FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Data-Parallel Transcoding for the 3D Internet

Al-Waleed Shihadeh

FAKULTÄT FÜR INFORMATIK

DER TECHNISCHEN UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics

Data-Parallel Transcoding for the 3D Internet

Parallel Transcoding-Daten für die 3D-Internet

Author:	Al-Waleed Shihadeh
Supervisor:	Prof. Dr. Florian Matthes
Advisor:	Bernhard Waltl, M.Sc.
Date:	November 05, 2014

I assure the single handed composition of this master's thesis only supported by declared resources.

Munich, November 05, 2014 Al-Waleed Shihadeh

Abstract

Building a 3D web collaboration environment is a huge challenge as it involves the development of a high performance web application which allows engineers to view and exchange 3D models over the Internet. This is considered to be a huge challenge because the current status of the 3D Internet is still evolving, as well as the complexity of loading large 3D files into the browsers. Fortunately, this objective can be achieved by using current 3D technologies, like X3DOM and X3D. However, when working with large 3D models, it is not enough to just rely on the performance of these technologies to build a high performance application. Large 3D models consume a large amount of memory and therefore, rendering these models will take a long time. One way to enhance the performance of rendering 3D models is to transcode the geometries of these models to binary files. In addition, the transcoding process will enhance the interactions between the 3D models and the users since the size of these models is much smaller. To minimize the required transcoding time, a data-parallel transcoding approach should be used. The aim of this thesis is to investigate the possibility of developing a data-parallel transcoding system based on the Hadoop framework. The outcome of this thesis is a web data-parallel transcoding system that provides the end users with a 3D data-parallel transcoding service and a 3D model viewer. The implemented system is a distributed application that allows engineers to upload a single X3D document, transcode it to binary format and deploy it on the web. In addition, this thesis provides an evaluation and performance analysis of the implemented transcoding approach. The main outcome of the evaluation part is that using a data-parallel approach for transcoding 3D data will dramatically reduce the required transcoding time compared to sequential transcoding approaches. In addition, the main properties that influence the required transcoding time are the shapes number, files number, split size and cluster nodes number.

List of Figures

List of Tables

Contents

Part I.

Introduction and Theory

1. Introduction

1.1. Overview

Building 3D models requires experience with Digital Content Creation (DCC) tools. DCC tools can be a modeling system such as CATIA[1], 3D Studio Max[2] and Maya[3]. It can also be a computer-aided design system (CAD), simulation application or analysis tool. The functionality of these tools is diverse from the creation and design of 3D models to the simulation and rendering of the 3D models. For instance, CATIA is a 3D Product Lifecycle Management application suite that enables engineers to create 3D parts, design advanced geometry shapes based on a combination of extensive multiple surface features and wire-frame with full specification capture, and perform analysis and simulation directly on the 3D design [48]. Furthermore, it provides engineers with a set of tools for creating and manipulating mechanical surfaces in the design of complex parts and products. These tools include wire-frame elements such as angle, plane, curve, spiral sphere, etc....

On the other hand, there are many stakeholders who are involved in the process of building 3D models. Each stakeholder has its own responsibilities and capabilities for building the 3D model. System architects are responsible for designing the high level design of the 3D model without specifying the details of the parts. System designers develop the detailed design for the 3D parts of the digital mockup (DMU) using different CAD tools. DMU integrators are responsible for integrating the 3D parts designed by the designers into a single DMU. Domain experts are professionals performing complete performance analysis of the DMU or the parts of the product. For example, domain experts perform thermal analysis to find the effects of the temperature on the marital used in the product, structure analysis to find the effects of forces, deformations, or accelerations on the physical structures and components of the DMU, and electromagnetic analysis of the DMU.

Usually the stakeholders do not use the same DCC tools and that's for several reasons. One reason is that these DCC tools are very expensive and not every department or company can offer such a budget for buying licenses of DCC tools. Another reason is that there are DCC tools that are specialized in a specific domain and provide the user with features that are not provided by any other tool. For instance, CATIA provides a basic analysis extension that allows domain experts to perform a simple analysis and simulation tasks; however, domain experts need sophisticated tools for performing the thermal, structure and electromagnetic analysis of the DMU. One more reason is that some of the engineers

[1]http://www.3ds.com/products-services/catia/welcome/ Retrieved on 2014-10-24
[2]http://www.autodesk.com/products/3ds-max/overview Retrieved on 2014-10-24
[3]http://www.autodesk.com/products/maya/overview Retrieved on 2014-10-24

prefer to work with a specific tool or others do not have the required experience to work with a specific tool.

During the 3D product development process communication and DMU data exchange between different stakeholders plays a major role in the final 3D model. For instance, domain experts and DMU integrates rely on DMU designed by the designers for generating analysis models. On the other hand, designers depend on the feedback of the domain experts and the results of their analysis to modify the DMU. Sharing the DMU is not an easy and simple process. The difficulty in sharing 3D DMU's between stakeholders is due to several reasons.

First, not all stakeholders have access to the same DCC tools due to the cost of DCC licenses. For example, there is a probability that designers and domain experts use a completely different set of tools. Second reason is that a 3D DMU is usually very large in size, which is more than one GB and therefore managing and sharing these files in a file server includes some difficulties especially when there are multiple versions of the same DMU. The third reason is security, sharing a DMU with the supplier or even domain experts may involve hiding information or sharing only specific parts of the DMU.

Another important issue in the 3D Model development process is the integration of the products from different DCC into one DMU. Each of DCC tools has its own file format, these file formats are proprietary formats and therefore the only way to open or modify these files is using their original DCC tool. However, Some of the DCC tools and other specialized tools such as polyTrans [30] enable engineers to convert their 3D DMU to a variety of file formats including some open-source file formats such as X3D [17] and VRML [18]. Such tools ease the integration and sharing process between stakeholders and provide the engineers with the ability to build their own applications for data exchange, viewing and even manipulating the DMU outside their own DCC tool.

1.2. The Vision

Due to the reasons stated above, namely the expensive cost of DCC tools, data exchange difficulties and integration difficulties Airbus Group vision is to build a web-based collaborative engineering environment where the key entry point and reference for this environment is the DMU. The main objective of this environment is to support the communication and data exchange between different stakeholders within and outside AirBus Group. Furthermore, the implemented environment must allow the end users to access and view DMU with low cost prices. There are no DCC tool licenses required for using the environment, a web browser is the only requirement for using it.

Figure 1.1 presents the core concept of the proposed collaboration environment. Different stakeholders such as domain experts, suppliers, and designers can connect to a web platform using their own browser client to perform one of the flowing functionalities.

1. Add new 3D model to the platform.

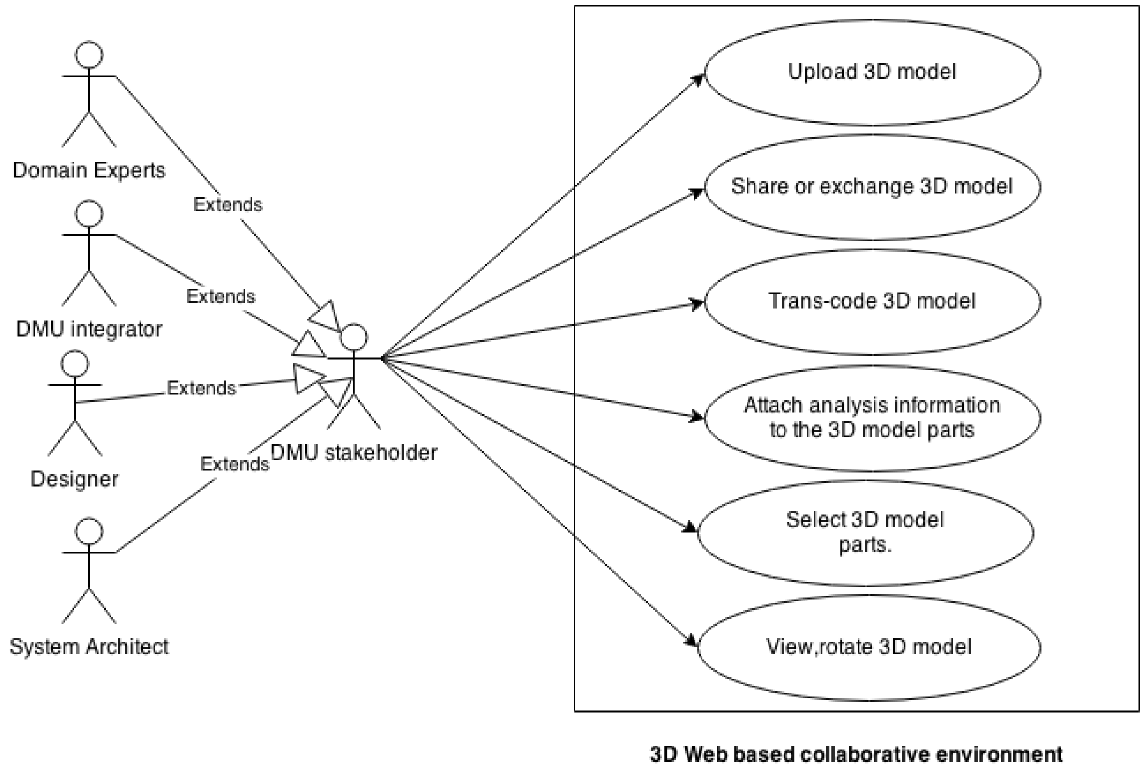

Figure 1.1.: Web based collaborative environment

2. Exchange a 3D model with other stakeholders.

3. Attach analysis information to specific parts of the 3D model.

4. Open a 3D model in different views such as design view and thermo analysis view.

5. Dynamically interact with the 3D model in a web browser and view it from different angles.

The current status of 3D Internet and today's technology supports and provides tools for developing such an environment. For bringing the 3D DMU into a web browser, there are a lot of options such as X3DOM [60] and O3D [28]. However, there are many challenges in order to develop a reliable, stable and efficient web 3D platform [42, 10]. For instance, performance is a key concept in such an environment and it also affects the usability of the platform. Another challenge is the richness of the user experience with the platform. Even if the platform is very powerful and provides a lot functionality; that does not mean that the platform will be used by the stakeholders. A friendly user interface that eases the process of sharing and viewing 3D models is still very important for such platforms.

1.3. Motivation

1.3.1. Problem

Frequently, DMU files are very large, and in most cases are larger than one GB. Therefore, loading DMU as a single unit into a web browser is not applicable to implement an efficient 3D runtime and collaboration environment. To find an optimal approach for rendering and loading large DMU's into a web browser window, we investigate and evaluate the current technologies that enable us to achieve our goals. As a result, we found a reasonable approach that increases the efficiency of the environment and decreases the loading time into the web browser. For simplicity reasons, the proposed approach assumes that we have a representation for the DMU in X3D format. This way, DMU's developed using different CAD applications will be rendered using the same method. In addition, commercial CAD applications use proprietary format to store 3D data. As a result, we do not have any clue how the 3D objects are represented in these files.

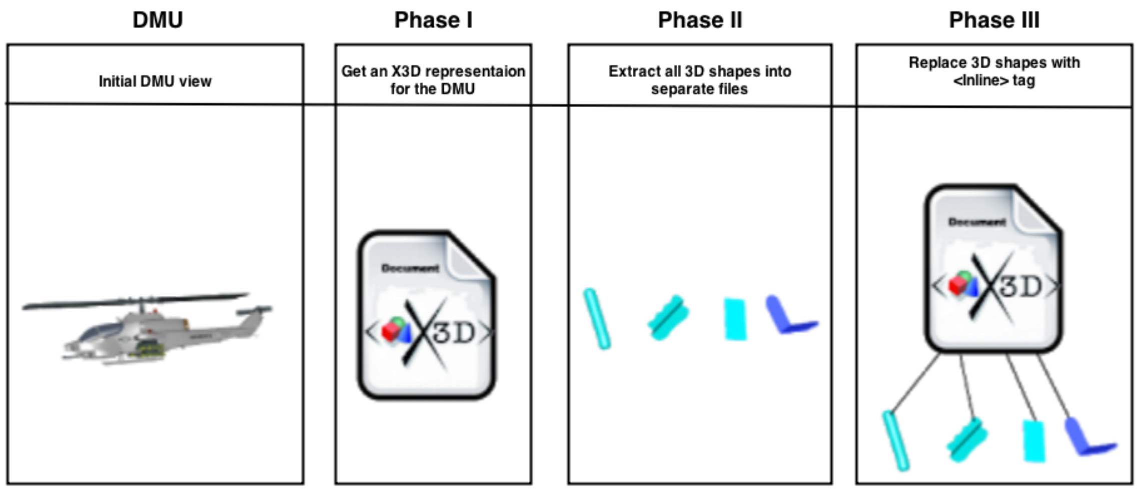
DMU
Phase I
Phase II
Phase III
Initial DMU view
Get an X3D representaion for the DMU
Extract all 3D shapes into separate files
Replace 3D shapes with <Inline> tag

Figure 1.2.: DMU preperation phases

Table 1.2 presents the three phases of the proposed approach. **Phase I**: deals with the conversion process of DMU from its proprietary format to the declarative X3D format which is an XML-based file format. There are many tools on the market that enable us to perform this step easily and one popular tool is polyTrans[4]. **Phase II**: extracts all 3D geometry shapes and stores each shape in a different X3D file. The aim of this phase is to partition the 3D model into small files that can be loaded into a web browser. **Phase III**: replaces all the shapes in the original X3D file with the *inline* tag. The inline tag is simply a reference to the corresponding shape file extracted in **Phase II**. One important feature of the *inline* tag is that it enables the browser to load the 3D geometry in the background without locking the main thread of the browser.

[4]http://www.okino.com/conv/conv.htm Retrieved on 2014-10-24

Breaking down the DMU into several X3D files has many benefits. First of all, loading DMU files is no longer done in one process; Browsers will first load the main X3D file, which contains the references for all shapes of the DMU. The next step is to start loading the X3D shapes in background jobs. This feature will enable the end user to interact with the DMU even before the browser completes loading the whole DMU. Another benefit for extracting shapes from the DMU is that we can create a web resource for every single shape in the DMU. Creating web resources for 3D shapes will ease the process of sharing and data-exchange of 3D data between different stakeholders.

The proposed approach helps us in improving the efficiency of the environment; however, since the 3D shapes for DMU is often very large, browsers crashes when they try to load such shapes. Furthermore, the interaction with a very large DMU is still very difficult and very slow. Therefore, another idea rises to improve this approach, which is transcoding X3D shapes into binary formats, and that will compress the X3D shapes and reduce their sizes dramatically. For instance, the shape's size of the Ariane 5 ME cylinder [2] is reduced from **84** MB to **20** MB. The transcoding process must not lead to any data loss from the original DMU data.

Extracting large DMU introduces a very large number of X3D shape files. For instance, a small 3D model for the Ariane 5 ME cylinder with the size **84** MB contains **873** shapes. The number of shapes in a DMU depends on the level of details that the model implements. DMU's that implement a very detailed level of the 3D object contains shapes larger than those which represent a high level abstraction of the 3D object.

Transcoding this large number of X3D files to a binary format in a sequential way is time consuming and is an inefficient method. For instance, transcoding the Ariane 5 ME which includes only **873** X3D files using python script in a single computer with 8 GB of RAM and 2.93 GHz processor took more than two days. The Ariane 5 ME is a very simple model that does not contain detailed information. Detailed models contain a larger number of shapes and will probably require much more time to complete the transcoding of all their shapes.

Our main problem at this stage is the time required for transcoding the X3D files. This problem will prevent us from building a high performance 3D collaboration environment; Since the DMU is the key entry point for the environment and it requires an endless time to be prepared for the usage of a web browser. For the reasons stated above, and mostly the time required for transcoding , we decided to use a data-parallel approach for transcoding the X3D files into a binary format. The transcoding approach should ensure a reduction in the processing time required for the transcoding process.

1.3.2. Goals

The main goals of this thesis are to investigate the idea of transcoding 3D models data represented in X3D/XML format in parallel using a MapReduce environment and to develop a transcoding system that scales reasonably well on commodity clusters. In order to achieve these goals there are several requirements and tasks that must be achieved. These requirements vary from data analysis tasks to web development and evaluation tasks. A list of the high level requirements and tasks are presented below.

- Understand the X3D dataset and extract a general structure of the dataset. Understanding the dataset will help us in defining a general structure of the 3D models which can be used later in defining and implementing a strategy for partitioning the X3D/XML data.
- Develop a method for partitioning X3D/XML data. X3D data format is an XML-based data format which means that the partition method must not lead to any loss in the data or change of the hierarchy of the model. One more issue addressed here is partitioning large X3D/XML files and therefore parallelizing the partition process could be considered.
- Build several Hadoop MapReduce environments for both developmental and evaluation purposes. These Hadoop clusters will be used to evaluate the effects of environment parameters on the transcoding approach. This step includes installing and configuring a Hadoop server on different machines with different hardware specifications. It also includes the installation and setup of transcoding tools, Hadoop APIs, and third party libraries that are used in the development of MapReduce jobs.
- Develop MapReduce jobs for transcoding X3D/XML data. The MapReduce jobs are responsible for parallel transcoding the 3D geometry data in a 3D model to binary data. The transcoding process is already implemented by a third party tool and therefore, the main focus of MapReduce jobs is to parallelize the transcoding of partitioned X3D/XML data.
- Develop a Web application for automating the transcoding process and viewing the 3D models. The web application will provide an interface for the end user that enable the below functionality.
  1. Upload X3D/XML data to a webserver.
  2. Partition the uploaded X3D/XML data.
  3. Move X3D/XML partitioned data to Hadoop cluster.
  4. Run MapReduce job to transcode the X3D/XML data.
  5. Copy the transcoded data to a web server and notify the end user.
- Evaluate the performance and the scalability of the system: the evaluation of the system will cover the effects of hardware specification and different dataset parameters. For instance, the effects Hadoop cluster node number or 3D model size.

1.4. Thesis Content

This thesis consists of three main parts. First of all, the introduction and theory parts, which give the reader a clear overview of the thesis topic, also they contain detailed information regarding the definitions, concepts, and technologies which are used throughout the thesis work. It starts by introducing three-dimensional models, it also explains how these models are built, furthermore, it illustrates the different tools that are used for creating and manipulating 3D models, and several representations for the 3D models. Part one continues with an overview of the current status of 3D Web. This overview will give the reader a clear idea of the core technologies that are used to view 3D models in a browser window such as Google Chrome and Mozilla Firefox. The next part illustrates in detail all the technologies that were used in the implementation phase.

The contribution of the thesis part presents the related work achievements. It contains three chapters. The first chapter presents the theory behind the proposed data-parallel approach and illustrates the phases and architecture of the system. Chapter two contains information about the environments that are used during the implementation and evaluation phases. This chapter will provide information regarding the hardware and software specification of the used environments. The last chapter in this part provides detailed information regarding the implementation steps of the proposed data-parallel approach.

The evaluation and performance analysis part includes two chapters. The first chapter presents the evolution results of the implemented system into several environments with different configurations. The main aim of this chapter is to give the reader an overview of the performance of the implemented approach and to compare it with different solutions. The second chapter discusses the limitations of the proposed approach and suggests areas which require future work.

The main contribution of this thesis can be summarized into two main items.

1. Definition and implementation of a data-parallel transcoding system for X3D/XML data.
2. Experimental evaluation and performance analysis of the system.

2. Foundation

2.1. 3D Design Introduction

2.1.1. What is 3D Modeling?

3D modeling is a process that aims to create three-dimensional representations of a surface or a real object. This process can be realized either manually or using sophisticated 3D software for creating 3D models such as AutoCAD or CATIA. 3D modeling process involves manipulating and defining polygons, edges, and vertices of the modeled object in the 3D space. Many different fields rely on 3D modeling in order to enhance the understanding and visualization of their core concepts and objects. For example, game development, commercial advertising, entertainment design, film and engineering are fields that use 3D modeling for repressing real objects.

The output of 3D modeling process is called 3D model, which is the three-dimensional representation of the real object. The 3D model can be displayed as two-dimensional image using a process called 3D rendering. In addition, it is feasible to create a physical representation of the 3D model using a 3D printer.

Engineers usually use one of the main four modeling methods; which are polygonal, primitive, non-uniform rational B-spline, or splines and patches [6]. Polygonal modeling implicates linking and joining line segments through points in the 3D space. These points and lines represent the edges and coordinates of the modeled object. Points that connect line segments are called vertices. One of the main advantages of polygonal modeling is that the generated models are very flexible and can be rendered quickly on a computer screen. However, using this method it is very difficult to generate and render precise curved surfaces.

Primitive modeling is another modeling method which is used mainly in building 3D models of technical application [6]. This method is based on the mathematical representation of object coordination and therefore, the forms defined by this method is more precise than models generated by polygonal modeling . It is also easy to learn even for beginners and that's because it uses basic geometry shapes in defining 3D models such as cones, cubes and spheres.

Non-uniform rational B-spline modeling method also called NURBS method is one of the best options for engineers to build realistic curved smooth surfaces [6]. In contrast to polygonal modeling which cannot generate precise curves, this method can **bend** the 3D space to generate smooth surfaces. For this reason, most of the platforms and tools support this method of 3D modeling [6].

