implementation of file deduplication framework on hdfs, *International Journal of Distributed Sensor Networks* **10**(4): 561340.

Taylor, R. C. (2010). An overview of the hadoop/mapreduce/hbase framework and its current applications in bioinformatics., *BMC Bioinformatics* **11**: 1 – 6.

Wegman, E. (2000). Affordable environments for 3d collaborative data visualization, *Computing in Science Engineering* **2**(6): 68–72, 74.

Wu, Y., Ye, F., Chen, K. and Zheng, W. (2014). Modeling of distributed file systems for practical performance analysis, *IEEE Transactions on Parallel and Distributed Systems* **25**(1): 156–166.

A Configuration Manual

This configuration manual provides a step-by-step guide to replicate the solution developed in this research project and run the HoloHDFS application. This manual is divided into two subsections: Setting up the Hadoop Master Node and Setting up the Application Environment. To replicate this research project, you will need the source files that are available at `https://github.com/sarthak0403/HoloHDFS` public GitHub repository. You can choose to either download the project files from the repository or you can clone the GitHub repository.

A.1 Setting up the Hadoop Master Node

This research project requires Hadoop version 2.8.0 along with Python 2.7 framework and AWS CLI installed and configured on the master node. All the nodes in the Hadoop cluster for this research project are running on AWS EC2 t2.micro instance type with Ubuntu 16.04 LTS version. However, you can choose any other AWS instance type and a compatible Ubuntu version. To get the Hadoop master node ready for the processing, you need to perform the following steps:

1. Install and set up a Hadoop cluster using Ubuntu operating system with at least two worker/data nodes and change their hostname as: hadoop-master, hadoop-worker1 and hadoop-worker2. Since, this research project is dependent on AWS CLI to dynamically launch and destroy instances/nodes from a cluster, you will need to set up the Hadoop cluster on AWS cloud. Also, while installing Hadoop, you need to make sure you use “hduser” as the default username for Hadoop cluster nodes.

2. While creating the nodes on the AWS cloud, you will need to use a private key, therefore, it’ll will be extremely helpful if you use the same key for all the AWS EC2 instances. Now, copy this private key, say “hadoop-master.pem”, to the “/home/hduser/.ssh/” directory of all the instances and make sure this private key file is set as default key for all the ssh connections amongst each other.

3. You will have to make sure at least the master node accepts public traffic so it can be used to run RESTful API which is essential for this project.

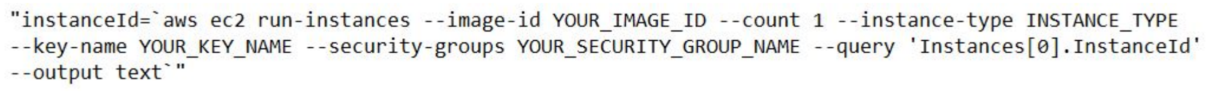
4. Create an image of the hadoop-worker1 or hadoop-worker2 instance and note the image-id which will be required in later steps to quickly launch an instance and add it to the Hadoop cluster.

5. After setting up Hadoop cluster, you need to copy the files in the “Hadoop-master Directory Files” folder under the root folder of the GitHub repository to the “/home/hduser/” folder on the master node (hadoop-master).

6. Replace the image-id and key name that you created/copied in step 2 and 4 and insert meaningful values for security group and instance type in Bash the command given in the Figure 1.

7. Now, open the file “launchAWS.sh” from “/home/hduser/” directory, and replace command in line 4 with the command in Figure 1 without double quotes.

8. Create a directory in the HDFS root folder and copy a sample text file that will be used by the WordCount MapReduce job. To do this, perform the following steps:



```
"instanceId=`aws ec2 run-instances --image-id YOUR_IMAGE_ID --count 1 --instance-type INSTANCE_TYPE
--key-name YOUR_KEY_NAME --security-groups YOUR_SECURITY_GROUP_NAME --query 'Instances[0].InstanceId'
--output text`"
```

Figure 1: The Bash Command to Launch an Instance

(a) Execute the following command in the terminal: "hdfs dfs -mkdir /user"

(b) Execute the following command in the terminal: "hdfs dfs -mkdir /user/hololens"

(c) Execute the following command in the terminal: "hdfs dfs -mkdir /user/hololens/input"

(d) Copy any text file in the "/user/hololens/input" HDFS directory using: "hdfs dfs -put ⟨local-src-path⟩ /user/hololens/input"

To start the Python RESTful service through which the HoloHDFS application will be communicating with the Hadoop master node, perform the following steps:

1. Login to the hadoop-master terminal and create a directory "/home/hduser/api". Copy the "HoloHDFS/Hadoop-master Directory Files/api/api.py" Python script from the GitHub repository to the "/home/hduser/api/" directory.