The last method is the splines and patches modeling method, which is the most advanced method for generating 3D models. It enables engineers to identify and project the visible surface of the modeled object using curved lines. 3D models generated using this mode seem to be the most realistic and life-like 3D models [6]. The only drawback of this model is that it takes more time to build and execute commands on the 3D model.

Choosing one modeling method over another is a matter of the design goals and the software capabilities. As mentioned above it is easier to build 3D models using applications that are developed particularly for the 3D modeling purposes. There are many applications on the market these days; most of them are commercial applications which are very expensive. However, there are some open-source programs that are available for free. Most of these applications support both polygonal and NURBS modeling methods and some of them support complex environmental objects such as rain, clouds and blowing sands [6].

2.1.2. Tools used to Build 3D Models

There are tens of software applications and tools which offer various functionalities for 2D and 3D designs. These programs range from consumer-based packages such as Sketch up Pro[1] to professional applications like AutoCAD[2] and CATIA[3]. Furthermore, some of these applications focus on a specific domain while other applications provide a general environment design and simulation of 3D models.

Choosing a suitable CAD application is not an easy job. The customer needs to put into consideration several aspects and features of these tools. For instance, when building a 2D and 3D shapes, what are the supported **Design Tools** used in the application. Also, which editing tool does the application support. Point markers, layer managers, and snap tools are important editing tools that help the user keep the design organized. In addition, the customers should consider the supported interface features, which provide the user with command line and the ability to import existing designs. One major aspect of these tools is the price. Usually these applications are very expensive; however, applications such as Turbo CAD Deluxe are cheap and can be afforded by individuals. Moreover, when buying a CAD program customers need not to forget that not only the technical features supported by the application are important, but also other features such as the usability, ease-of-use, help and support and compatibility should be considered.

This section presents a set of tools and applications that allow designers, engineers, architects and domain experts to build and customize designs from the ground up. The section will end by a brief comparison between the presented CAD tools. The section does

[1]http://www.sketchup.com/products/sketchup-pro Retrieved on 2014-10-14
[2]http://www.autodesk.com/products/autocad/overview Retrieved on 2014-10-14
[3]http://www.3ds.com/products-services/catia/welcome/ Retrieved on 2014-10-14

not cover all the CAD programs that exist on the market. It only describes a sub set of the most common CAD applications. These application were selected based on their popularity.

BricsCAD Classic

BricsCAD Classic[4] is a simple CAD application that provides a powerful set of tools for building 2D drawings and 3D models. BricsCAD supports engineers with tools such as hatching tools, photo realistic rendering and a layer manager tool. It also allows users to view their 2D or 3D designs as wireframes or as fully rendered objects. Although BricsCAD Classic includes many features and tools for 3D modeling, it still misses many important architectural tools such as a wall tool or a house wizard. BricsCAD Classic is compatible with the most important file formats for CAD designs. Both DWG[5] and DXF[6] are supported, along with many other popular file formats.

DesignCAD 3D Max

DesignCAD 3D Max[7] is a CAD application that focuses on 2D and 3D modeling for architectural purposes. Nevertheless, it can be used for various types of projects. DesignCAD 3D Max includes drawing and modeling tools that help the user build more realistic 3D models and ease the 3D modeling process. For instance, the wall tool helps users to construct each individual wall within the house in a comfortable and convenient way. However, a house tool that helps users in creating an entire building or house is missing from the application. Other tools such as hatching, texture, transparency options, and photo realistic rendering capabilities are tools that help engineers to build more realistic 3D models and improve the visualization and lightings of the 3D model. Furthermore, DesignCAD 3D Max offers more than 600 complementary models, and therefore users are not forced to start from scratch each time they want to build a 3D model, they can simply add these models to their designs to construct new designs.

TurboCAD Deluxe

TurboCAD Deluxe[8] is a commercial CAD application for beginners. It provides a lot of design tools to help users build their 3D models in an easy way. For instance, for architectural purposes TurboCAD Deluxe provides tools; such as, house and wall tools that help the user in the process of building architectural designs. These tools guide the user through wizards for defining, selecting and modeling the properties of the modeled house. TurboCAD Deluxe wizards enable the user to select the number of rooms, room types and dimentions. TurboCAD Deluxe help users to build more realistic models that look like real objects instead of wire-frames. TurboCAD Deluxe also has a wide range of editing tools for editing 2D and 3D designs such as snap tool, which is used to move elements to

[4]https://www.bricsys.com/en_INTL/bricscad Retrieved on 2014-10-14
[5] a binary file format used for storing 2D and 3D designs
[6]CAD data file format developed by Autodesk
[7]http://www.imsidesign.com/Products/DesignCAD Retrieved on 2014-10-14
[8]http://www.turbocad.com/ Retrieved on 2014-10-14

precise locations for easier modeling. However, TurboCAD Deluxe does not support 3D animation. TurboCAD supports 28 industry-standard formats including PDF and DWG. It also allows the user to import and export 3D models from and to other CAD applications.

Autodesk Products

Autodesk, Inc[9] is an American software company that builds software applications for many fields, such as engineering, construction, game development, and manufacturing. Autodesk, Inc. has a variety of CAD products. Below is a description of three different Autodesk products: AutoCAD, Maya and Autodesk 3ds Max.

AutoCAD

AutoCAD[10] is a CAD application that provides a 2D and 3D design and drafting functionalities. AutoCAD is developed and marketed by Autodesk Inc. The main objective of AutoCAD is to help engineers create 3D models for buildings, products and public spaces without using papers. AutoCAD was one of the first CAD applications that are available on personal computers [51].

The basis of AutoCAD's first release is a legacy system called Interact [62] and it supported only basic entities such as lines, polygons, arcs and text for constructing more complex objects. The following releases started to support more complex entities, provide users with more features, and an advanced programming interface developed using C++ programming language. Current version includes a full set of tools for 3D modeling that provides the engineers with a greater user experience with fast-moving, high-quality rendering [57]. Furthermore, current versions support programming languages interfaces such as VBA, .NET, AutoLISP, and Visual LISP. AutoCAD's native file format is called DWG (drawing) and it also supports Design Web Format (DWF), which is a file format introduced by Autodesk for publishing CAD data.

AutoCAD is used by many different users such as architects, engineers, and structural designers to model, and design buildings and other objects. Autodesk, Inc. targets both Macintosh and Windows platform and also offers multiple versions such as AutoCAD LT, which delivers drawing tools with a cost-effective price. AutoCAD has been used in many real projects around the world. For example, it has been used in the New York Freedom Tower [39] and Tesla electric cars [13].

Maya

Maya[11] is another CAD program that is developed by Autodesk, Inc. Maya was originally developed by Alias Systems Corporation. Unlike AutoCAD Maya focuses more on building and manipulating animated 3D models where AutoCAD does not support 3D animation. Maya relies on the natural and physical lows to control the behavior of the

[9]http://www.autodesk.com/ Retrieved on 2014-10-14
[10]http://www.autodesk.com/products/all-autocad Retrieved on 2014-10-14
[11]http://www.autodesk.com/products/maya/overview Retrieved on 2014-10-14

virtual objects in the 3D space. Maya improved 3D rendering by providing techniques and tools for rendering natural effects. Modeling and rendering effects such as objects movement caused by gravity, smoke blowing in a breeze, or rotation of clouds was very difficult to achieve before Maya. Furthermore, it provides tools for simulating the movement of objects in the 3D space. Autodesk, Inc. offers versions for both IBM-compatible and Macintosh operating systems. The main features that Maya supports are 3D rendering and imaging, dynamics and effects, 3D animation, 3D modeling, and pipeline integration [34].

Autodesk 3ds Max

Another CAD product provided by Autodesk, Inc. is Autodesk 3ds Max[12]. This application is defined as 3D computer graphics software for creating, manipulating and building 3D models, games and animations. People usually use Autodesk 3ds Max for 3D game development, movie effects and TV commercials [32]. Autodesk 3ds Max is only available on Windows platform and offers the following features UI, workflow, pipeline, dynamics and effects, 3D modeling and texturing, 3D rendering, 3D animation [33].

CATIA

CATIA is a commercial multi-platform CAD application suite. The term CATIA stands for Computer Aided Three-dimensional Interactive Application. CATIA was developed by a French company called Dassault Systemes, and marketed around the world by IBM. The first release of CATIA was in the late 1970s for the purpose of developing Dassault's Mirage fighter jet [44]. After that, many engineering domains adopted CATIA as a platform for 3D modeling. For instance, these days CATIA is used in aerospace, automotive, industrial machinery, electronics, building ships, plant design, and consumer goods.

CATIA is not only used for various engineering fields, but also by various stakeholders of the 3D modeling process such as designers, assemblers, architects, industrial engineers etc. Thousands of engineering companies around the world are rely on CATIA to develop 3D models for their products. Also, CATIA played a major role in the designing and developing process of various space equipment inside NASA [44]. CATIA is being used world wide, heavily used in North America, Europe and Australia. Also, it is increasingly being used by countries like India and Japan. The following is a short list of the engineering companies that use CATIA around the world: AirBus Group , Kelsey-Hayes , Boeing, Lear Jet , BMW, Volvo, Black and Decker, Fiat Peugeot, Northrop Grumman Corp, Ferrari, Lockheed Martin , Porsche , Daimler Chrysler, Goodyear, Freightliner Truck , Allied Signal , Sauber Formula, Volkswagen, Pratt Whitney, United Airlines, Toyota, Hyundai , Ford, Mercedes-Benz , Honda [44].

One reason why CATIA is spreading around the world; and is being used by a lot of engineering companies is that it is the only application in the market that is capable of providing customers with the tools for dealing with the complete product development

[12]http://www.autodesk.com/products/3ds-max/overview Retrieved on 2014-10-14

process, from the product concept specification to product-in-service [44]. It also provides engineers with tools for managing digital mock-ups, machining, analysis, simulation and it allows the reuse of the product design knowledge [50].

Summary

Building a 3D model is not an easy task and requires special software that enables engineers to design these 3D models. Today, there are many CAD applications in the market and most of these applications are commercial applications. The previous section highlighted some of these applications and showed the uses of each of the presented programs. The presented applications varies from simple CAD tools for personal or small business uses such as BricsCAD Classic to more advanced CAD platform like CATIA. Table 2.1 shows a brief summary of all the applications presented in the section.

2.1.3. 3D Model Representations

Representing 3D models is a key point in the 3D modeling process for several reasons. The most important reason is that the representation method affects the rendering and display of 3D models. Thus, 3D models must be represented in a structure that is suitable for rendering the 3D model and ease the interaction between the 3D model and the user. This section will illustrate some of the common methods for representing 3D objects. It will highlight mesh models, surface-edge-vertex models, generalized-cylinder models, and octree models. This section also presents some of the most common and standard file-format that is used to represent 3D models.

3D Mesh Models

A 3D mesh model is a simple representation for a 3D object. It represents the 3D objects by a collection of vertices, edges, faces, polygons and surfaces [49]. Figure 2.1 presents the elements of 3D mesh. The basic elements of 3D mesh are the vertices which represent points in the 3D space and edges which connect vertices to form a closed set of edges called faces. Most common used faces are triangle faces with three edges and quad faces with four edges [49]. A polygon is a coplanar[13] set of faces. Surfaces are groups of polygons; their aim is to group polygons to smooth the shading of the objects. 3D mesh can be used to represent objects in an abstract or a detailed way [49].

Surface-Edge-Vertex Models

Wire-frame is the most basic three-dimensional model. It only consists of edges and vertices [49]. This type of 3D models assumes that the surfaces of the modeled object are planar and it only has straight edges. One common generalization of wire-frame models is the surface-edge-vertex representation.

[13]Points or faces lie in the same geometric plane

Application	Company	Description
Autodesk 3ds Max	Autodesk, In	Modeling application offers a complete modeling, animation, rendering, and simulation solution for games, film, and motion graphics artists.
CATIA	Dassault Systeme	CATIA is a commercial multi-platform CAD application suite that is used in aerospace, automotive, industrial machinery and many other domains. CATIA is used around the world by many big companies such as AirBus and BMW. CATIA goes beyond an ordinary CAD application because it provides a comprehensive product development process.
Maya	Autodesk, Inc	Modeling and animation application that offers a creative feature set along with effects tool sets for 3D modeling. MAYA is mostly used by game developers.
AutoCAD	Autodesk, Inc	Professional CAD application and one of the first CAD applications. Usage of AutoCAD varies from creating simple drawings to model complex 3D models.
TurboCAD Deluxe	IMSI/Design, LLC	Commercial CAD application for beginners that provides a powerful set of tools for 2D and 3D modeling. It also supports various industry-standard formats for professional CAD applications.
DesignCAD 3D Max	IMSI/Design, LLC	CAD application that focuses on 2D and 3D modeling for architectural purposes. It offers wizard tools and built-in 3d models to make the 3D modeling process much easier for beginners.
BricsCAD Classic	Bricsys Inc.	Simple CAD software that provides the basic tools for building 3D models.

Table 2.1.: CAD applications summary

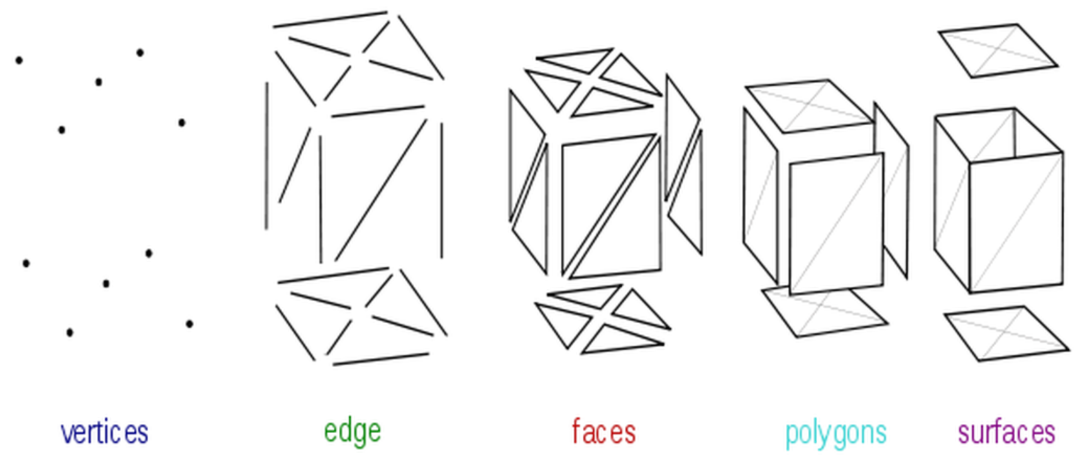

Figure 2.1.: Elements of mesh modeling [56]

Surface-edge-vertex is a data structure that consists of four elements; which are the vertices of the modeled object, the edge segments, the surfaces of the object and the topological relationships that define the location of the surfaces and vertices[49]. The special case *wire-frame* can be constructed from this generalization by letting the surfaces to be planner and the edges are in straight line segments. On the other hand, this representation includes curved edge segments and curved surfaces.

An object represented by this method has three components. First, the vertices which consists of 3D point and a collection of edges that meet at that point. A 3D point is a coordinate in the 3D space defined by (x,y,z). Second, the edges, where an edge contains a start point, end point, right and left faces and an arc in case the edge is not in a straight line. The third component is the faces which defines the shapes and boundaries of the modeled object.

Generalized-Cylinder Models

Every generalized cylinder model is composed of three parts, the first part is the generalized cylinders, these cylinders can be described by their cross section width, their axis length, the ratio of the cross section with and axis length, and the cone angle. The second part is the relationship between these cylinders; one of the most important relations is the connectivity which describes how these cylinders are connected in the 3D space. The last part is the global properties of the modeled object [49]. These properties may include the number of cylinders and summary of the connections between cylinders. Figure 2.2 shows how a person can be modeled using this method. Different cylinders used to model the head, arms, torso and legs. To model more details of the person the high level parts must be divided and represented by different cylinders. For instance, the torso could be divided into two parts the neck and the lower torso. In addition, hands could be divided into the main pieces and five fingers.

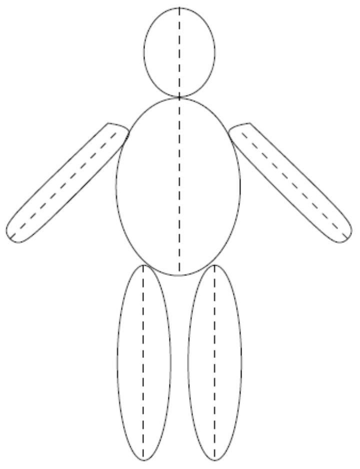

Figure 2.2.: Generalized cylinder model of a person

Octree

Octree is a hierarchical tree data structure. Each node in the tree belongs to a cubic region in the 3D space and has eight children. Every node can be in one of three different states, the first state is called full state where the corresponding cube is completely enclosed with 3D objects. The second is the empty state, in which the corresponding cube does not contain any 3D objects. The last state is the partial state where the corresponding cube contains one to seven 3D objects [49]. Full and empty nodes do not have any children while partial nodes contain eight children.

To represent three-dimensional objects using octrees data structure, a $2^n \times 2^n \times 2^n$ three-dimensional array must be constructed. Where n is the number of the parts that compose the 3D object. Elements of the array have binary values of **1** to indicate that the node is full and **0** to indicate that it is empty. Tree elements are called ***voxels*** and their values indicate the existence and absence of 3D objects. Octree encoding and the three-dimensional array representation are equivalent. However, Octree encoding requires less space than the dimensional array representation.

2.1.4. File Formats

In today's market there are various file formats for storing 3D models. Some of these file formats are closed proprietary formats such as DWG and .3ds. Others are open proprietary formats such as VRML and X3D. Every CAD application has its own file format and most of CAD applications use closed proprietary formats where the 3D representation method used is unexplained or in the best cases is vague. For example, CATIA V5 platform use the following file format (*.CATPart, *.CATProduct). Autodesk 3ds Max use 3DS file format and AutoCAD use DWG file format. Both open and closed file formats rely on the 3D representations described in the previous section. For instance, file formats such as .3ds, .wrl, .mesh, and .x3d are depending internally on 3D Mesh Models representation.

Although every CAD application has its own file format, there are some standard and open file formats that are used for representing 3D models. These file formats are also used to exchange 3D models between different CAD application. For instance, X3D is a standard XML-based file format used to represent 3D models built in different applications in a declarative way.

2.2. 3D Internet Status

Since the early days of 2D HTML pages, users were always demanding the existence of the 3D content on the Internet [7]; and a reason to those demands is evolution [3]. User interfaces evolved through the last decades, it started with Command line Interface applications (CLI), then it evolved into desktop Graphical User Interface Application (GUI), after that Web applications appeared [3]. The next logical step was to integrate the 3D content in web applications [3]. Another reason for demanding this technology is that 3D models are more suitable for providing collaborative virtual environment for services, interaction and communication between various stakeholders [3].

The current version of the Internet provides the users with a comfortable and familiar means, which is to communicate with each other, get the latest news, shop, pay their bills and a lot more. The Internet as we know is organized in a set of flat interconnected hierarchical documents [3]. The content of 2D websites consist of a collection of documents and media, such as, photos or videos. In order for the user to not get lost while using the web pages, the web pages provide the users with navigation [3]. As a result, users rely on the back button on the browsers for navigating between web pages and on search engines to find content on the Internet [3].

On the other hand, 3D Internet will help users to find what they are looking for using a navigation method only without having to use search engines. For instance, instead of searching for restaurants in a specific city, using 3D Internet you can use 3D map to navigate to the given city, and then try to locate the restaurants in that city. Another example is a library web application where the books and documents are 3D virtual entities, and the users can organize their document as they want. This way, it is easier for the users to remember the location of their documents and they do not need a search functionality to locate their data [3].

The ability to view and interact with 3D content on a browser is thrilling and therefore, various standards and implementations had been developed during the last 20 years to bring the 3D content to the Internet. Some of these implementations improved the 3D rendering model dramatically such as online game engines (e.g. Second Live). However, most of these implementations disappeared over time mostly for performance and efficiency reasons. These days, we still have a number of systems for viewing 3D content on the internet; these systems can be categorized into two groups. The first group is for systems that render the 3D content using browser plugins. The second group is for systems that render the 3D content without plugins by either faking 3D rendering process or integrating the rendering process with the browser architecture directly. This section will provide a brief overview of some of the currently used approaches.

2.2.1. Rendering with Plugins

A plugin is a software component that extends an existing software by adding new features to the application [27]. A plugin is a client side component and it generally has a full access to the client resources, therefore, the implementation of plugins is more flexible than server side components. Web browsers such as Chrome and Firefox support plugins to enable the customization of the browser and to allow users to extend the functionality of the browser. For instance, plugins used to add search-engines, virus scanner tools to the browses. This section provides an overview of a number of common plugin-based systems that is used for 3D rendering.

Flash and PaperVision

Adobe Flash is a multimedia platform introduced by Adobe systems in 1996. Adobe systems are still responsible for the development and distribution of Adobe Flash till today. The main aim of Flash is to add interactivity and animations to the web pages. It is commonly used to create advertisement, animation movies, and even content rich web applications.

The first nine releases of Adobe Flash did not support real-time 3D. Flash in these releases only have the ability to display 2D vector shapes on the screen. However, developers built 3D rendering systems for Adobe Flash such as PaperVision3D project (PV3D)[14] which is an open source 3D engine for Adobe Flash platform. PV3D was developed by Carlos Ulloa in November 2005, and maintained by a small core team.The functionality of PV3D is to enable developers to create and control 3D models in Adobe Flash platform using Actionscript. PV3D and other similar systems are based on exploiting 2D vector shapes and some authoring tools that support 3D graphics natively such as Director, Cult3D, and Anark.

Adobe Flash version 10 was the first release including and supporting 3D objects and 3D transformation. However, the 3D support is very limited [7]. It is designed to support only simple transformation, 3D composite and GUI effects. On the other hand, developers started updating their projects to take the advantage of the new features of Adobe Flash. For instance, the PaperVision3D, people started updating their system to add new features for displaying and viewing 3D models. The new version will be called PaperVisionX,

[14]https://sites.google.com/site/multimediatechnologyvu/technology-wiki-s/papervision3d Retrieved on 2014-10-14

and they have remarkable results on creating and controlling 3D models using 2D render pipeline.

Silverlight

Microsoft Silverlight is a development tool for creating vector graphics, audio-video playback and animation for rich web and mobile applications. Silverlight is a free plug-in developed by Microsoft as an alternative for Adobe Flash, it is based on the .NET framework and it is compatible with multiple browsers operating systems such as Mac OS. Microsoft Silverlight provides the developers with the functionality to create and engage into an interactive user experience for both web and mobile applications. The installation and user base of Microsoft Silverlight is much smaller than those of Adobe Flash.

Because Microsoft follows the same path of the development of Adobe Flash, Silverlight and Adobe Flash platform almost share the same situation today. At the beginning Microsoft released Silverlight with no native support for rendering 3D objects. Developers used to build solutions to fake the rendering of 3D content using a similar technology used in Flash. Perspective transforms is a new feature added to Silverlight version 3. This features allows developers to transform 2D objects into 3D coordinate system, however, it does not generate real 3D shapes.

Java, Java3D, JOGL and JavaFXn

For several years, Sun worked to push java as client-side technology through providing browser plugins to integrate java applets into the browsers. However, today Java is mainly used as a server-side programming language. One reason is that Microsoft never officially supported Java on Windows and that affected the number of users of user and install base of Java [7].

Java3D is an application programming interface (API) developed by Sun to incorporate the VRML/X3D designs. Java3D is a client side Java API for rendering interactive 3D graphics. It was commonly used for desktop applications but it was never utilized for the web development [7]. Moreover, Sun stopped supporting Java3D and dropped the high-level Java3D library [7]. Instead, it provided new lower-level interface called JOGL which provides direct bindings to the OpenGL interface.

Sun's last attempt to build an alternative to Flash was JavaFX, which was announced in 2008. JavaFX 8 offers 2D media and simple transformation of the 2D GUI graphics into 3D graphics via the rotation of the z-axis [19]. However, JavaFX does not have any native 3D support.

X3D Plugins

X3D is a royalty-free ISO standard XML-based file format and scene-graph architecture. The 3D geometry is represented by different node types with the X3D document. For instance, shape node is used to define 3D geometry, which includes a mandatory geometry node such as box or sphere and an optional appearance node. Figure 2.3 shows a simple X3D document content. Every X3D document includes only one *Scene* node. This node contains the definition of all the 3D geometries in the 3D model. The presented X3D document includes only one geometry which is a sphere shape.

```
<?xml version="1.0" encoding="UTF-8"?>
<X3D profile='Immersive' version='3.2'
     xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
     xsd:noNamespaceSchemaLocation=
     'http://www.web3d.org/specifications/x3d-3.2.xsd'>
  <Scene>
        <Shape>
          <Sphere/>
          <Appearance>
            <Material diffuseColor='0 0.5 1'/>
          </Appearance>
        </Shape>
  </Scene>
</X3D>
```

Figure 2.3.: Simple X3D file contents

X3D is a successor to the Virtual Reality Modeling Language (VRML97) and it includes a large number of extended new components and features (e.g. CAD[15], Geospatial analysis[16], NURBS[17] etc.). The main purpose of X3D is representing 3D computer graphics. X3D uses XML to represent and encode the 3D geometry and its behavioral capabilities. Furthermore, it allows program scripting and node prototyping, which provide a support of scene-graph extensions. X3D is not only a simple exchange file format, its specifications include a full runtime event, behavior model and various external and internal APIs.

An X3D plugin (also known as X3D player or viewer) is a browser add-on that is capable of parsing X3D files and then rendering these files on to a web browser window. X3D plugins enable displaying 3D models from various viewpoints or angles, real time interaction with the 3D object and object animation.

Figure 2.4 is a simple prototype of X3D plugin architecture. The X3D plugin takes X3D scenes or streams as inputs, which are basically files that include X3D data that are read by the browser. Parsers are the components that are responsible for parsing the X3D data from various file-format encoding and generating X3D nodes. The scene graph manager keeps monitoring the defined 3D geometry and its properties such as appearance, location and orientation. The event graphs' main purpose is to monitor all animation nodes which are used to generate and pass value-change events into scene graph. The received events

[15]computer software that assist designers in the development of a design.
[16]An approach to applying statistical analysis to geographical data
[17]a mathematical model used in graphics applications for generating and representing curves and surfaces

have the ability to change any property of the rendered images. To extend the animation nodes the scripting engines component was added to the architectures. This component adds the following functionalities for the plugin, sending and receiving events, generating 3D geometry in the scene and removing 3D geometry from the scene. The Scene Authoring Interface (SAI) main purpose is to provide the ability to control and update the content of 3D geometry.

The actual implementation of each X3D plugin is different from other implementations. However, all these plugins share the same basic concept for paring the X3D data and rendering it on the browser window. In addition, all of these plugins have the same goal which is providing a convenient user experience for viewing X3D scenes [14].

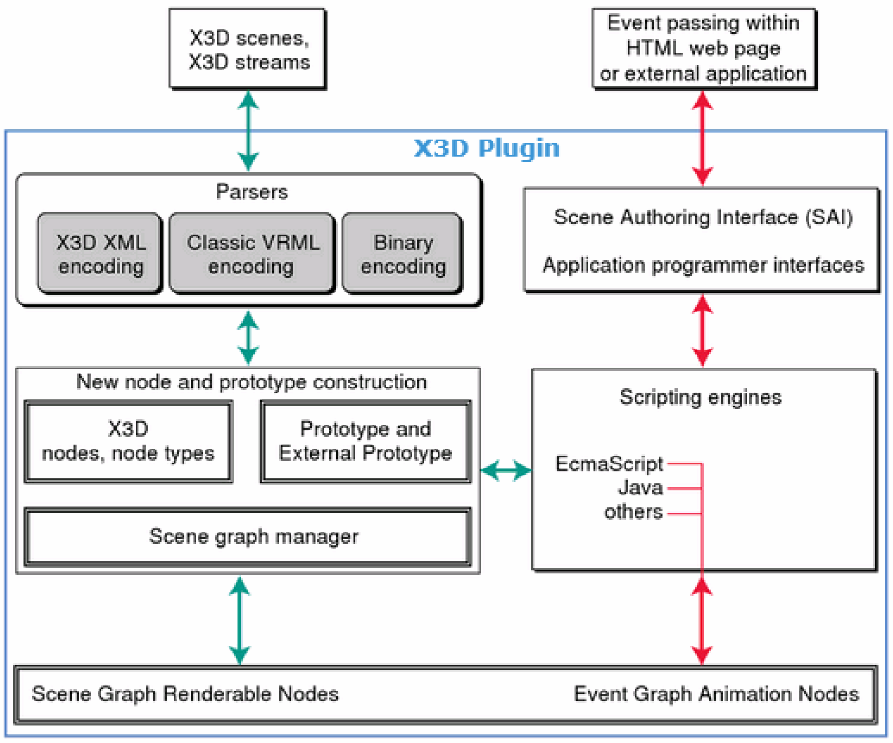
X3D scenes, X3D streams
Event passing within HTML web page or external application
X3D Plugin
Parsers
X3D XML encoding
Classic VRML encoding
Binary encoding
Scene Authoring Interface (SAI)
Application programmer interfaces
New node and prototype construction
X3D nodes, node types
Prototype and External Prototype
Scene graph manager
Scripting engines
EcmaScript
Java
others
Scene Graph Renderable Nodes
Event Graph Animation Nodes

Figure 2.4.: Prototype of X3D plugin architecture [14]

All presented plugin-based systems share two major disadvantages. The first one is that they are based on plugins, which are not installed by default on client machines. As a result, it is the users responsibility to install these plugins, update them, and deal with their security issues. The second drawback is that these systems define an application and event model inside of the plugin. This means that the developer of the web-application that uses the plugin and the DOM/browser has to care about plugin-specific interface and its synchronization capabilities. Both of these drawbacks explain why the plugin-based system is not successful over the last years and not widely used.

2.2.2. Rendering without Plugins

This section will provide an overview of the current technologies, which are used to display 3D models on the browser directly without the need for a plugin. Some of these proposals are depending on internal 2D browser techniques to fake a 3D model and some of them are relying on trying to build an abstraction for the 3D hardware layer which can be used directly by the developer.

CSS/SVG Rendering Solutions

Currently neither Cascading Style Sheets (CSS) nor Scalable Vector Graphics (SVG) provide support for 3D graphics and their main goal is to support 2D graphics. However, with the improvement of the browsers and JavaScript performance, over that last years developers started working on building new solutions utilizing 2D objects to build 3D pipelines based on dynamic HTML techniques. A good example of such systems is SVG-VML-3D which is an open-source JavaScript library used for creating and manipulating 3D objects in HTML pages using SVG.

In addition, in **2008** Apple added some 3D CSS transformation to their WebKit , which was later added to other browsers such as Firefox and Opera [7]. These transformations enable the web developer to rotate, translate, scale, skew and change the viewpoint of any DOM object in the 3D space. However, these transformations apply only to the 2D objects only [7].

The HTML5

HTML5 specifications support the integration of X3D declarative 3D models into webpages. The proposed way for the integration is through embedding the 3D geometry into an XHTML page. However, the specification does not define how to access the scene-graph content or how the integration between the X3 D scene-graph and the DOM should look like.

O3D

O3D is an open-source graphics API developed by Google for creating rich, interactive 3D graphics and applications that run in the browser window. Earlier versions O3D were released as a component that consisted of two layers (a browser plugin and a JavaScript library). The lower layer which is the browser plugin is developed using C/C++ and its goal was to provide an abstraction that is mapped to the OpenGL. The high-level layer is a JavaScript library that provides developer with a scene graph API to manipulate the 3D geometry.

On May 7th, 2010, Google announced that the new release of O3D will no longer contain the plugin layer [12]. Instead, the new release will be a JavaScript library running on top of WebGL. One advantage to the new release of O3D is that it enables the dynamic loading, rendering and transforming of 3D models and their relevant textures using Ajax and/or COMET[18] in real-time. On the other hand, O3D does not provide a technique for defining 3D model content in a declarative way. JavaScript developers who want to use O3D have to build, update and manipulate the scene-graph using JavaScript Code. However, there are some transcoding tools that enable the transcoding of declarative 3D data into JavaScript code.

Hardware Accelerated Rendering

Hardware accelerated rendering proposal advocates the use of hardware components to run rendering functions faster than it is possible in software solutions. During the last 15 years engineers developed and presented many algorithms and approaches that rely on this proposal [23, 45]. The reason behind the big interest of this proposal is that it is almost impossible to purchase a new personal computer without the hardware support of stencil testing hardware [23].

Furthermore, browser companies worked on similar goals. For instance, the Canvas3D project[19] aims to provide support for low-level hardware accelerated with 3D rendering functionality through HTML canvas element. The Canvas3D is developed by Mozilla and delivered as a Firefox addon [41]. Canvas3D and other projects such as 3D-Context [7] from Opera work as a wrapper for OpenGL[20]. The main functionality that these projects provide is that it enables developers to call OpenGL commands directly [7]. Since OpenGL is a stable API for rendering 2D and 3D graphics and it is used since 1991 in game development such as DOM III and second live, and since it is available on all major platforms, projects such as Canvas3D and 3D-contect works quiet well [7].

On March 29th, 2009, The Kronos Group [21] launched an initiative to develop a standard for such type of integration. The Kronos Group started the WebGL working group with the participation of Apple, Google, Mozilla, Opera, and others [58]. WebGL is a web technology that employs hardware-accelerated functionality for rendering 3D graphics into the browser without installing additional software. It evolved out of the Canvas3D and it becomes a JavaScript API. WebGL can be used for rendering interactive 3D and 2D graphics in compatible web browsers such as Chrome, Opera, and Firefox without the usage of any software of plug-in.

Building and lading 3D models into a web browser works pretty well using WebGL. Furthermore, it enables programmers to use arbitrary visualization methods. However, it is difficult to visualize complex and large 3D models. Performance is another issue for WebGL since it is implemented in JavaScript.

[18]a web application model which enable web servers to push data to the client without requesting it using a long-held HTTP request.

[19]https://wiki.mozilla.org/Canvas:3D Retrieved on 2014-10-14

[20]Open Graphics Library is a cross-language, multi-platform application programming interface (API) for rendering 2D and 3D vector graphics.

[21]Not-profit company focused on the creation of open standard and royalty-free application programming interfaces (APIs)

X3DOM

X3DOM is an open-source JavaScript framework. It is also a runtime for rendering, manipulating and loading 3D models on a web browser [60]. X3DOM main goal is to integrate declarative 3D content with HTML5 by including X3D elements as parts of HTML5 DOM tree [7]. X3DOM enable users to add, remove or change 3D models in a web browser by manipulating their associate DOM objects without using any plug-in or software.

In general X3DOM can been seen as an abstract scene-graph layer that is mapped directly to the DOM elements [7]. In contrast to application programming interfaces such as OpenGL, WebGL, and O3D, X3DOM allows the software developers to define their 3D models in a declarative way using a X3D file format. The main technologies that are involved in the development of X3DOM are X3D file format and HTML5. X3DOM simply works as a connector between HTML5 and X3D.

X3DOM is different from all other solutions presented. First of all, X3DOM is not a plug-in and therefore it does not requires any effort from the end user for configuration or setup. Second X3DOM is not an application programming interface only, X3DOM provides an API along with a runtime environment for rendering 3D models. Furthermore, it provides a connection between the static representation of the 3D model and their corresponding DOM elements to enhance the interaction between the end user and the 3D model. This feature is not provided by any of the presented technologies. For the mentioned reasons we decided to use X3DOM for rendering purposes in this thesis. Section 2.3.5 describes in details the architecture, runtime system, and the supported features of X3OM framework.

2.2.3. Summary

This section presented a number of technologies, platforms, and plug-ins used to transport 3D models to the web browser. It started by describing various plugin-based solutions such as Flash and Silverlight. Then it continues to illustrate technologies used to render 3D models without the need of plugins.

Plugins are programs that are used to extend the functionality of another application. This section highlighted several plugin-based solutions such as Flash, Silverlight, Java, Java3D, JOGL and JavaFX, and X3D Plugins. Flash and Silverlight plugins support 3D rendering in a limited way, therefore software developers who use these technologies still face many difficulties in developing and controlling visual effects. Java technologies are rarely used for building 3D models and loading them into a web browser.

X3D plugins are powerful plugins for rendering 3D models. However, they are pieces of software that need to be installed and administrated by the end user and they are not included in browsers by default. In addition, these plugins have their own runtime environment for visualization, interaction, and communication with the 3D models. That means that the application developers need to learn the specification of each plugin in order to be able to use it. This drawback not only applies for X3D plugins, but also applies for all plugin-based solution presented in this section.

On the other hand, rendering solution such as CSS/SVG exists in the market, however, it only supports the rendering of 2D graphics with a very limited support to 3D graphics. HTML5 offers a specification that supports the integration of X3D declarative 3D models into webpages, but this specification does not specify how the DOM integration should look like or how the user can access the scene-graph nodes.

O3D and Hardware Accelerated Rendering such as WebGL provide application developers with JavaScript applications programming interfaces. These APIs can be used for building and controlling 3D models. They are efficient in the rendering process and their performance being quiet well, however, developers need to implement every transformation of the 3D model. Moreover, the rendering performance for large 3D models is still insufficient.

X3DOM was the last proposal illustrated by this section. X3DOM is a JavaScript platform and runtime that is based on both X3D and HTML5. X3DOM is different from all presented methods since it does not introduce any new technologies but instead it integrates existing technologies to achieve its goal. The main aim of X3DOM is to connect the static representation of the 3D model to the DOM elements and provide a set of APIs to control and manipulate the 3D model by modifying the DOM elements.

2.3. Technologies used in the Thesis

The aim of this section is to present and explain to the reader all the technologies that have been used during the thesis. Moreover, the section will provide the rationale and reasons behind selecting these specific technologies, and it will highlight the advantages and drawbacks of these technologies.

The section begins with a brief overview of distributed environments. After that, it will present the distributed methodology or the programming model used in the thesis and its capabilities for distributed processing. Next, a description of the distributed framework architecture used to distribute the processing of 3D data is presented. Then, it moves to explain the structure of the X3D file format and how 3D models are represented within this format. After that, the core concepts of XQuery are explained briefly. Next, the architecture of the 3D rendering framework that is used for displaying and controlling 3D models in the browser will be discussed. Furthermore, the section will continue to highlight the transcoding tool used to convert X3D shapes to binary format. Finally, this section gives the reader an overview of the Amazon elastic computer cloud **EC2** web service and how it is used in the thesis.

2.3.1. Distributed Environments

One goal of this thesis is to define a data-parallel approach for transcoding geometry data to binary format in an efficient and productive way. The parallel transcoding of the geometry data will help us in reducing the processing time of a single 3D model.

To develop an environment that is capable of performing data-parallel operations we had two options. The first option is to use **parallel programming model**. This model is used by developers to develop applications that are able to run on and take the advantage of multi-core computers. The main aim of this model is to increase the performance of computer programs by exploiting the usage of multiple processors on the computer [35].

Throughout the last years, various parallel programming specifications and implementations have been developed. For instance, Pthreads, OpenMP[22], and Fortress are parallel programming libraries used for developing such an application [35]. Choosing one implementation over another one will affect the productivity of software development process and the performance of the developing software.

Parallel programming model has several drawbacks that affect the goals of this thesis. First of all, to be able to develop a parallel programming application the developers need to learn new programming paradigms and APIs that are specific for parallel programming. Moreover, developing an application with this model involves dividing the tasks and work load of the application and mapping these tasks or units of work to the processors on the computer. The learning and development processes are very slow and it would consume too much time to be done within this thesis. Second applications that are developed by this model can only run in a single computer and the parallelism power is restricted by the number of processors on that computer.

The other option for performing data-parallel operations is **distributed computing**. The key idea of distributed computing is to distribute the processing of data or tasks over multiple computers which are located in different physical locations. There are many advantages for distributed computing over parallel programming. One advantage is that distributed computing enables developers to build very powerful clusters of computers and this is more scalable than parallel programming model. A cluster is a collection of computers that are connected to each other through a network and responses as a single computer to the outside world.

In contrast to parallel programming where the application is restricted to the computer resources where they are running; in distributed computing, applications can take advantage of the resources of all computers on the cluster. Furthermore, it is easier and cheaper to enhance these clusters by just adding new computers to the cluster, while in parallel programming you need to replace the computer by another one with different specifications.

For the reasons named above, mostly to ease the building of powerful distributed clusters and the scalability of distributed computing model, we decided to go for the distributed computing since it will enable us to have a more powerful processing system and therefore the improvement of transcoding large 3D models will be greater than the improvement using parallel programming in a single computer.

[22]http://openmp.org/ Retrieved on 2014-10-14

After the investigation of tools and technologies that is used in building and developing a distributed system for transcoding 3D data we found that the **MapRreduce programming model** is one of the best options that is suitable for our objectives. One advantage of this programming model is that it is easy to learn and to develop distributed application with it, but the most important benefit is the powerful data processing that this model provides for the end user. The following section will present the core concept of MapReduce programming model and how it is used in processing large datasets.