2. Open the "/home/hduser/api" directory in the hadoop-master terminal and run the following commands one-by-one:

   (a) virtualenv flask

   (b) flask/bin/pip install flask

   (c) chmod a+x api.py

   (d) source flask/bin/activate

   (e) pip install psutil

   (f) ./api.py

   Note: This service runs on port 5000 by default. Make sure the port number 5000 is open and is not already in use by any other process on hadoop-master node.

3. The API should be running now using the public IP of the AWS instance. Test the API using the following URL:
   `http://PublicIPofhadoop-master:5000/workers`
   You should get a list of all the workers in JSON format as a response.

A.2 Setting up the Application Environment

This section will guide you to set up the environment to run the HoloHDFS application on a Microsoft HoloLens emulator. First, you need to make sure your system meets the minimum system requirements as per the guide provided by Microsoft at `https://developer.microsoft.com/en-us/windows/mixed-reality/install_the_tools#hololens_emulator`

Next, follow the below checklist to make sure you have the necessary Software Development Kit (SDK) and software to open and run the HoloHDFS applicaiton:

- Install Visual Studio 2017 with Universal Windows Platform development and Game Development with Unity plugins.
- Make sure Hyper-V is enabled on your system.
- Install HoloLens Emulator build 10.0.14393.1358 or later, if available.
- Install Unity 5.6.1f1 or later version.

Note: All the above mentioned tools are available for free.

After installing the required software and SDKs on a compatible system, perform the following steps to run the HoloHDFS application:

1. Copy the “HoloHDFS/HoloHDFS Unity Project/” folder from the GitHub repository to your local machine.
2. Open the Unity software on your machine.
3. Click on “Open” button on the home screen. A “Open Existing Project” window will appear.
4. Now, navigate to the “HoloHDFS/HoloHDFS Unity Project/” directory on your local machine, select the “HoloHDFS Unity Project” folder and click the “Select Folder” button.
5. Once the project opens in Unity editor, click on the “Scripts” folder under the “Assets” folder in the “Project” window in Unity editor.
6. Next, you need to change the RESTful API URL in the following script files to call the right APIs:
   - Open the “Assets/Scripts/JobManager.cs” file and update the IP address in the “url” variable to match the IP address of your hadoop-master in line 23 and 31. It should look similar to: url = “http://0.0.0.0:5000/startjob”;
   - Open the “Assets/Scripts/NodeManager.cs” file and update the IP address in the “url” variable to match the IP address of your hadoop-master in line 23. It should look similar to: url = “http://0.0.0.0:5000/launch”;
   - Open the “Assets/Scripts/NodeSelector.cs” file and update the IP address in the “url” variable to match the IP address of your hadoop-master in line 91 and 125. It should look similar to: url = “http://0.0.0.0:5000/launch”;
   - Open the “Assets/Scripts/NodeSelector.cs” file and update the hadoop-master node private and public IP address in line 110. Similarly, change the private IP address of the hadoop-worker1 and hadoop-worker2 node in line 116.
   - Open the “Assets/Scripts/WorkersData.cs” file and update the IP address in the “url” variable to match the IP address of your hadoop-master in line 39. It should look similar to: url = “http://0.0.0.0:5000/workers”;
7. Open the Build Window using the “HoloToolkit –⟩ Build Window” menu on the menu bar at the top of the Unity editor window, and click on the “Build Visual Studio SLN” button in the Build Window.

8. Minimise the Unity editor window and navigate to the “HoloHDFS/HoloHDFS Unity Project/App/” directory on your machine and open the “HoloHDFS/HoloHDFS Unity Project/App/HoloHDFS.sln” project file in Visual Studio 2017.

9. Once the application project has finished loading in Visual Studio 2017, ensure the “Platform” is set to “x86” from the menu bar.

10. Ensure the “Target Device” is set to “HoloLens Emulator” from the menu bar.

11. To run the application on the emulator, go to “Debug –⟩ Start Debugging” or press F5 key and the emulator will start launching.
    Note: In case, the application fails to run the first time you open the emulator, you’ll have to run the application again either using “Debug –⟩ Start Debugging” or pressing F5 key. Also, if you make changes in the application, you’ll have to reboot the emulator before deploying the application again to the emulator.