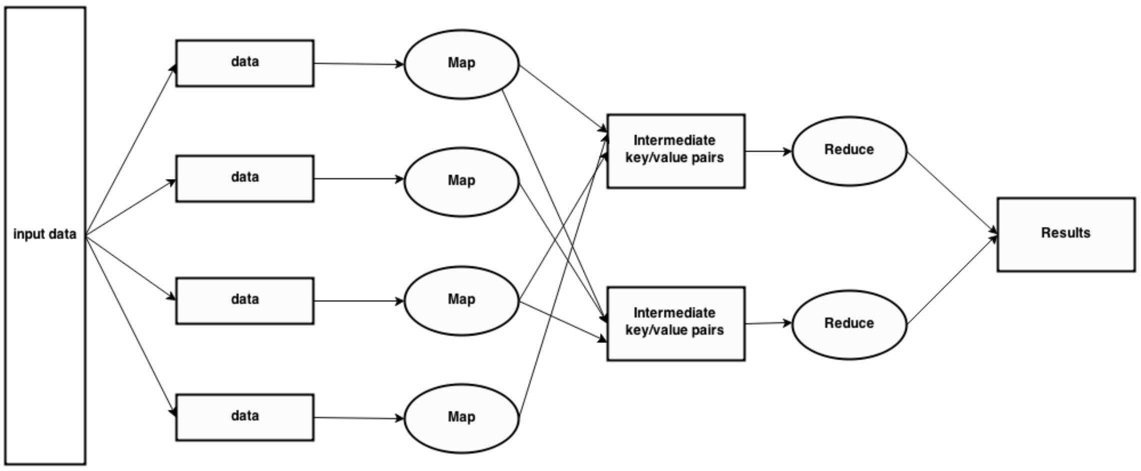

Figure 2.5.: MapReduce programming model

2.3.2. MapReduce

MapReduce is a parallel programming model that aims to enable developers to build applications with the ability to process massive amounts of data on large distributed cluster of commodity computers [20]. MapReduce programming model can basically be seen as a simple API consisting of two methods, which are the **Map** function and **Reduce** function [61]. To develop a MapReduce job, developers need to implement the **Map** and **Reduce** functions without specifying anything related to the distribution of the data or tasks on cluster nodes. MapReduce model is used by big companies like Google for many purposes such as data-generation, sorting data, data mining and machine learning [20].

According to [36] MapReduce model has many advantages and drawbacks. To begin with the drawbacks, MapReduce does not support any high-level languages such as SQL for databases. Developers must write their own implementation of **Map** and **Reduce** functions. MapReduce model is designed in a way that it provides the developers with a fix data-flow. As a result, MapReduce cannot be used for many complex algorithms where multiple inputs are required. The last drawback is that MapReduce focus more on ensuring fault-tolerance and scalability and therefore its operation is not optimized for I/O efficiency.

On the other hand, MapRduce has advantages as mentioned in [36]. It is very simple to learn by developers and even beginners since the development of a MapReduce job involves only the implantation of **Map** and **Reduce** functions without specifying how the job and data should be distributed across cluster nodes. It is also very easy to use since it hides the complexity of parallelization; fault-tolerance, locality optimization and load balance from the developers.

MapReduce is also very flexible, since it does not have any kind of dependency on data models and it can work with structured or unstructured data. MapReduce ensures good distributed systems quality with its support for non-functional requirements such as fault-tolerance and high scalability. In 2008, Yahoo! declared that their own Hadoop gear could scale out more than 4,000 nodes [5]. In addition, Google stated that MapReduce jobs are completed successfully even in the presences of both **Map** functions and **nodes** failures with an average of 1.2 per job [20]. In other words, MapReduce jobs are robust and fault tolerant jobs.

Programming Model

MapReduce is more than a parallel programming model, it is also a framework that supports the proposed parallel programming model [36]. The core concept of MapReduce model and framework is to hide the complexity of parallel execution strategies in order to allow the developers to focus on the data-processing strategies [36].

MapReduce framework provides developers with a simple API that consist of two functions: **Map** and **Reduce**. All MapReduce Jobs take a list of key-value pairs (*key1, value1*) as their initial input. The **Map** function which is written by the developer is applied to each key/pair in the input list. The aim of the **Map** function is to generate intermediate key-value pairs (*key2, value2*). After applying the **Map** function to all pairs in the input list, all intermediate key-value pairs that have the same key will be composed into a separate group. The result of this step will be in the form (*key2, listOfAll(value2)*) The **Reduce** function is then applied to all *key2* pairs in order to merge all the intermediate pairs associated with the same key and then generate outputs.

In order to have a clear understanding of the MapReduce programming model, imagine that we want to develop a word count application using MapReduce model. The main goal of the application is to count the number of occurrences for each word in a large set of documents. As mentioned before the developer should implement the **Map** and **Reduce** functions first. Figure 2.6 shows a pseudo code for the **Map** and **Reduce** function [21]. The **Map** function inputs are the file name and its contents. The **Map** function loops through all the existing words in the content of the file and generates an intermediate key-value pair for each word with the **word** itself as the key and **1** as the value where the value represents the number of occurrences of the word. After grouping the intermediate key-value pairs, the **Reduce** function will be applied to merge the occurrences of each word. The proposed **Reduce** function basically iterate over all the intermediate key-value pairs with the same

```
map(String key, String value):
//key: file name
//value: string representing file contents
 for each Word w in value
 emit(w,"1");


reduce(String key, Iterator values):
 //key: word
 //values: list of word counts
 int sum=0;
 for each c in values
 sum += getInteger(c);
 emit(key,sum);
```

Figure 2.6.: Proposed **Map** and **Reduce** functions to solve word count problem

word as key and sum their values. Finally, the **Reduce** function generates the output to the end user or pass it to another algorithm or program.

Figure 2.7 illustrates the execution of the proposed pseudo code on a MapReduce environment. The input for the job is a large set of documents. As shown in the figure each instance of the **Map** function receives a content and file name as key-value pairs. The **Map** function generates the intermediate key-values in our case (*"Waleed",1*) key-value pair is generated three times which indicates that the word *Waleed* has three occurrences. One instance of **Reduce** function will receive a key-value pair that look like (*"Waleed",list(1,1,1)*). This instance will iterate over the list values, calculate the sum and at the end generate the output which is (*"Waleed",3*).

MapReduce Design Patterns

MapReduce Programming model can be used to solve various types of problems and it is used in many computer science fields such as machine learning and distributed systems. This sub section will highlight some of uses of the MapReduce model and it will give some examples for MapReduce usage. According to [38, 40, 11] MapReduce usage can be categorized into the following three categories

1. Basic MapReduce Patterns
2. Complex MapReduce Patterns
3. Relational MapReduce Patterns

Basic MapReduce Patterns category includes solutions for common problems that many developers face in order to prepare or perform a simple data-analysis operation on large

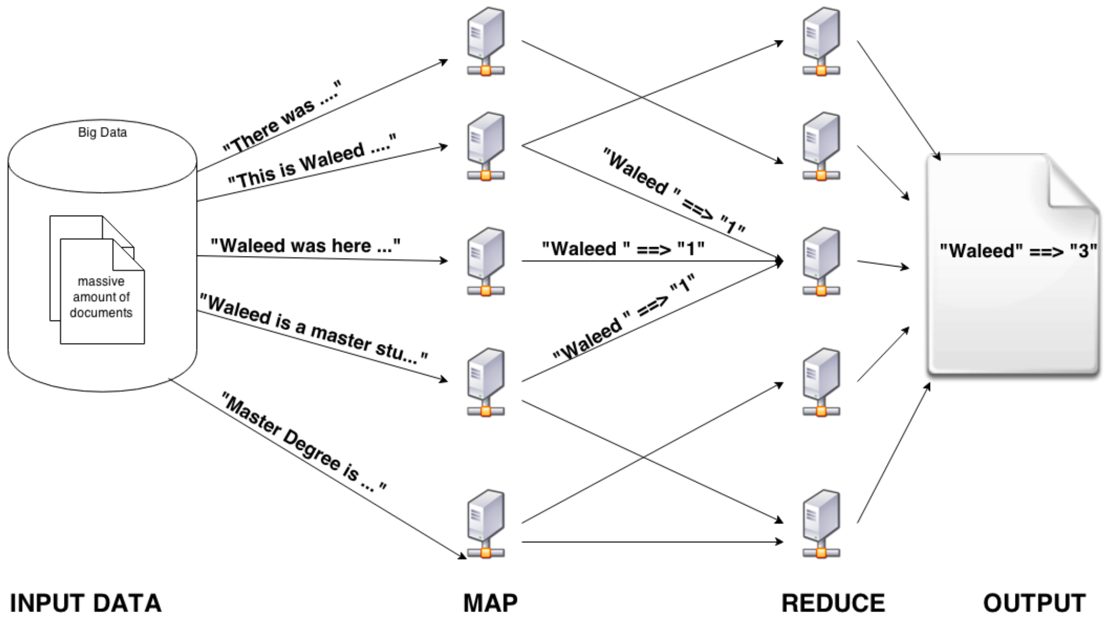

Figure 2.7.: Execution of the proposed pseudo code on a MapReduce environment

datasets. These basic problems could be a counting and summing task where the goal is to calculate the total number of occurrences for each word in a large set of documents or calculating the average response time from a large set of log files [38]. Also it could be collating problem [38] where the goal here is to group items based on a given function, and then storing these items with identical results in separate files. An example of collating is building of inverted indexes[23].

MapReduce can also be used for solving filtering, parsing, and validation problems [38], where the main goal is to select a subset of records from a large set of records or text files by applying a selective criterion on these records. Another basic MapReduce pattern is distributed task execution, where the aim in this type of problems is to distribute the computational processing of job large computation problems. Finally, sorting problems can be solved with MapReduce and considered to be basic MapReduce problems. The goal of this kind of problems is to sort a large set of records based on some values and present them in a specific order.

Complex MapReduce Patterns includes three sub categories. First of all, *Graph processing* problems [38] . This type of problem deals with the processing graph data in order to generate status of graph nodes based on their properties and neighbors. For instance [38] proposed MapReduce algorithms for parallel breadth-first search and PageRank. Second set of problems are *selecting distinct values* from a set of records, this type of problem is considered a complex one for two reasons, first the input dataset is processed by multiple nodes on the cluster and second the input dataset could be a complex dataset that consists

[23]An inverted index provides quick access to documents ids that contain a specific term.

of multiple fields. The third complex set of problems is the *Cross-Correlation* problems, where the goal is to find a relation between different items in the dataset. The dataset for such problems usually consists of a set of items and a set of tuples formed from these items. An example of such problems is a text analysis problem where the words are the items and sentences are the tuples.

Relational MapReduce Patterns as presented in [1] and [15] MapReduce programming model can be used to perform and calculate relational operations. These documents also proposed various solutions for using MapReduce to implement MapReduce jobs that are capable of performing selection, projection, union, interception, group by and aggregation, difference, and even joining tasks on a relational dataset. For instance, for implementing the difference operation, consider that we have a dataset that consists of two sets of records **A** and **L** and we want to compute the difference between these two sets **A**-**L**. To implement such an operation using MapReduce, developers should write the **Map** function in a way that it emits for every tuple both of the tuple as a key and the value should be the name of the set that the tuple belong to. The **Reduce** function should be written in a way that it only emit tuples that belong to set **A**. Figure 2.8 present a pseudo code for such an implementation.

```
Map(rowkey key, tuple t):
//key: tuple primary id
//tuple: object that represent the tuple values
// t.getSetName is either 'A' or 'L'
Emit(tuple t, t.getSetName())


Reduce(tuple t, array s) :
//tuple: object that represent the tuple values
// array s can be ['A'], ['L'], ['A' 'L'], or ['L', 'A']
 if s.size() = 1 and s[1] = 'L'
 Emit(t, null)
```

Figure 2.8.: Proposed **Map** and **Reduce** functions to solve rational difference operation

As mentioned above the MapReduce model is used in fields such as machine learning and math. For instance, in [16] MapReduce model is used to demonstrate the parallel speed up on various machine learning algorithms including k-means, locally weighted linear regression, logistic regression, naive Bayes, gaussian discriminant analysis, and many other algorithms. Another example of the usage of MapReduce is presented in [52] where it is used for integer factorization and in [46] MapReduce is used for providing a high-level matrix computation primitives.

MapReduce originally developed by Google [52]. However, these days there are multiple implementations of MapReduce programming model. For instance, MongoDB[24], Riak[25], Infinispan[26], Apache Hadoop[27], and Apache Couchdb[28] are different implementations for MapReduce model. Most of these implementations are open-source implementations and each of them are used for specific domain. For instance, Apache CouchDB is a database that is using JavaScript for MapReduce indexes while Apache Hadoop is an open-source platform for distributed processing of large data sets across clusters of computers using different programming models. After reviewing the different open-source implementations and their advantages we decided to use Apache Hadoop as the MapReduce environment in this thesis. This decision was based on the needs of the thesis and the capabilities of the Apache Hadoop. One of the reasons behind this decision is that Apache Hadoop is a reliable implementation and can scale up for thousands of servers [47], it is used by many companies around the world and it has a big community that supports it [31], and it allows developers to use Java or other programming languages such as Python and Perl for developing MapReduce jobs.

2.3.3. Hadoop Framework and HDFS

This part of the section will begin with an overview of Hadoop project and its origins, and then it will present the main components of the Hadoop project and their purposes. Next, it will describe the Hadoop file system component (**HDFS**) and finally it will explain the basic architecture of the HDFS.

Hadoop Origins

Hadoop is a Java-based and open source implementation of MapReduce programming model maintained currently by the Apache Software Foundation [15]. Hadoop originally developed by *Doug Cutting* based on another Apache project called Apache Nutch[29] which is an open source web search engine. At the early stages of Nutch scalability was a major problem for the system, however after the development of Google File system (GFS) in 2003 , Nutch project was influenced by the GFS idea and therefore Nutch project develop a similar file system called Nutch Distributed File System (NDFS) [15].

When the concept of MapReduce was introduced to the world in 2004 [20] , Nutch devlopers adopted this concept and implemented it for their own purposes. In February 2006, Hadoop was born because Nutch developers decided to move NDFS under a separate sub project [20]. After two years, in January 2008 Hadoop became a top level project and the NDFS was renamed to HDFS Hadoop Distributed File System.

[24]http://www.mongodb.org/ Retrieved on 2014-10-14
[25]http://docs.basho.com/riak/latest/theory/why-riak/ Retrieved on 2014-10-14
[26]http://infinispan.org/ Retrieved on 2014-10-14
[27]http://hadoop.apache.org/ Retrieved on 2014-10-14
[28]http://couchdb.apache.org/ Retrieved on 2014-10-14
[29]http://nutch.apache.org/ Retrieved on 2014-10-14

Module Name	Description
Hadoop Commo	The common utilities that support the other Hadoop modules
Hadoop Distributed File System HDFS	A distributed file system that provides high-throughput access to application data
Hadoop YARN	A framework for job scheduling and cluster resource management
Hadoop MapReduc	A YARN-based system for parallel processing of large data sets

Table 2.2.: Hadoop main modules [26]

Project Name	Description
HBase	Distributed column-oriented database
ZooKeeper	Coordination service for distributed applications
Pig	Dataflow language and framework for parallel computation
Hive	Data warehouse infrastructure
Avro	Data serialization system
Chukwa	Data collection system
Spark	Compute engine for Hadoop data
Mahout	Machine learning and data mining library
Cassandra	Multi-master database

Table 2.3.: Hadoop related projects [26]

Hadoop is the best large scale data processing framework available in the market these days [43]. Hadoop provides developers with tools and sub systems for handling distributed data storage management, analysis and transformation for vast amounts of datasets using MapReduce programming model in a reliable and scalable way. For instance, Yahoo! Hadoop clusters can span to 25,000 servers and handle 25 petabytes of data on their HDFS [47].

Hadoop Components

Currently Hadoop is an open source apache project that consists of multiple components and models. All Hadoop components are available via the apache open source license. Table 2.2 lists all Hadoop project modules, while Table 2.3 lists some of the projects that are related to Hadoop. Hadoop components and modules developed by different companies. For instance, HDFS and MapReduce modules developed mainly by Yahoo! with contribution percentages of 80% [47]. Yahoo! also originated and developed the ZooKeeper, Chunkwa and Avro modules. Facebook contributes to Hadoop project by developing Hive and Cassandra. Microsoft is currently developing HBase which was originally developed at Powerset [47].

Hadoop Distributed File System

HDFS is a distributed file system used for storing and managing huge amounts of data files on commodity clusters [55]. The main advantages of HDFS are high fault-tolerant and the possibility to run low-cost hardware [25]. It is developed to handle massive amounts of data even larger than petabytes [25]. HDFS separate between file system metadata and application data and store them separately. The application data are stored on devoted servers called DataNodes while the metadata are stored on a dedicated server, called NameNode. All DataNodes and NameNode servers are connected to each other and communicated through TCP-based protocol. Instead of data protection mechanisms such as RAID, HDFS support data replication where the file contents are replicated on multiple DataNodes on the cluster. The data replication enhances the performance because there are higher chances that the computations are performed near the needed data, and thus the data access is local data access as much as possible.

HDFS Architecture

HDFS is developed using master/slave architecture [25]. An HDFS cluster consists of several computers called *nodes* with different roles. HDFS is developed using Java programming language thus any computer that supports Java can be a part of an HDFS cluster. Figure 2.9 presents the HDFS architecture, HDFS cluster has only one NameNode which is responsible for storing all the metadata of the files on the cluster , a single master server that controls the file system access by clients and manages file system namespace, and several DataNodes which are used for storing data. Files are partitioned into one or multiple blocks and stored on a set of DataNodes. DataNodes and NameNodes are not the only components defined by the HDFS, HDFS has seven different components for the nodes in the cluster, below is a brief description of these seven components as described in [47].

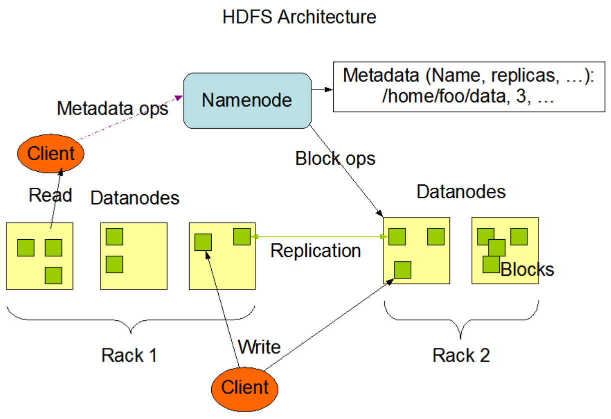

Figure 2.9.: Hadoop Distributed File System architecture [25]

NameNode

NameNode are responsible for controlling and maintaining the HDFS namespace. The namespace is a hierarchal representation of the files and directories that exist on the file system. The NameNode keeps track of the attributes of the files and directories such as modification and access times and represent them as *inodes*. As mentioned above, HDFS split file content into multiple blocks and then replicate each of these blocks independently on the DataNodes. Another responsibility of the NameNode is to map file blocks to their physical location on the DataNodes. An HDFS cluster can only have one NameNode , this node will guide HDFS clients during read and write operations on the file system.

DataNode

An HDFS cluster can have one or thousands of DataNodes. These nodes are used for storing and managing the file blocks on HDFS. Each of these blocks are called *replica* and it is represented by two files. The first file contains the replica content and the second for storing metadata about the replica. DataNodes can join the HFFS through an operation called *handshake*, this operation make sure that the DataNode and NameNode have the same *namespave* ID and *software version*. If this condition does not apply then the DataNode will be terminated.

After a successful *handshake* , a DataNode need to *register* with the NameNode in order to get its storage ID. These IDs are assigned to DataNodes when they register for the first time and never change again. The reason behind this behavior is to use storage ID for identifying DataNodes even if they change their IPs. To remain a node of the HDFS, DataNodes need to keep sending messages to the NameNode. These messages are called *heartbeats* and their main purpose is to confirm that the DataNode is still running and available for requests. Moreover, *heartbeats* also have information about the total storage capacity and storage in use for the sending node.

HDFS Client

HDFS clients are applications that are developed to communicate, manipulate and access file system on HDFS cluster. The main operations that these applications perform are read, write and delete files in addition to create and delete directories. These applications do not know anything about how the data is represented internally on the HDFS. For instance, they do not know that the files are replicated on multiple servers. To perform a file read operation, the client requests a list of DataNodes that host the blocks of the file from the NameNode.

Then the client connects to a DataNode directly to read the wanted block. For file write operations, the client starts by asking the NameNode for a list of DataNodes those are available for hosting replicas of the first block of the file. When the client receives this list it builds a pipeline from these nodes and then starts sending the data. After completing the transfer of the first block the client again asks the NameNode for another list of DataNodes that are available to host the second block and then repeat the above steps for sending the data. Figure 2.10 illustrates the communication and interaction between the client, the NameNode and the DataNodes.

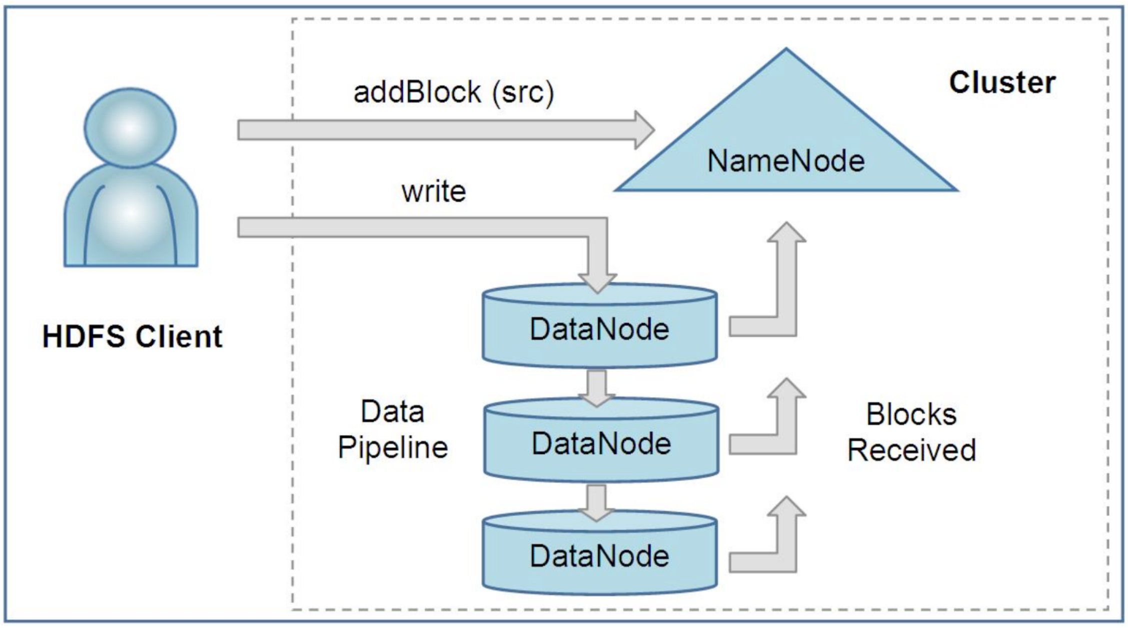

Figure 2.10.: Interaction among an HDFS client, NameNode and DataNodes [47]

Image and Journal

The HDFS namespace *image* is a metadata that describes the *idnodes* data and the organization structure of the files and directories in the file system. The namespace *image* is stored in the RAM. when a copy of the image is persisted to the disk it is called a *checkpoint*. The *journal* is a log file that stores the *image* modification. The *image,journal*, and *checkpoint* main purpose is to allow administrators to restore the HDFS system to an earlier state in case of data corruption.

CheckpointNode

One method for protecting the file system metadata is to provide the functionality of periodic checkpoints. These checkpoints will enable the system to recover from the last one in case the namespace image and journal are unavailable. CheckpointNodes' job is to combine existing Checkpoints and journals to create new ones. CheckpointNode runs on a different host from the NodeName because they share the same memory requirement [47].

BackupNode

In addition to creating periodic checkpoints, BackupNode is capable of maintaining an image of the file system namespace. This image is stored in memory and is kept up-to-date and synchronized with the status of NameNode. Moreover, the BackupNode creates these checkpoints without the need for downloading checkpoint and journal files from active nodes. It can also perform NameNode operation which does not require namespace modification or block location knowledge.

Upgrade, File System Snapshots

Minimizing possible data damage during a system upgrade was a main reason for implementing snapshots in HDFS. The snapshot technique allows the HDFS administrators to save copies of the current status of the file system for emergency uses. For instance, if the upgrade process results in data loss or corruption, administrators can still restore the file system to its previous working status. Only one snapshot instance can exist in HDFS cluster and it is created whenever the system is started. Snapshots can be taken by requests from the administrator. Before taking the snapshot, the NameNode prepares the checkpoint and journal files for the new snapshot based on the old ones. Then the snapshot is created during the handshake between NameNode and DataNodes.

2.3.4. X3D File Format

X3D is a file format that is used for the representation of 3D models [14]. 3D models are represented by various combinations of X3D node types such as box and cone. X3D represents 3D models in a scene-graph architecture. This architecture provides the capability of connecting the parts of the 3D model and building relations between them. X3D is mostly used as a standard exchange file format between CAD tools and between engineers [14]. This section will describe the structure of the X3D file format, the components of the X3D file and the various nodes used for representing geometries.

X3D File Format

There are two different encoding styles for representing X3D scene graph. The first style is the standard VRML encoding. Files represented with this encoding style use the file extension *.x3dv*. The other encoding style is the XML encoding style with the extension *.x3d*. Both of the styles are using different roles for structuring the content of the file and different syntax for representing the geometries. However, both of the styles have the same scene-graph structure and both of them maintain the consistency of the scene-graph [14]. X3D consists of three top-level structure elements. These elements are: **file header**, **X3D root node**, and **X3D graph seen nodes**. Each of these elements is described in more details below.

File header: This section includes basic information that describes the X3D scene. The file header content is not rendered into the 3D scene and it could be one of three encoding types: XML, ClassicVRML, or Compressed Binary [14]. The file header section includes required and optional information structures that describe the capabilities of the X3D file. On the other hand, some of these nodes must be included only one time where other nodes can be included multiple times. The first structure included in the file header is the X3D header statement which is used for defining the X3D file with a specific X3D identifier, X3D version number and the text encoding used.

The second structure included in the file header is the profile statement. X3D supports multiple profiles such as core profile, interactive profile and full profile. Each of these profiles is designed for specific goals and uses. For instance, the core profile is designed to provide the minimal definition required by the X3D browser, while the interactive profile includes most of the nodes needed for creating 3D models and interacting with the scene. The aim of these profiles is to help the browser to implement the required level of X3D specification rather than implementing a large specification at once. Another advantage for profiles is that they help conversion programs convert various file formats [14].

The third structure within the file header is the component statements. Component statements are optional nodes which can be included multiple times in the file header. Each of the X3 profiles consists of several components that describe the capabilities of the profile nodes. Component statements are used to specify the used X3 nodes in the X3 file from the chosen profile [14]. Component statements, enhance the performance of browsers while loading X3D, since they limit the number of loaded components from the profile. The last part of the file header is the META statements. Figure 2.11 shows a simple X3D file with the file header structures. The presented documents define two geometrical shapes, the first object is a simple 3D sphere and the second one is a text object with the value *Hello world!*. From this figure we can see that X3D is declared in the following order.

1. Begins with the X3D header statement.
2. Declare the X3D root node.
3. Declare the **head** node to add the component and META statements (optional).
4. Declare the **Scene** node and declare all its children .
5. Close the **Scene** node and then close the X3D root node .

X3D Root Node: This is the root node for every X3D document and it can be included only once in the X3D file. All the content of the X3D file is placed inside the root node except the X3D header statement. This node is similar to the body node in HTML documents. Some of the file header structures such as used profile and X3D version are set as attributes for this node while other structures like component statements and META statements are placed in the <*head*> tag inside the X3D root node.

X3D Graph Scene Nodes: All the X3D nodes that should be rendered onto the browser must be declared in the <*Scene*> tag. The **scene** node includes all the information about the geometries that compose the 3D model and how they are connected to each other. Moreover, it includes information in regards to the various predefined viewpoint that the 3D model supports. For instance, top, bottom, right and left are common views that are used by engineers. The **scene** node is the root node for all the geometric nodes defined by X3D and can be included only one time in the X3D document. The **scene** node is following the model-centric approach to define 3D objects [14]. This approach allows engineers to define the various geometry nodes in a hierarchical way.

```xml
        <?xml version="1.0" encoding="UTF-8"?>
        <!DOCTYPE X3D PUBLIC "ISO//Web3D//DTD X3D 3.2//EN"
        "http://www.web3d.org/specifications/x3d-3.2.dtd">
        <X3D profile='Immersive' version='3.2'
         xmlns:xsd='http://www.w3.org/2001/XMLSchema-instance'
         xsd:noNamespaceSchemaLocation='http://www.web3d.org/specifications/x3d-3.2.xsd'>
        <head>
          <meta name='title' content='AllenDutton.x3d'/>
          <meta name='creator' content='LT Allen Dutton'/>
          <meta name='description' content='Laser scan by Cyberware'/>
          <meta name='created' content='26 February 2001'/>
          <component level='1' name='DIS'/>
          <component level='1' name='Geospatial'/>
          <component level='1' name='H-Anim'/>
          <component level='3' name='NURBS'/>
        </head>
        <Scene>
          <Group>
            <Viewpoint centerOfRotation='0 -1 0' description='Hello world!'
             position='0 -1 7'/>
            <Transform rotation='0 1 0 3'>
              <Shape>
                <Sphere radius='2.0'/>
                <Appearance>
                  <Material diffuseColor='1.0 0.0 0.0'/>
                </Appearance>
              </Shape>
            </Transform>
            <Transform translation='0 -2 0'>
              <Shape>
                <Text string='"Hello" "world!"'>
                  <FontStyle justify='"MIDDLE" "MIDDLE"'/>
                </Text>
                <Appearance>
                  <Material diffuseColor='0.1 0.5 1'/>
                </Appearance>
              </Shape>
            </Transform>
          </Group>
        </Scene>
   </X3D>
```

Figure 2.11.: Example of X3D file structure in XML type

X3D provides engineers with a wide range of geometries from the basic geometry shapes to the most complex 3D shapes [14]. These nodes can be categorized into two main groups: **Geometry Primitives** , and **Points, Lines and Polygons**. Below is a detailed description on both of the categories.

Geometry Primitives

Geometry primitives are nodes that represent the basic primitive text and shapes supported by X3D [14]. This category includes the following nodes: Shape, Box, Cylinder, Cone, Sphere, Text and FontStyle. The main purpose of this category is to provide the user with a set of predefined geometries that can be used to build more complex 3D models with the minimal efforts and to quickly begin constructing 3D models. Figure 2.12 shows some of the primitive X3D nodes.

Shape Node is the main geometry node in X3D files. It is used to hold all other nodes and to specify their properties. As shown in Figurer 2.11, the shape node contains other geometry nodes and an **Appearance** node that is used to define the properties of the 3D object such as its color.

Box Node is a six-sided rectangular parallelepiped [14]. That means that it is not necessary that the box is a cube, but it can be defined as a cube. Defining a box node can be done by specifying the width, height and depth dimensions. the values of these dimensions must all be positive values.

Cylinder Node is a capped right-angle cylinder with a central axis oriented along the local y-axis. Cylinders can be defined by providing the cylinder radius and the height of the cylinder.

Cone Node consists of a circular base and perpendicular axis that is centered in the local y-axis [14]. To declare a cylinder in X3D, two values must be specified: the height of the cone and the radius of the bottom circle. Both of these values should be non-zero and non-negative values.

Sphere Node is the simplest node to define. It only needs to specify the radius to be successfully declared.

Text and FontStyle Nodes are nodes used to add 2D text strings in the 3D model and to define the feel and look of the text.

Points, Lines and Polygons

Points, Lines and Polygons are the fundamental building blocks for developing arbitrary 3D models [14]. These nodes are usually developed by using CAD tools and not manually by the engineers. This category includes several X3D nodes such as PointSet, Coordinate, IndexedFaceSet, and LineSet nodes.

The **IndexedFaceSet** node allows engineers to create a set of faces (polygons) and group them to form a 3D object. This node is special because it defines these faces based on the coordinates of their vertices's. This is important because it is the basis for transcoding X3D objects to binary objects.

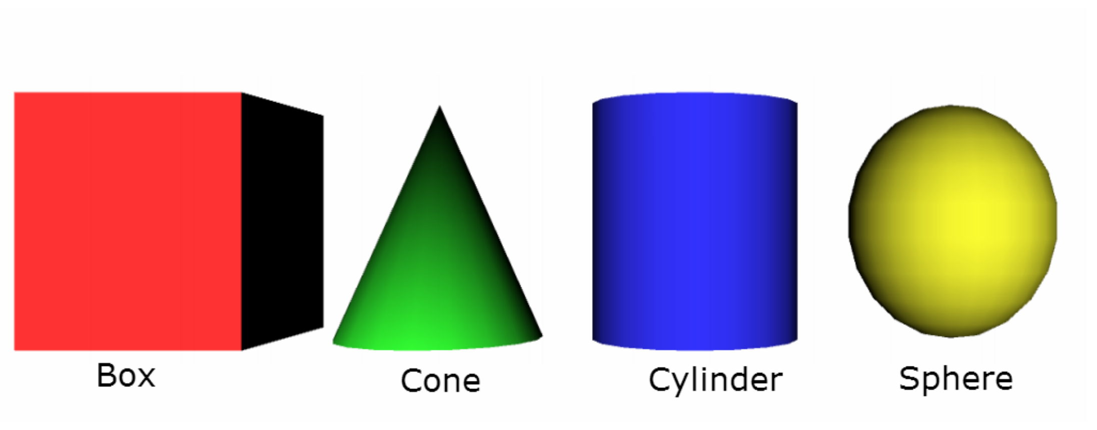

Figure 2.12.: Geometry primitive shapes

2.3.5. X3DOM Framework

X3DOM Framework is a JavaScript library that extends the capabilities of web browsers by providing 3D contents rendering and manipulation runtime [9]. As stated in the Section 2.2 X3DOM has many advantages over all other concurring technologies. X3DOM does not requires the installation of any kind of plug-ins, it provides an access to a DOM object for every X3D node, and it also provides an API for controlling and transforming 3D objects.

The main goal of X3DOM is to provide the application developers with an approach for implementing *life* X3D scene in such a way that allows them to add, remove and change a 3D content through manipulating the corresponding DOM element [8]. Therefore, X3DOM job is not only to connect the HTML world to the X3D world, but to bridge the gap between them. This bridge will provide developers with the missing links between The HTML and X3D. Developers will be able to modify the properties of the geometries using the X3DOM API. For instance, hide/show , clip plane, and rotate functionality can be implemented using X3DOM. The X3DOM architecture consists of three main components which are shown in Figure 2.13 and their functionalities are described below.

User Agent could be any of the supported web browsers such as Firefox and Chrome. The responsibilities of the user agent are to hold the DOM tree, perform the final rendering tasks to integrate 3D elements in the DOM tree , and resolve URIs to load X3D content such as images and music [9].

X3D Runtime main purposes are to support the build and update of X3D scene, handle the scene rendering when change occur, and handle user events such as navigation and piking [9].

Connector is the core component of the X3DOM architecture. It is responsible for connecting both the DOM tree on the user agent and the X3D runtime. It is also responsible

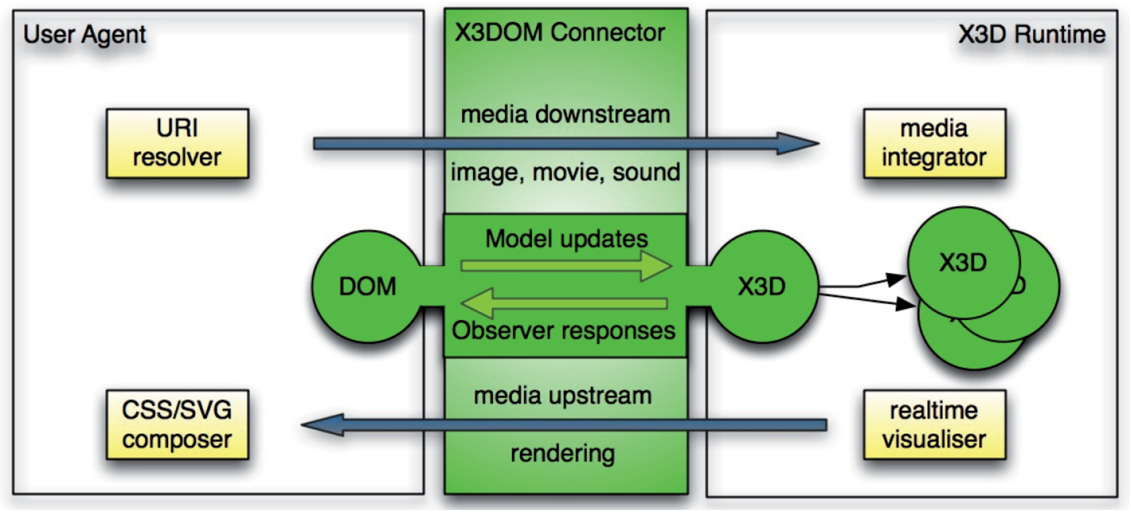

Figure 2.13.: X3DOM system architecture including the UA (i.e. web browser), the Web3D runtime and the X3DOM connector [9]

for distributing the significant changes on both directions and handling the upstream and downstream. The upstream and downstream are methods for loading and rendering images, movies and sound [9].

X3DOM framework provides the application developers with a wide range of features to make the interaction process more convenient, flexible and to enhance the performance of X3D nodes loading [8]. Below is a brief description of a number of the features that X3DOM provides.

User Input and Events is an important feature of X3DOM. This feature is a result of integrating X3D runtime with the DOM tree. It allows the interaction between the browsers and the X3D content. This interaction can be implemented using standard web APIs. The user interaction can be categorized into two main classes: navigation interactions and event interactions [8].

1. Navigation interaction occurs when the end user is driving or moving the camera from one view point to another viewpoint of the 3D model. This type of interaction requires scene redraw each time the viewpoint is changed and only if its viewport is visible [8].

2. DOM events are events like *onclick* or *onmouseover*. These events enable the end user to interact with the 3D models by adding, moving, deleting and inserting 3D objects in the scene graph [8]. They can be classified into three main categories: **UI Events** which are generated by the end user (ie.*click*), **UI Logical Events** such as focus change event, and **Mutation Events** such as node insertion or deletion events.X3DOM supports a wide range of DOM events from the basic **DOM Level 2 Events** like *mouse-*

down, *mouseout*, and *keypress* to **Touch and Multitouch Events** such as *touchstart* event which fires when it is placed on the screen [8].

3. Environmental Events are events that rely on HTML events and they do not require user interaction to be fired [8]. These events can be triggered by either the user interaction or the interaction between objects in the 3D space. For instance, the *3DVisibilityEvent* is fired when the visibility of the corresponding sub-trees is changed. The *3DTransformEvent* fires when the transformation of an object has been changed.

Animation and Automated Scene Changes: X3DOM supports two types of animation technologies in order to provide the availability of declarative animation and automated scene changes. These types are **CSS 3D-Transforms & Animations** and **X3D Interpolators** [8]. Both of these techniques are used to allow the engineers to animate their 3D virtual objects. **CSS** is an easy method for integrating animation with 3D objects, however, it is only useful for simple transformation changes. On the other hand, X3D Interpolators is an animation framework with corresponding nodes. It is supported by X3D standards and allows the creation of more complex animations [8].

2.3.6. XQuery and BaseX

The Extensible Markup Language (XML) is a markup language that aims to describe data objects in a way that is possible to serve, receive, and process these data objects on the web in a similar way to the way that HTML is used over the web [53]. One advantage of XML is that it is readable by humans and machines. An XML processor is an application that reads, processes, and views XML documents content and structure.

XML documents can store large amounts of data objects and it can be considered as a lightweight database XML documents can contain structured and semi-structured data, relational databases, and object repositories [54]. For these reasons, many developers and applications are using XML to store, exchange and present data and therefore, the need of tools or methods to retrieve specific data objects from XML documents has been raised. Today there are multiple methods for retrieving data from XML such as XPath and XQuery. XPath is an XML query language that addresses the nodes of XML documents. Using XPath, users can select specific nodes from an XML document based on the structure of the nodes and their relations.

On the other hand, XQuery is more than a query language that is used for selecting specific XML nodes. XQuery is a high level and declarative functional programming language that can be used to query and transform XML data. XQuery allows developers to write functions or scripts to extract data from XML documents and transform this data to other types of documents. For instance, XQuery can be used to generate HTML pages from XML documents. XQuery is an easy language to learn even by beginners [29]. Today

```
//First Query:
//library/book[price<30]


//Second Query:
for $x in //bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
<book category="Programming">
  <title lang="en">Programming Python</title>
  <author>Mark Lutz</author>
  <year>2005</year>
  <price>30.00</price>
</book>
<book category="Distributed Systems">
  <title lang="en">Distributed Systems Concepts</title>
  <author>George Coulouris</author>
  <year>2005</year>
  <price>29.99</price>
</book>
</library>
```

Figure 2.14.: Example of XML file and XQueries

XQuery has many implementations such as Sirix[30], BaseX[31], eXist[32], MarkLogic[33], Zorba[34] , and W3C [35].

Figure 2.14 shows a simple XML file with a sample of XQuery scripts that is used for manipulating XML files. The first Query is used for selecting all books that cost more than 30. The second query is almost doing the same thing except that it orders the results by title and only selecting the books titles and not the whole book node. The second query is written using the FLWOR[36] expression. Each iteration within the FLWOR statement is executed in a separate thread [24]. In other words, FLWOR statement is following a parallel programming model and this will help us in enhancing our data-parallel approach by using these statements for preparing X3D files for further processing.

[30]https://github.com/sirixdb/sirix Retrieved on 2014-10-14
[31]http://basex.org/ Retrieved on 2014-10-14
[32]http://www.exist-db.org/exist/apps/homepage/index.html Retrieved on 2014-10-14
[33]http://www.marklogic.com/ Retrieved on 2014-10-14
[34]http://www.zorba.io/home Retrieved on 2014-10-14
[35]http://www.w3.org/XML/Query/#implementations Retrieved on 2014-10-14
[36]acronym for ”For, Let, Where, Order by, Return”

In this thesis, we need to use an XML query language to extract the geometric from the X3D documents which are basically XML documents. **Our intent is to use XQuery to write scripts to partition large X3D files into small chunks for later parallel processing**. After reviewing the current implementation of XQuery we decided to use BaseX implementation. BaseX implementation provides a user interface for both Windows and Linux and it is written in java. It also provides the users with a command line interface. However, we chose BaseX because it provides an API for Java to allow developers to write applications that are able to execute XQuery scripts.

2.3.7. Instantreality Framework

One of the main goals of this thesis is to define data-parallel approach for transcoding 3D models. By transcoding we mean converting 3D shapes represented in the X3D files into another format. In our case, we want to convert these shapes into binary format. Implementing a tool for transcoding 3D shapes to binary is out of the scope of this thesis and there are many tools that are available for this purpose.

One of the easiest tools that provide transcoding functionality is the Avalon-Optimize (*apot*[37]) which is included in the InstantReality[38] packages. There are many advantages for using the *apot* tool. First, it supports Mac, Windows and Linux. Second it provides a command line tool which can be used for transcoding 3D models. Third, it provides many features for manipulating 3D models such as analyzing a 3D model for generating basic statistics, mesh restructuring ,and convert geometry nodes from one type to another [59] .

2.3.8. Amazon Elastic Compute Cloud - EC2

Amazon Elastic Compute Cloud (Amazon EC2)[39] is a web service that provides a configurable computing instances on the cloud. EC2s' main goal is to provide developers with a virtual cloud computing environment. It allows developers to create and manage virtual machine instances on the cloud through a web interface. In addition, developers can connect to their virtual machine instances to install application, setup web pages, or even build networks from these instances through *ssh* connection. EC2 enables developers to either use a pre-configured AMI[40] or to create new AMI containing all the needed applications, libraries, configuration and data. EC2 provides AMIs with different operating systems such as Windows and Linux [4].

EC2 provides multiple locations to host their virtual instances. These locations are divided into regions and availability zones [4]. Launching instances in multiple availability zones means that these instances are executed in different data centers [22]. As a result, applications are protected from single location failure. Regions consist of multiple availability zones which are geographically separated. Currently Amazon EC2 is available in

[37]http://www.instantreality.org/downloads/ Retrieved on 2014-10-14

[38]Framework that provides features to support classic Virtual Reality, 3D rendering and advanced Augmented Reality

[39]http://aws.amazon.com/ec2/ Retrieved on 2014-10-14

[40]Amazon Machine Image

Model	vCPU	Mem (GiB)	SSD Storage (GB)
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Table 2.4.: Hardware specification of compute optimized family models

nine regions [4] around the world. These regions are US East (Northern Virginia), US West (Oregon), US West (Northern California), EU (Ireland), Asia Pacific (Singapore), Asia Pacific (Tokyo), Asia Pacific (Sydney), South America (Sao Paulo), and AWS GovCloud[41].

Amazon EC2 provides five family types of virtual instances: general purpose instances, compute optimized instances, GPU instances, memory optimized instances, and storage optimized instances [4]. Each of these types are designed for a specific goal and each of them has a set of different models. Each of the models has a different capacity in terms of RAM size, CPU capacity, vCPUs number, and SSD Storage Table 2.4 shows the announced hardware specifications of the *Compute Optimized* family models. From the table we can see that this family provides the users with high performing processors.

This thesis uses Amazon EC2 web service for the purpose of evaluating the proposed data-parallel transcoding approach. We chose to perform the evaluation on Amazon EC2 platform instead of evaluating it on local computers at AirBus Group for two reasons. The first one is the lack of resources at AirBus Group. The evaluation process requires building powerful Hadoop clusters and that requires powerful computers. Unfortunately these resources are not available currently at AirBus Group and even the buying and preparing of these resources cost both money and time.

The second reason is that the evaluation involves building multiple Hadoop clusters with different specifications and different number of nodes. Building these clusters locally is also time-consuming. However, using Amazon EC2 is easier to build these clusters. For instance, Amazon EC2 allows users to save their virtual instances as AMI and later launch new instances based on these AMI even with different specifications. This feature will save the needed time for installing the desired libraries and building the clusters. For evaluation purposes, this thesis uses virtual instances from compute optimized and general purpose families. More details regarding the used hardware specification of Amazon EC2 is found in *Hadoop Test Environment* Chapter and the *Evaluation* Chapter.

[41] isolated AWS Region designed to allow US government agencies and customers to move sensitive workloads into the cloud

Part II.

Contribution of the Thesis

3. Data-Parallel Transcoding Architecture

Developing a valuable data-parallel transcoding approach requires a good system architecture that ensures achieving the goals of the system. For this reason, we investigated the current technologies and systems that could help us in building a reliable system architecture for transcoding the 3D data. As a result, we found that Hadoop framework and MapReduce programming model can add many advantages to the proposed approach, and thus the architecture of the propose approach is influenced by these technologies.

This chapter aims to describe in detail the proposed data-parallel transcoding architecture and its various phases. The chapter will begin with an overview of the whole architecture in section 3.1. Then, section 3.2 will present the idea behind transcodding X3D data to binary and the method used to transcode 3D geometry into binary geometry. After that, each of the following sections will present a phase of the dynamic architecture. In section 3.3 : Preparsing phase, section 3.4 : Partitiong phase, section 3.5 : *MapReduce* phase and section 3.6 : Deployment phase. Finally, the chapter will end with a brief summary that concludes and addresses the benefits of the proposed architecture.

3.1. Overview

Our main goal is to build a system that is capable of transcoding 3D models from various CAD tools and data formats into a binary format. For achieving this goal we decided to use a distributed environment that enables us to process 3D data into parallel without losing any of the 3D data or the structure of the model. Hadoop and MapReduce were selected for their effectiveness and efficiency in processing big data on commodity clusters.

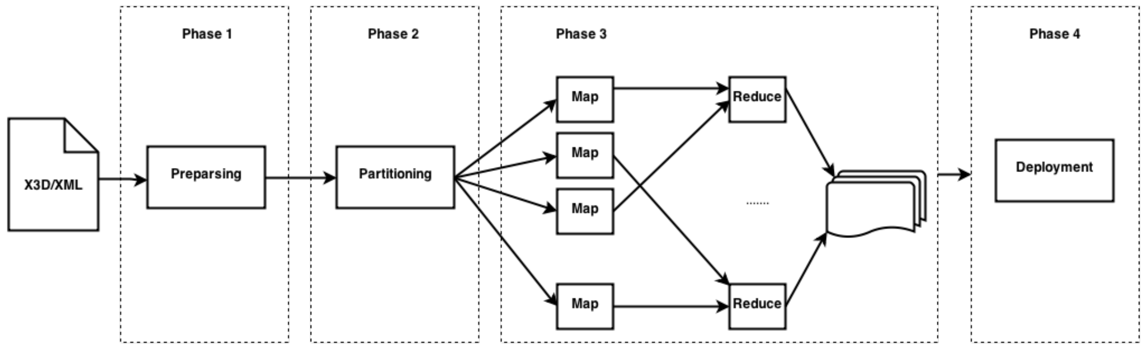

Figure 3.1.: Proposed data-parallel approach architecture

```
<IndexedLineSet coordIndex="">
  <Coordinate/>
  <Color/>
</IndexedLineSet>
```

Table 3.1.: IndexedLineSet structure

Our proposed system architecture which is shown in Figure 3.1 is influenced by the MapReduce programming model. The architecture is a simple pipeline of four phases: *preparsing phase*, *partitioning phase*, *MapReduce phase* and *deployment phase*. For simplicity reasons, we assumed that a conversion tool is available for converting 3D models from various 3D file formats such as 3ds and JT to X3D file format.

During this thesis, we relied on **Polytrans**[1] for converting 3D models to X3D format. After the conversion of the 3D model to X3D file format , the *preparsing* phase is entered and some preparation steps are performed in order to prepare the model for the next phases. Then, during the *partitioning* phase the model will be divided into several X3D chunks. The next step in the pipeline is to run a *MapReduce* job for transcoding all the X3D chunks into binary format. The last phase is to deploy the X3D model into a web server to be available for the end user. The generated web page allows users to perform the basic functionality for viewing 3D models. For instance, the user can rotate the 3D model, zoom in and zoom out, and hide or show the various parts of the model.

3.2. Transcoding Geometry Nodes to Binary-geometry

X3D provides the users with different node types for defining 2D and 3D objects. As described in the Foundation Chapter, X3D provides nodes for defining primitive types of geometries such as cube, cone, and sphere. Users specify one or more general properties in order to create one of these types. For instance, to define a sphere, the user only needs to specify the radius of the sphere and not all its coordinates in the 3D space.

On the other hand, X3D also provides the users with a set of nodes that allows them to define more complex geometries by specifying the coordinates of the object in the 3D space. For instance, in order to define a 3D object that consists of lines and points only, engineers can use the **IndexedLineSet** node for this purpose.

Table 3.1 shows the general structure of the **IndexedLineSet** node which consists of two main parts: *coordIndex* attribute and *Coordinate* node. The color node is not an obligatory node to specify geometry color. The coordinate node is an array that holds the set of 3D points in the 3D space that composes the 3D object. On the contrary, the *coordIndex* attributes aims to provide information for connecting the specified set of 3D points to form the 3D object. This way, engineers are able to draw objects in the 3D space simply

[1]http://www.okino.com/conv/conv.htm Retrieved on 2014-10-21

```
<IndexedFaceSet coordIndex="">
  <Coordinate/>
  <Color/>
  <Normal/>
</IndexedFaceSet>
```

Table 3.2.: IndexedFaceSet structure

by defining the object points in the 3D space and how these points are connected to each other in the 3D space.

IndexedLineSet nodes are a very good option for creating wire-frame 3D models for several reasons. First, rendering these nodes is relatively very fast since the computers only need to draw lines and even these lines are not shaded. Another reason, is that it is much easier for engineers to create wire-frame models with **IndexedLineSet** because they only need to specify the points and their relations and not more. However, complex wire-frame models are very difficult to understand by engineers.

For this reason, X3D provides another node type for drawing 3D objects using faces or polygons instead of lines and points. This node is called the **IndexedFaceSet**. It has the same structure of **IndexedLineSet** except that *coordIndex* is used to represent faces or polygons rather than lines. Table 3.2 shows the general structure of the **IndexedFaceSet**. As the **IndexedLineSet**, the **IndexedFaceSet** also has optional nodes which are the color and normal nodes. Using the **IndexedFaceSet**, node engineers are able to build a more realistic 3D model. They can define the color, shape, and other properties of each of the polygons that compose the 3D object.

Both the **IndexedLineSet** and the **IndexedFaceSet** nodes allow the engineers to create 3D objects by specifying a list of points in the 3D space, and a list of coordinates for connecting these points to construct either lines or polygons. In X3D, these lists are stored as a sequence of numbers in a plain text form. For complex 3D models, these sequences are very large especially for 3D models that model the details of the 3D objects. As a result, the size of the geometries is extremely large and loading these geometries directly into a web browser may lead to crash the browser, or in the best-case scenario it will take a long time to be loaded and rendered into the screen.

A solution for this problem is to compress the size of these geometries in order to be able to load them into a web browser without crashing the browser. Fortunately, X3D supports this idea by providing another node type called **BinaryGeometry**. This node can read the points and the coordinates of a 3D object directly from binary files.

To transcode a 3D object defined in X3D **IndexedFaceSet** or **IndexedLineSet** nodes to **BinaryGeometry** nodes two steps have to be done. First, read the points and coordinates of these nodes, convert them to binary format and save them to the disc. The second step

```
<X3D>
 <Scene>
  <Shape DEF='_G_0'>
   <Appearance>
    <Material ambientIntensity='0.198' diffuseColor='0.8 0.99 0.99'
    shininess='0.5' transparency='0.5'/>
   </Appearance>
   <IndexedFaceSet solid="false" coordIndex="0 1 2 -1 1 3 2 -1 2 3 4 -1 3 3 4 -1 4 3
        9 8 -1 8 9 10 -1 9 9 10 -1 10 9 11 -1 9 12 11 -1 11 12 13 -1 12 12 13 -1 13
        .....
        -1 15563 15565 15564 -1 15564 15565 15566 -1 15565 15567 15566 -1 "
         normalPerVertex="true" >
   <Coordinate point=" 527.303 -1220.73 -56.4536,527.357 -1219.39 -55.941,485.394
        -1218.57 -57.1208,443.553 -1215.83 -57.7831,443.47 -1214.48 -57.2818,443.378
        .....
        -1210.8 -57.9456,401.556 -1212.13 -58.4504,370.494 -1208.49 -58.9437"/>
   </IndexedFaceSet>
  </Shape>
 </Scene>
</X3D>
```

Table 3.3.: 3D Shape defined using IndexedFaceSet node

is to replace **IndexedFaceSet** or **IndexedLineSet** nodes in the X3D file with a **BinaryGeometry** node that holds a link for the binary files that contains all the defined points and coordinates. Table 3.3 shows a 3D geometry before the transcoding process and Table 3.4 shows the same 3D geometry after the transcoding process. In this thesis, we relied on the Avalon-Optimizer (*aopt*) command line from the InstantReality packages[2] for transcoding the geometries into binary format.

3.3. Phase I: PreParsing

As stated in the overview of this chapter, we assumed that an X3D representation for the 3D model that we want to transcode is available. The X3D representation can be generated using various methods and techniques. One method is to build the 3D model using commercial or open-source X3D editors such as **X3D-Edit Authoring Tool**[3]. However, these tools are designed to build simple 3D objects and it is very difficult to build a detailed 3D model using such tools. Engineers are using more sophisticated applications for building their 3D models such as CATIA and AutoCAD.

Fortunately, there are some tools and applications in the market today that enable engineers to export their models to various file formats including X3D file format. In addition, some of the CAD applications provide the user with the ability to export the 3D models in a range of other file formats. During this thesis, we worked mainly with 3D models which are developed by using CATIA. Unfortunately, CATIA does not support the conversion

[2]http://www.instantreality.org/downloads/ Retrieved on 2014-10-21
[3]https://savage.nps.edu/X3D-Edit/ Retrieved on 2014-10-21

```
<X3D>
 <Scene>
  <Shape DEF='_G_0'>
   <Appearance>
    <Material ambientIntensity='0.198' diffuseColor='0.8 0.99 0.99'
    shininess='0.5' transparency='0.5'/>
   </Appearance>
   <BinaryGeometry DEF='BG_0' vertexCount='34' primType='"TRIANGLESTRIP"'
   size='1 1 0.10000000149' index='binGeo/BG_0_indexBinary.bin'
   coord='binGeo/BG_0_coordNormalBinary.bin+8' coordType='Uint16'
   normalAsSphericalCoordinates='true'/>
  </Shape>
 </Scene>
</X3D>
```

Table 3.4.: 3D shape after transcoding process

of the 3D models into X3D file format directly. For this reason we relied on a commercial conversion called PolyTrans[4]. This tool provides the users with a wide range of conversion methods for a various range of file formats such as CATProduct, STEP, and JT files.

The *preparsing* phase is the first step in the proposed data-parallel architecture. The main goal of this phase is to prepare the X3D file that represents the 3D model for further processing in the next phases. In this phase, three different main tasks are performed on the input file to prepare it for the *partitioning* phase, these tasks are described in details below.

3.3.1. Remove Unwanted Nodes

Usually X3D files contain nodes that are not relevant for rendering the 3D model onto the browser, or do not affect the rendering process. For instance, META, Component, WorldInfo, Viewpoint, and PointLight are nodes used to either add more effects to the rendered scene or to control the loaded X3D profiles and components. These nodes are not required for the rendering process and even without them X3DOM still can render the 3D model and load it onto the browser screen. For these reasons and to make the *partitioning* phase as simple as possible, we decided to remove these nodes from the X3D file in this phase.

3.3.2. Replace USE in the X3D file

Often, when engineers build 3D models they define global nodes and reuse these nodes all over the 3D model. These nodes could be any node types defined by X3D standards. However, the common used nodes as global nodes are **Material** nodes; that specifies the color and other properties of the 3D surface material used to model the 3D object such as the transparency or shininess. The **Appearance** nodes are nodes used to specify the visual

[4]http://www.okino.com/conv/conv.htm Retrieved on 2014-10-21

```
<X3D>
 <Scene>
  <Transform DEF='House'>
   <Transform DEF='LivingRoom'>
    <Transform DEF='LivingRoomWalls' translation="0 1 0">
     <Group DEF="">
      <Shape DEF='RoomWalls'>
       <Appearance>
        <Material diffuseColor='1 1 0'/>
       </Appearance>
       <IndexedFaceSet solid="false"
       coordIndex="0 1 2 -1 1 3 2 -1 2 3 4 -1 3 3 4 -1 4" normalPerVertex="true" >
       <Coordinate point=" 527.303 -1220.73 -56.4536,527.357
        -1219.39 -55.941,485.394"/>
       </IndexedFaceSet>
      </Shape>
     </Group>
    </Transform>
    <Transform DEF='LivingRoomRoof' translation="0 2.5 0">
     <Group DEF="">
      <Shape DEF='RoomRoof'>
       <Appearance>
        <Material diffuseColor='1 0 0'/>
       </Appearance>
       <IndexedFaceSet solid="false"
       coordIndex="0 1 2 -1 1 3 2 -1 2 3 4 -1 3 3 4 -1 4" normalPerVertex="true" >
        <Coordinate point=" 527.303 -1220.73 -56.4536,
        527.357 -1219.39 -55.941,485.394"/>
       </IndexedFaceSet>
      </Shape>
     </Group>
    </Transform>
   <Transform>
   <Transform DEF='Kitchen'>
    <!-- reuse, make smaller -->
    <Transform DEF='KitchenWalls' translation="0 1 0" scale="0.5 0.5 0.5">
     <Group DEF="">
      <Shape USE="RoomWalls"/>
     </Group>
    </Transform>
    <Transform DEF='KitchenRoof' translation="0 2.5 0" scale="0.5 0.5 0.5">
     <Group DEF="">
      <Shape USE="RoomRoof"/>
     </Group>
    </Transform>
   <Transform>
  </Transform>
 </Scene>
</X3D>
```

Table 3.5.: Reuse nodes in X3D

properties of geometry and to hold the Material and Texture nodes. The **Shape** nodes are nodes used to hold the defined geometry and its appearance properties.

The last common used node type, as global node is the **Group** node.The **Group** node is used to combine a set of shapes nodes into one group. Table 3.5 shows how global nodes can be defined in X3D and reused on the rest of the X3D document. As shown in the Table, X3D defines global nodes by adding **DEF** attribute for the node and specifies a global identifier for that node. To reuse the node, X3D provides the **USE** attributes; the value of this attribute should be the one of the **DEF** values in the X3D file.

X3DOM are able to render models that include **DEF** and **USE** attributes and load them into the browser as long as the **DEF** and **USE** belong to the same X3D file. Unfortunately, we are intending to partition the X3D file into small chunks in the next phase. As a result, X3DOM will not be able to render the 3D model completely if it includes **DEF** and **USE** attributes.

To solve this issue we decided to replace all the occurrences of **USE** with their corresponding **DEF** nodes in the X3D file. Then to remove the **DEF** attribute from all nodes since it does not add anything to the 3D model and it will be redundant over the X3D file. This solution works quietly well and the X3DOM will be able to render the 3D models completely, because we eliminate the existence of the nodes that use **DEF** and **USE** attribute. After the replace step every shape in the X3D file will be independent from all other shapes in the document and it can be extracted and viewed separately.

3.3.3. Create Product Breakdown Tree

The last task in the *preparsing* phase is to scan the X3D file, and generate a HTML list that describes all the shapes included in the X3D file. This list aims to organize the X3D nodes in a hierarchical way. We call this list the **product breakdown tree**, since it is used to display all the parts that compose the 3D object and their relations to each other as parent-child relationship. Table 3.6 shows the expected product breakdown tree for the X3D shown in Table 3.5 . This tree will be used as a part of the web page that views and control the transcoded 3D model. This web page will be created as the last phase of the transcoding approach.

3.4. Phase II: Partition X3D Files

The second phase in the proposed architecture is the *partitioning* phase. This phase aims to break down the X3D files into small chunks. The *partitioning* phase should not lead to any type of data loss. Moreover, the *partitioning* phase should create independent files that can be processed in parallel without affecting the final result. In other words, the order of processing the extracted files should not have any influence on the final results of the transcoding process.

```
<ul>
 <li>House
  <ul>
   <li>LivingRoom
    <ul>
     <li>LivingRoomWalls</li>
     <li>LivingRoomRoof</li>
    </ul>
   </li>
   <li>Kitchen
    <ul>
     <li>KitchenWalls</li>
     <li>KitchenRoof</li>
    </ul>
   </li>
  </ul>
 </li>
</ul>
```

Table 3.6.: Expected product breakdown tree structure

X3D files are XML files and therefore, any idea of partitioning X3D files into several files based on the line number, or the content size is not applicable. Although this would not lead to data loss, it will lead to loss in the structure of the X3D documents. In addition, the order of the partitioned files plays a major role for reconstructing the original X3D file. This will lead to a high dependency between these chunks and they cannot be transcoded in parallel. Furthermore, partitioning X3D files in this way does not guarantee that each of the partitioned files includes one or more geometries and it does not guarantee that geometries are not split over multiple files.

After studying the structure of X3D documents carefully and trying to come out with a general structure and method for partitioning X3D documents, we conclude that X3D documents represent two main categories: shapes and parts. Shapes are X3D nodes that include the geometries data and their appearance details. On the other hand, X3D *Transform* nodes represent the parts. Each of these parts corresponds to one specific physical part of the 3D object. For instance, in a 3D model that represents an airplane, the right wing, left wing and the rudder are considered to be high-level parts of the airplane. These parts are composed of sub parts and their sub parts are composed either of shapes or sub parts.

As a result of this understanding, we decided that the *partitioning* phase should include the following steps in order to partition the X3D files into equal chunks of files that can be processed in parallel without affecting the final results or lead to data or structure loss.

Extract All Shapes: This step aims to extract every single shape that exists in the X3D document and stores it into a separate file on the disk. The result of this step is a file hierarchy that represents all the shapes within a given X3D file. In other words, the result is a folder that contains sub folders that represent each of the shapes in the X3D document. This way, we ensure that each of the extracted files includes only one shape and all the extracted files can be processed in parallel with any order. After extracting each shape in a separate file, the shape definition in the X3D documents should be replaced with an *inline* tag that is connected to the corresponding extracted shape.

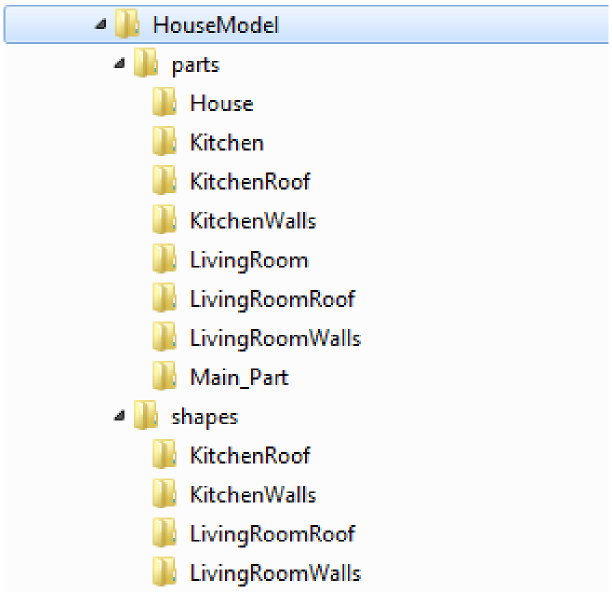
HouseModel
parts
House
Kitchen
KitchenRoof
KitchenWalls
LivingRoom
LivingRoomRoof
LivingRoomWalls
Main_Part
shapes
KitchenRoof
KitchenWalls
LivingRoomRoof
LivingRoomWalls

Figure 3.2.: Expected output of partitioning phase

Extract All Parts: This step focuses on extracting all the high-level and sub parts of the 3D object included in the X3D document. The input for this step is the X3D document resulted from the previous step, where the only difference between this document and the original one is that all shapes are replaced by *inline* tags.

As stated above, parts are represented by *Transform* nodes. The *Transform* nodes are used to group shapes and specify a local coordinate system for their children. Not all the *Transform* nodes exist on the X3D file that represents a physical part of the modeled object, only *Transform* nodes that include **DEF** attribute are considered to represent the object physical parts. For all these *Transform* nodes, we need to extract them into separate files and store them on the file system in a similar way to the *Extract All Shapes* method. The result of this step will also be a folder that includes all the sub parts that represent the 3D models. All the parts extracted in this phase include *inline* tags that link the parts to the corresponding shapes files extracted in the the previous step instead of the real shape definition.

Extract The Main Part: This is the last step in this phase and it is very similar to the previous step. The main purpose of this step is to extract the root part of the 3D model and store it in the file system. The root part includes all the sub parts that compose the 3D model and using the main part users can view the complete model and all its details. This step is necessary because the main part will be used later in constructing the view web page and will serve as the default part for the viewer.

Figure 3.2 shows the expected output of the previous three steps. As mentioned before, the result of performing these steps are a file hierarchy that describes the shapes and parts that compose the 3D model.

3.5. Phase III: MapReduce Jobs

The *MapReduce* phase is the phase that is responsible for transcoding the geometries generated from the partitioned files to binary geometries. We are looking for a data-parallel approach for transcoding the 3D geometries; therefore, for this reason and reasons stated in the **Foundation** Chapter, the efficiency and the capability of Hadoop and MapReduce programming model, we decided to implement the transcoding process using these technologies. There are two main tasks that should be done during this phase. The first task is to build a Hadoop cluster that is capable of hosting and executing MapReduce jobs. The second task is to implement the MapReduce jobs needed to transcode the geometries into binaries.

3.5.1. Setup Hadoop Cluster

Building a Hadoop cluster is an essential part of this phase simply because we cannot run MapReduce jobs without having a Hadoop cluster. The specification of Hadoop cluster depends on the specification of the machines that is used to build the cluster and how much resources these machines could contribute to the cluster.

During this thesis, multiple Hadoop clusters were built either for the development purposes or for the evaluation purposes. The types of the clusters also vary from single node cluster to multiple node clusters. More information about the clusters used during the development and evaluation of MapReduce jobs can be found in Chapter 4.

3.5.2. Design and Implements Mapreduce Jobs

Hadoop is developed using the Java programming language and therefore to develop a native MapReduce job, a developer should use Java as programming language for developing these jobs. However, Hadoop provides an alternative for developing MapReduce jobs using Java, which is Hadoop streaming technology.

Hadoop streaming is another technology or method to develop MapReduce jobs. Using this method a developer can develop MapReduce jobs in wide a range of programming languages like Python, Perl and Ruby. Then, to execute the implemented MapReduce jobs on Hadoop, the developer use a special streaming JAR file provided by Hadoop to start these jobs. The only constrain on the jobs developed using this method is that both the **Map** and **Reduce** functions should read data from the standard input and write the output to the standard output.

MapReduce jobs are responsible for transcoding the geometries into binaries using the *aopt* tool. The main goal of these jobs is to iterate over all partitioned files in the previous phase, and to try to convert these X3D files to binary. The results will be that binary files and X3D files which contain links to the generated binaries. The results of MapReduce jobs will be stored on the Hadoop File system (HDFS). The results will be stored in the same way that the partitioned files were stored in the previous phase. Since only shapes include geometries and not the parts, we only need to run the MapReduce jobs against the shape files.

3.6. Phase IV: Deployment

The deployment phase is the last phase in our approach, and it aims to create and setup a web pager for viewing the 3D model on a web browser. This web page is simply a 3D viewer that allows the end users to view the complete 3D model, select to view specific parts of the 3D model, hide and show geometries from the screen, and rotate the 3D model or change the viewpoint in the 3D space. The main tasks done during this phase are listed below.

1. Create folder on the web server that serves as the home folder for the model web page.
2. Copy transcoded shapes files from HDFS to the home directory.
3. Copy the parts of the X3D to the home directory.
4. Copy all required JavaScript and CSS files to the home directory. These files include the X3DOM library files.
5. Integrate the product breakdown tree into the 3D viewer template. The viewer template integrates the X3DOM library in HTML and link the X3D scene inside the template to the main part of the transcoded model.
6. Save the integrated web page into the home directory.

After performing these tasks, the users should be able to access and view the transcoded 3D model using a web browser that support X3DOM such as Chrome or Mozilla Firefox. Users will also be able to hide and view parts of the 3D Model. In addition, they can see the product breakdown tree on the left side of the web page and select which parts to display on the screen.

3.7. Summary

To sum up, our proposed data-parallel transcoding approach is a simple pipeline that takes an X3D file as input and it consists of four main phases. The result of the proposed approach is a 3D viewer that is used to view and control the transcoded 3D model into a browser screen. The four main phases of the proposed approach are briefly described below.

Phase 1 - The Preparsing Phase: This phase aims to prepare the X3D document for further processing by removing some of the X3D nodes, and replacing all the nodes that include **USE** attributes. The product breakdown structure is generated during this phase.

Phase 2 - The Partitioning Phase: During this phase all shapes and parts included in the X3D document are extracted into separate files and stored in the file system. In addition, a special part called the Main part are also extracted during this phase. This part serves as the root part for the 3D model and it includes all the parts in the 3D model.

Phase 3 - The MapReduce Phase: This phase is responsible for executing the MapReduce jobs on Hadoop cluster to transcode the geometries to binary files. MapReduce jobs only transcodes the shapes since the parts do not include any geometries.

Phase 4 - The Deployment Phase: In this phase, a web page for viewing and controlling the transcoded model is created from a predefined template and deployed on a web server to be accessed by the end users.

4. Hadoop Test Environments

4.1. Overview

Developing and executing a MapReduce job requires building a host environment that is able to execute the implemented MapReduce job and to provide the distributed processing of large data. In our case, this environment is the Hadoop framework. Hadoop environments (clusters) can consist of a single machine (node) or up to thousands of nodes. Each of the nodes that compose the cluster is contributing some of its resources to the cluster. For instance, imagine that we have four computers with **8** GB of RAM for each of them, with these resources we can build a Hadoop cluster with RAM size up to **32** GB if the computers contributes all their RAM resources.

The power of Hadoop cluster depends on how much resources are the nodes contributing to the cluster, and it also depends on the processing power of each of the individual nodes. Hadoop cluster and framework can be installed on both Windows and Ubuntu operating systems. This thesis involved building multiple Hadoop environments with different capacities and specifications on the Ubuntu operating system.

Clusters built during the practical work of this thesis can be classified into two main categories: local clusters and remote clusters. The local clusters include a single node cluster and a dual node cluster. The main purposes of the local clusters are the development and testing of the MapReduce jobs for transcoding the X3D data. Therefore, local clusters are not powerful clusters and cannot be used in a production environment.

On the other hand, remote clusters are the clusters built using the EC2 web service. EC2 clusters are more powerful than the local clusters in terms of RAM capacity and CPU processing power. In addition, EC2 clusters consist of more nodes than the local clusters; For instance, the smallest cluster consists of four nodes. The main purpose of all EC2 clusters is to evaluate the implemented MapReduce job with different system and environment parameters.

This chapter aims to give the reader a clear understanding on the set of the clusters built and used during the thesis. Nodes hardware specification, Hadoop setup and configuration will be described in details for the clusters developed in the thesis. This chapter begins by describing the Hadoop prerequisites. Next, it describes the hardware specifications for the local and EC2 clusters. After that, it presents the Hadoop installation and configuration steps used to build these clusters.

4.2. Hadoop Prerequisites

Before installing and setting up a Hadoop cluster a set of prerequisites should be done. These prerequisites vary from installing required libraries to configuring the Ubuntu operating system. Without setting up all these requests the Hadoop framework will not work properly and even in many cases it will not work at all. For instance, if the network is not setup correctly, nodes of the clusters will face communication errors and the system will fail to launch the Hadoop cluster. Below is a description of all the important prerequisites for launching a Hadoop cluster.

Java development kit (JDK) : Hadoop framework is developed using Java programming language and therefore, it is necessary to install Java JDK in order to be able to launch a Hadoop cluster and execute the MapReduce jobs. Without installing the JDK, there is no way to run a Hadoop cluster. Hadoop requires a working JDK 1.5+ installation. All the clusters developed during this thesis have a working Oracle JDK 6 installation.

Dedicated Hadoop system user : This step is not essential for running a Hadoop cluster, but, it is an important and recommended step. Adding a dedicated user of the Hadoop cluster will help us manage and separate the Hadoop installation from other installed applications on the same machine, and it help us to define security permissions for the Hadoop installation. All the cluster nodes developed in this thesis have a dedicated system user for the Hadoop installation called *hduser*.

SSH Communication: Hadoop clusters are composed of multiple machine nodes. These nodes need to communicate and exchange data with each other. Hadoop framework uses SSH connections as the communication protocol between the cluster nodes. Hadoop also uses the SSH connection to manage the cluster nodes and the data stored in these nodes.

Three steps need to be done in order to have a working SHH connection on a Hadoop cluster. First, installing SHH server and client on all the cluster nodes. There are many implementations of SSH protocol, some of them are commercial and some of them are for free. In this thesis, all the developed clusters use the *OpenSSH* implementation which is a free implementation of the SSH protocol.

The second step is to generate a RSA key pair for all the *hdusers* on the cluster nodes. The RSA is an encrypting method that is used wildly to secure the data transmissions between computers on a network environment. These keys will be used to connect using SHH connections from one node to another node instead of using passwords.

Each Hadoop cluster is composed of one *master* node and a given number of *slave* nodes. The *master* node is the node that is responsible for starting the cluster, manage the cluster nodes, and shutdown the cluster. Therefore, it is very important that the *master* node can reach and access all the *slave* nodes as well as read and write data on these nodes.

Hard Disc	500 GB
RAM	8 GB
CPU	Intel (R) Xeon(R) CPU X3470@ 2.93GHz
Operating System	Ubuntu Server 14.04 LTS , 64-bit

Table 4.1.: Single node hardware specification

The Third step is to enable the SSH access from the *master* node to all the cluster nodes. This step can be done by simply adding the public key of the *master* node to the authorized keys list on the rest of the nodes. At this stage, an important step is to add the access from the *master* node to itself in the same way. This is an important step because Hadoop treats the *master* node as the rest of the nodes and it uses SSH protocol to manage the *master* node.

Disabling IPv6: Since Hadoop uses **IPv4** for the cluster nodes and the *0.0.0.0* address may be linked to an **IPv6** on Ubuntu, we decided to disable **IPv6** on all the nodes used in this thesis.

AOPT Command line: The transcoding process is done during the MapReduce phase on the local file system of the cluster nodes. This job is performed using the *aopt* tool; therefore, all the cluster nodes should be able to run the *aopt* command in order to be able to transcode the 3D data locally. The *aopt* tool is included in the Instantreality framework. By installing this framework, the *aopt* tool will be available for transcoding the 3D data.

4.3. Hardware Specification

During this thesis three main types of clusters were developed. The first type is the single node cluster, which consists of only one machine. The second type is the dual node cluster, which consists of two nodes, and the third type is the multiple node cluster, which consists of three nodes or more. The single and dual node clusters were developed locally at Airbus Group using the company machines, while the multiple node clusters were developed using the EC2 web service. This section will describe the hardware specification of the nodes that composes all the implemented clusters during this thesis.

4.3.1. Single Node Cluster

This is the first cluster developed during the thesis and the main purpose of this cluster is the development of several data-parallel transcoding approaches using MapReduce programming model. As the name specifies, this cluster consists of only one computer. Therefore, this cluster can provide data-parallel processing, but the processing of the data will not be distributed on multiple machines. Table 4.1 shows the specification of the computer used to build this cluster. As shown in the table the computer is considered a powerful machine and it is sufficient for development purposes.

Hard Disc	250 GB
RAM	2 GB
CPU	Intel (R) Xeon(R) CPU MP E7450@ 2.40GHz
Operating System	Ubuntu Server 14.04 LTS , 64-bit

Table 4.2.: Dual node hardware specification

4.3.2. Dual Node Cluster

The dual node cluster is a Hadoop cluster that is composed of two computers. The main aim of this cluster is to extend the power of the Hadoop single node cluster by adding more nodes to the cluster. The hardware specification of the first computer used to build this cluster is the same as the hardware specification of the computer used in the single node cluster. These specifications are listed in Table 4.1. On the other hand, the hardware specification of the second computer is described in Table 4.2. From these tables we can find that the hardware specification for the first and second computer is not equivalent. We could not build the dual core cluster with equivalent nodes mainly for cost and time reasons.

4.3.3. EC2 Clusters

Amazon Elastic Compute Cloud web service is used in this thesis as an evaluation platform for building Hadoop clusters. For the evaluation purposes, multiple Hadoop clusters were developed during this thesis. These clusters used to evaluate the implemented MapReduce jobs and the Hadoop environment properties such as the evaluation of the effects of shape type, shapes number and memory capacity on the transcoding time.

All the clusters developed during this thesis belong to three EC2 families, the *General Purpose* family, the *Memory Optimized* family and the *Compute Optimized* family. The *General Purpose* family provides the user with virtual computer instances that have a balance of the memory, network, and compute resources. On the other hand, the *Compute Optimized* family provides the user with virtual computer instances that have the highest performing processors. Table 4.3 and Table 4.4 show the hardware specifications of the instances that belong to both of the families. From the *General Purpose* and the *Memory Optimized* families we used only one model which are *m3.2xlarge* and *r3.4xlarge*. On the contrary, we used four models from the *Compute Optimize* family.

The evaluation process evaluates both the system parameters and the environment parameters. For all the evaluation cases that aim to evaluate the system parameters such as the shapes' number and the shapes' type, only one Hadoop cluster is used. The system evaluation cluster consists of four virtual machines from *General Purpose* family. All the four nodes have the same hardware specification and all of them belong to the same model which is *m3.2xlarge*. The instances that belong to the *m3.2xlarge* have **30** GB of RAM. However, each of the nodes that compose the system evaluation cluster contribute only **8** GB of RAM to the cluster.

Model	vCPU	Memory GiB	SSD Storage GB
m3.medium	1	3.75	1 x 4
m3.large	2	7.5	1 x 32
m3.xlarge	4	15	1 x 40
m3.2xlarge	8	30	1 x 80

Table 4.3.: General purpose family models

Model	vCPU	Memory GiB	SSD Storage GB
c3.large	2	3.75	2 x 16
c3.xlarge	4	7.5	2 x 40
c3.2xlarge	8	15	2 x 80
c3.4xlarge	16	30	2 x 160
c3.8xlarge	32	60	2 x 320

Table 4.4.: Compute optimize family models

On the other hand, each of the evaluation cases that evaluate the environment parameters involve the creation of multiple Hadoop clusters. The first evaluation case is the memory capacity case. This evaluation case involves the creation of **7** Hadoop clusters that belong to the *r3.4xlarge* from the *Memory Optimized* family. The only difference between these clusters is the memory capacity of each of them. For instance, the nodes of the first cluster contributes only **1** GB of RAM for the cluster which means the cluster's total capacity is **4** GB, While the last cluster has a total memory capacity of **256** GB.

The second evaluation case is the virtual CPUs number case. During this evaluation case four Hadoop clusters were built, each of these clusters consists of four nodes and each of them belong to one model from the *Compute Optimize*. For instance, the first clusters is composed of four nodes from model *c3.xlarge* model, while the nodes of the fourth cluster is composed of *c3.8xlarge* model. The difference between these clusters is the number of the virtual CPUs in the instances that compose the cluster.

The third evaluation case is the evaluation of the cluster size. During this case, eight Hadoop clusters were built. All these clusters are constructed using instances that belong to the *m3.2xlarge* from *General Purpose* family. The difference between these clusters are the number of the active nodes in each of them.

4.4. Hadoop Installation and Configuration

Once all the prerequisites of Hadoop framework are installed and configured, we can continue installing and setting up the Hadoop cluster. This section describes the installation and configuration steps that are performed to build the clusters used in this thesis: the local and EC2 clusters. The installing and configuration steps are similar in all the created clusters and can be generalized to create any Hadoop cluster.

Configuration File	Description
yarn-site.xml	ResourceManager and NodeManager configuration items
core-site.xml	Hadoop site-specific configuration
mapred-site.xml	MapReduce configurations
hdfs-site.xml	Hadoop Distributed file system configuration for NameNode and DataNode

Table 4.5.: Hadoop configuration files

4.4.1. Hadoop Installation

Installing Hadoop framework is not a complicated process especially for the single node cluster.The reason for this is that there is only one computer in the cluster and there are no network or communication issues to worry about. In this thesis, Hadoop framework version 2.2.0 was used to construct all the created clusters, the local clusters and the EC2 clusters. Two steps are required in order to install the Hadoop framework on Ubuntu operating system .

The first step is to download the Hadoop framework from Apache website and extract it in the local file system. The owner of the Hadoop directory should be the *hduser*. The second step is to modify the **.bashrc** file for the *hduser* to include the Hadoop home directory and Java Home directory. These two paths should also be included in the **PATH** variable. Once these steps are done, the Hadoop installation is completed. However, before starting the Hadoop cluster we need to specify the basic configuration of the Hadoop cluster.

The installation steps are the same in all the created clusters in this thesis. On the single node cluster these steps are done only one time on the computer that composes the cluster. However, in the dual and EC2 cluster, every node in the cluster should include the Hadoop installation files in its file system . Moreover, in these clusters the path to the Hadoop home directory should be the same in all the nodes that compose the cluster.

4.4.2. Hadoop Configuration

Hadoop provides a set of XML files that is used to configure the various Hadoop properties. Table 4.5 shows some of the Hadoop 2.2.0 configuration files and the purpose of each of these files. These configuration files can be found under *hadoop/etc/hadoop/*. These configuration files determine whether the nature of the Hadoop cluster is a single node or multiple node cluster; and some of the Hadoop specific configuration like the port used for the web interlace of the HDFS.

Single node cluster configuration

In order to configure a Hadoop single node cluster, the below steps should be performed on the configuration files specified.

Java and Hadoop Variables Edit the *hadoop/etc/hadoop/hadoop-env.sh* file to add Java home directory and Hadoop global variables. Figure 4.1 shows the lines that define the values of these variables. These lines should be added to the document right after the comment *The java implementation to use.*

```
export JAVA_HOME="`dirname $(readlink /etc/alternatives/java)`/../"
export HADOOP_COMMON_LIB_NATIVE_DIR="~/hadoop/lib"
export HADOOP_OPTS="$HADOOP_OPTS -Djava.library.path=~/hadoop/lib"
```

Figure 4.1.: Hadoop configuration : hadoop-env.sh

File System Modify the *hadoop/etc/hadoop/core-site.xml* file to add the default URL for the Hadoop file system. Figure 4.2 shows how this parameter should be configured. Simply a new property node should be added to the file with the provided name and value. This URL will be used by Hadoop to internally access the HDFS files system. The default port is **9000**, however, any port can be used for this purpose.

```
<property>
 <name>fs.default.name</name>
 <value>hdfs://localhost:9000</value>
</property>
```

Figure 4.2.: Hadoop configuration: core-site.xml

```
<property>
 <name>dfs.replication</name>
 <value>1</value>
</property>
<property>
 <name>dfs.namenode.name.dir</name>
 <value>${user.home}/hadoop/data/namenode</value>
</property>
<property>
 <name>dfs.datanode.data.dir</name>
 <value>${user.home}/hadoop/data/datanode</value>
</property>
```

Figure 4.3.: Hadoop configuration: hdfs-site.xml

HDFS Configuration Inside *configuration* tag of the file *hadoop/etc/hadoop/hdfs-site.xml* add the properties shown in Figure 4.3. These properties define the path to the *datanode* and *namenode* directories. In Hadoop clusters that are composed from more than one machine, the *datanode* and *namenode* directories should exist on every node in the cluster and should have the same path. In addition, *dfs.replication* property defines the number of the replications for each of the data blocks. Since there is only one machine in the single node cluster we set this property to **1**.

Node Manager Edit the *hadoop/etc/hadoop/yarn-site.xml* to set the shuffle service and class that needs to be set for MapReduce applications. Figure 4.4 shows how these properties can be defined.

```
<property>
 <name>yarn.nodemanager.aux-services</name>
 <value>mapreduce_shuffle</value>
</property>
<property>
 <name>yarn.nodemanager.aux-services.mapreduce.shuffle.class</name>
 <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

Figure 4.4.: Hadoop configuration: yarn-site.xml

MapReduce The file *hadoop/etc/hadoop/mapred-site.xml* is used to specify the used MapReduce framework and its parameters. Figure 4.5 shows the XML property that should be added to the *mapred-site.xml* in order to specify **YARN** as the MapReduce framework. **YARN** is the new MapReduce implementation included in Hadoop after version **0.23**. It is also called MapReduce 2.0.

```
<property>
 <name>mapreduce.framework.name</name>
 <value>yarn</value>
</property>
```

Figure 4.5.: Hadoop configuration: mapred-site.xml

Dual node and EC2 clusters Configuration

The most important difference between dual node and EC2 clusters is the size of the cluster. The single node consists of only one computer, while the dual node and EC2 clusters consist of more than one node which means that the Hadoop framework should handle the communication and data exchange between several computers in a network environment. For this reason, dual and EC2 clusters are called multiple node clusters. In these clusters, one node should be defined as a *master* node while all other nodes should be defined as *slaves* to the *master* node. The *master* node is the node that is responsible for running the NameNode. Other nodes only run an instances of the DataNode. Before starting to configure a multiple node cluster, all the prerequisites described above should be installed on all the nodes of the cluster.

In addition, another important issue in developing a multiple node cluster is to make sure that all the nodes of the cluster can reach each other. For simplicity reasons, we decided to connect all the nodes in the dual and EC2 clusters in a simple network without firewalls. This way, we will reduce the risk of network errors and issues. Each of the nodes in these clusters contain a list of the IPs and *hostnames* of the nodes that composes the cluster. This list is stored in the */etc/hosts* file.

Configuring a dual node or EC2 Hadoop cluster is almost the same as configuring the single node cluster. The first step in configuring these clusters, is to perform all the configuration steps done in the single node cluster for each of the node that composes the multiple node cluster. However, some of the single node configurations are replaced by other values that are suitable for the multiple node cluster, and there are some configurations that apply only to the multiple node cluster. All the configuration changes and the new configurations for the multiple node cluster is listed below.

File System Modify the *hadoop/etc/hadoop/core-site.xml* and change *fs.default.name* with the domain name of the master node instead of the local host. For instance, the value may look like *hdfs://masternode:9000*. This step should be done in all the instances that compose the cluster.

HDFS Configuration Since multiple node clusters consist of more than one node, we can set the value of *dfs.replication* property in the file *hadoop/etc/hadoop/hdfs-site.xml* to a value that is larger than *1*. However, the value of this property should not exceed the number of the nodes that composes the Hadoop cluster. Like the previous property the *dfs.replication* should be changed on all the nodes of the cluster.

MapReduce Add new property to the file *hadoop/etc/hadoop/mapred-site.xml*. This property will define the *hostname* and port that the MapReduce tracker should run at. Figure 4.6 shows the property definition in XML format.

```
<property>
 <name>mapred.job.tracker</name>
 <value>master:54311</value>
</property>
```

Figure 4.6.: Multiple node Hadoop configuration: mapred-site.xml

Master Node and slave nodes In multiple node cluster, it is important to define which nodes in the cluster are the *slave* nodes that compose the cluster, and which node is the *master* node that is responsible for managing the slaves. Any node on the Hadoop cluster could be the *master* node. On the chosen *master* node, edit the *hadoop/etc/hadoop/slaves* file and add the *hostnames* of all the nodes that compose the Hadoop cluster. All the nodes specifies here will be used to process, store, and write data to the HDFS. The *slaves* file should be edited only on the *master* node, and it could also include the *hostname* of the master node if it is intended to use the *master* node for processing data.

4.4.3. Start Hadoop Cluster

After completing the installation of the prerequisites and configuration of Hadoop cluster, the cluster can be launched to run MapReduce jobs and process data. The first step in starting the Hadoop cluster is to format the file system and to prepare the DataNodes. The format of the HDFS is done using the command *hdfs namenode -format*. At this stage, the Hadoop cluster is ready to be launched. The command *start-dfs.sh && start-yarn.sh* is used to start the Hadoop cluster. This command will start the following instances the *DataNode, NodeManager SecondaryNameNode, NameNode, ResourceManager*.

The *DataNode, NodeManager* will be launched on all the nodes of the cluster while the rest of the instances will run only on the *master* node. On the single node cluster, all these instances will run on the computer that composes the cluster. After a successful start of the Hadoop cluster the end user has the access to the following Hadoop web pages.

1. Cluster status: http://localhost:8088
2. HDFS status: http://localhost:50070
3. Secondary NameNode status: http://localhost:50090

4.5. Summary

Building a Hadoop cluster is not a straightforward task, especially when creating a multiple node cluster. All the prerequisite of Hadoop cluster should be installed and all the configuration steps should be done before starting the Hadoop clusters. These Hadoop clusters are essential for developing and running MapReduce jobs to process data in a parallel approach.

The main goal of this section is to describe the steps needed to build a Hadoop cluster. The first step in developing hadoop clusters is the preparation of the nodes that will compose the Hadoop cluster. The preparation steps include installing prerequisite libraries and run times such as *pydoop* and *JDK*. In addition, it includes the setting up of the environment for the Hadoop cluster. For instance, disabling IPv6 to eliminate conflict errors.

The second step in building a Hadoop cluster is the installation of Hadoop cluster. The installation of hadoop is done by downloading the Hadoop framework from the Apache web site and extracting it to the local file system. Moreover, the installation step will define the **home** directories of the Hadoop cluster and Java in the *.bashrc*.

The next step is the configuration step. This step is responsible for defining the identity of the Hadoop cluster. To build a single node cluster, a modification of a set of XML configuration files is used for saving the configuration parameters. These files can be found in *hadoop/etc/hadoop*.

Building a multiple node cluster is almost the same as the single node. However, because multiple node clusters consist of more than one node, all the nodes in the cluster should be able to reach each other. The *master* node should be able to connect to each of the slave nodes through SSH connection. Furthermore, some of the configuration items are different from the single node configurations. For instance, the number of the replication in the single node cluster must be exactly **1**, while in multiple node clusters it could be any number less than the number of cluster nodes and greater than **1**.

This section also describes the hardware specification of the clusters used in the development and evaluation of the data-parallel transcoding approach. These clusters are the single node cluster, the dual node cluster and the EC2 clusters. The single and dual node clusters are local clusters, while the EC2 clusters are developed using Amazon EC2 web service. The local clusters are used mainly for the development purposes, while the EC2 clusters are used for the evaluation of the system and environmental properties of the implemented MapReduce jobs .

5. Implementation

5.1. Overview

Chapter 3 : Data-Parallel Transcoding Architecture introduces the proposed data-parallel approach, which is suggested in this thesis and that is to transcode large 3D datasets in a parallel paradigm. The chapter highlighted the high level phases of the approach architecture and presented the tasks, objectives and goals of each phase of the architecture. The proposed data-parallel architecture consists of four main phases which are the *Preparsing* phase, the *Partitioning* phase, the *MapReduce* phase, and the *Deployment* phase.

On the other hand, this chapter will focus mainly on the implementation details of each of the phases of the proposed architecture. This chapter will describe the technologies used in each of the phases and difficulties faced in each of the phases. The rest of this chapter is organized as follows.

Section 5.2 of this chapter will begin with a detailed description of Phase I: The *Preparsing* Phase. This section will describe the tools and the technologies used in the development of this phase. Moreover, it will present the results of this phase and illustrate the state of the X3D file after completing this phase.

Then, Section 5.3 of this chapter will present the effects of Phase II: the *Partitioning* Phase. it will highlight the the main tools and scripts used during this phase. After that, in section 5.4, the chapter will explain the implemented MapReduce approaches for transcoding the data and the architecture for each of them.

Next, the *Deployment* phase will be presented in section 5.5. This section will present the tools and methods used for deploying the transcoded 3D model to the web. After that, section 5.6 will provide a complete description of the main components of the implemented transcoding web environment and it will also describe the data-flow and dynamic architecture within the implemented environment. Finally section 5.7 will summarize the implementation details of all the phases.

5.2. Phase I: PreParsing

As stated in Chapter 3, the main purpose of the *Preparsing* Phase is to prepare the 3D model for the further processing in the next phases of the architecture. The main objectives of this phase are to remove the unwanted nodes, replace all the nodes with **USE** attributes with the original X3D definition, and build the product breakdown tree.

The *preparsing* phase consists of two steps. The first step is a very essential step, and it aims to prepare an X3D representation from the 3D model original file. The proposed data-parallel approach assumes that there is an X3D representation for the 3D model, however, in reality, 3D is not developed using X3D editors, instead the 3D models are created using CAD applications. Therefore, the need for a conversion tool that converts 3D models from original 3D model files to X3D is raised.

The conversion of 3D models to X3D format is out of the scope of this thesis and therefore, we decided to rely on an external tool for performing this job. There are tens of software applications on the market that help engineers to convert 3D models from various 3D file formats such as *CATProduct* and *JT* files to X3D file format. During this thesis we used a commercial tool called **PloyTrans**.

PloyTrans is a software application that allows engineers to import 3D models from various 3D file formats, to perform some optimization on these 3D models such as the lighting of the scene, choose how the geometries are represented in the X3D file, and export the 3D model to various 3D model formats. One of the exported file formats is the X3D file format.

The second step in the *Prepasing* phase focuses on achieving the objectives of this phase. All of the three goals mentioned above require the parsing and processing of the X3D files. For instance, in order to achieve the first goal, which is eliminating the unwanted X3D nodes, the developer should write a script or application that reads the X3D file, search for the unwanted nodes then delete these nodes, and finally save the X3D file without these nodes.

There is more than one option to achieve the mentioned goals. One option, is to develop an application that treats the X3 file as text file. This application can use regular expressions to locate the parts of the X3 file that need to be changed or it can iterate over the lines of the X3D file to find them. This application can be developed by a wide range of programming languages such as Java, Python, and Ruby.

The main advantages of this type of application is that the developers have a control over the processed file and that these applications can be easily integrated within web applications or other programming languages, simply by providing APIs. However, developing such an application will result in a very low performance application. X3D files are large files and parsing them line-by-line or using regular expirations is not the most suitable method. Usually this kind of applications will try to load the file in the memory and then perform the processing on it. For this reason, processing X3D files with this approach will take a long time to be completed.

X3D files are files that follow the XML standards; therefore, X3D files are also XML files. Another option for achieving the goals of this phase is to use a XML query language to perform the tasks of this phase. The main benefits of using a XML query language is that it is much easier to write scripts that perform the same tasks using these query languages. Another reason is that these query languages are much faster in processing XML files.

XQuery is a functional programming language that allows the developer to select, delete, and replace XML nodes. Moreover, it allows developers to create XML files. During this thesis, we relied on a XML database engine called BaseX. BaseX is also an implementation for the XQuery programming language. More details about XQuery and BaseX can be found in Chapter 2.

The implementation of the *Preparsing* phase involves the development of a set of XQuery scripts that is used to achieve the phase objectives. These scripts work as a pipeline. Each of these scripts processes an X3D file and generates an X3D output file. The output of the last script is the X3D file that is ready for further processing in the *Partitioning* phase. The implemented scripts can be classified in three categories: remove unwanted nodes script, replace **USE** script, and create product breakdown script. Below is a description of the implemented XQuery scripts and their main functionality.

5.2.1. Remove Unwanted Nodes Script

XQuery language allows developers to write scripts that are able to manipulate XML nodes from a XML document. To remove a node from a document, the developer needs to use the keywords **delete node** and then add a reference to the specified node. Using these keywords every single node in the XML document can be deleted.

The script responsible for removing the unwanted nodes is called *cleanModel.xq*. This script is simply a bunch of **FLWOR** expressions that iterate over all the nodes that belong to the same category and delete them all from the X3D file. For instance, the script includes **FLWOR** expressions to delete each of the following nodes: *meta*, *head*, and *component*.

Figure 5.1 shows a simple **FLWOR** exposition that aims to remove all the *meta* nodes. The shown script iterates over all the nodes in the X3D file with the name - *meta* - and then deletes these nodes from the X3D document. After executing all the **FLWOR** expressions in the script, a new X3D file will be available, this file does not contain any of the unwanted nodes.

```
for $i in //*
where fn:contains($i/name(),"meta")
return
delete node $i
```

Figure 5.1.: Remove meta nodes using XQuery

5.2.2. Replace USE Script

Replacing nodes that depend on the **USE** attribute for defining their values is more complex than removing a node from the X3D document. The task of replacing these nodes is done in two parts. The first part is to copy the original definition of X3D node and replace the node that uses the **USE** attribute. This step will introduce redundant copies of the same X3D nodes all over the X3D document.

All the occurrences of these nodes are identical and therefore, they all contain a **DFF** attribute with the same value. As a result, the redundant **DFF** values will confuse the X3DOM engine in rendering the 3D model since there are more than one node with the same **DFF** value. Therefore, the next part of the of replacing the **USE** script is to remove the **DEF** attribute and its value from all the redundant occurrences.

As stated in Chapter 3 , any X3D node can be reused using the **DFF** and **USE** attributes. However, the chapter also specified that there are some common reused X3D nodes such as the *Material* nodes, the *Appearance* nodes and the *Group* nodes. The replace USE scripts developed in this thesis only replace a set of the X3D nodes and not all of them. For instance, the material and group replacement functionality is implemented in the script, but the shape replacement is not implemented. We decided to implement only a subset of the X3D nodes, because most of the models that we worked with include only reuse for this subset. The implemented subset includes only the *Material* nodes, the *Appearance* nodes and the *Group* nodes.

```
for $matDef in //Material[@DEF]
where (string-length(fn:data($matDef/@DEF))>0)
return
(: now select every usage of current material def :)
for $matUse in //Material[@USE=fn:data($matDef/@DEF)]
return
replace node $matUse with $matDef
```

Figure 5.2.: Remove material reuse from X3D document

Figure 5.2 presents the XQuery script used to replace all the occurrences of *Material* nodes that depend on the **USE** attributes with the original definition. The script begins by iterating over all the *Material* nodes that include **DFF** attributes. Then, for each of these nodes, the script iterates over all the *Material* nodes that include **USE** attribute, and the value of their **USE** attribute is the same as the **DFF** attribute value of the main node.

5.2.3. Create Product Breakdown Tree Script

One of the main goals of the *Preparsing* phase is to create the product breakdown tree. This tree is simply a HTML list that describes the hierarchical structure of the X3D document. The items of this list represent the physical parts of the modeled object. For example, in an airplane model, the right wing and left wing are considered to be physical parts and, therefore, they will be items on the product breakdown tree.

The create product breakdown tree script is called *DnyTree.xq*. The script main purpose is to create a HTML file that contains the product breakdown tree. Usually the product parts information is included in the **DFF** attributes of the *Transform* nodes. However, not all *Transform* nodes include a **DFF** attribute and not all of them represent physical parts. The script begins by processing the first *Transform* node within the X3D scene. This node is the root node for all other *Transform* nodes in the X3D documents.

Then, the script explores all the children of the root *Transform* node. For all *Transform* nodes that include **DFF** attribute, it will create an item in the product breakdown tree. In addition, for all the *Transform* nodes that include children, the script tries to explore their children and produce a sub tree that represents the structure of the children nodes.

As a side effect of this behavior, the script generates an empty sub tree for *Transform* nodes that does not include children that contain **DFF** attribute. For this reason, after completing the generation of the product breakdown tree, the script iterates over the tree nodes and removes all the empty items or sub trees. The last task of this script is to save the product breakdown tree into a HTML file on the file system for later use.

Figure 5.3 shows an X3D document before and after performing the steps of the *preparsing* phase. From the presented X3D document, it is clear that all the occurrences of the *Material* nodes that have **USE** attribute are replaced with the original definition of the node. In addition , all *meta* nodes have been removed from the document.

5.3. Phase II: Partition X3D Files

Once the *Preparsing* phase is completed and the X3D document is ready for further processing, the *Partitioning* phase can be entered and performed. The primary goal of the *Partitioning* phase is to divide or split the X3D document generated from the *Preparsing* phase into several chunks of X3D documents. Each of these chunks should be independent from all other chunks and can be processed or transcoded successfully without affecting the rest of the chunks.

```
<!-- X3D document before perfroming the preparsing phase tasks !-->
<X3D>
 <head>
  <meta name="filename" content="inter-rotation-2.x3d"/>
  <meta content='primitives-list.x3d' name='title'/>
  <meta content='Simple DEF/USE/Transform Demo' name='description'/>
 </head>
 <Scene>
  <Transform translation='-4 0 0'>
   <Shape>
    <Box size='1.8 1.8 1.8'/>
    <Appearance>
     <Material DFF="COLOR1" diffuseColor='1 1 0'/>
    </Appearance>
   </Shape>
  <Transform>
  <Transform translation='-4 0 0'>
   <Shape>
    <Cone bottomRadius='0.9' height='1.8'/>
    <Appearance>
     <Material USE="COLOR1"/>
    </Appearance>
   </Shape>
  </Scene>
</X3D>
<!-- X3D document after perfroming the preparsing phase tasks !-->
<X3D>
 <Scene>
  <Transform translation='-4 0 0'>
   <Shape>
    <Box size='1.8 1.8 1.8'/>
    <Appearance>
     <Material diffuseColor='1 1 0'/>
    </Appearance>
   </Shape>
  <Transform>
  <Transform translation='-4 0 0'>
   <Shape>
    <Cone bottomRadius='0.9' height='1.8'/>
    <Appearance>
     <Material diffuseColor='1 1 0'/>
    </Appearance>
   </Shape>
  <Transform>
</Scene>
</X3D>
```

Figure 5.3.: X3D document before and after *Preparsing* phase

The *Partitioning* phase consists of three major tasks. These three tasks ensure that the input X3D document is divided in a way that allows the parallel data processing of the generated X3D chunks. These tasks are to extract all the shapes from the X3D document, extract all the parts in the X3D document, and extract the main part of the 3D model.

Similar to the *Preparsing* phase, the *Partitioning* phase also needs to operate on the X3D document, locate a specific type of nodes, extract these nodes in separate files, and replace these nodes with links to the corresponding generated X3D chunks. For the same reasons stated in the previous section, mainly the ease of development and the high performance of XQuery scripts, we decided to use BaseX and XQuery for implementing the tasks of the *partitioning* phase.

The development of the *Partitioning* phase tasks involved the implementation of a set of XQuery scripts. Each objective of this phase has its own script for creating the results. the implemented scripts in this phase are the *Extract all shapes* script, The *Extract all parts* script, and the *Extract main part* script.

The responsibility of these scripts is to ensure the achievement of the phase objectives and goals. There is a specific order to execute these scripts in order to generate a complete independent dataset. For instance, the first executed script is the extract of all shapes script, then to extract all parts script and finally to extract main part script. Each of these scripts modify the X3D document and prepare it for further processing by the next script. Below is a description of the implemented XQuery scripts and their main functionalities.

5.3.1. Extract All Shapes

The first task in the *Partitioning* phase is to **Extract All Shapes**. This task aims to create a file hierarchy that represents all the shapes contained in the 3D model. Figure 3.2 shows the expected file structure. This task will create a directory for each shape that exists in the X3D document, and inside this folder it will create a X3D file that contains the definition of the shape.

As stated above, this task is implemented in XQuery language using the BaseX engine. The implementation result is a simple XQuery script that iterates over all the geometric shapes in the X3D document and extracts them. Figure 5.4 shows the extract shapes algorithm. The implemented scripts follow the steps of this algorithm in order to complete the task successfully. For each of the shapes in the X3D document the script performs a set of steps.

First, it figures out the shape identifier. This identifier is the **DFF** attribute value of the shape itself or the **DFF** attribute value of the parent node, if the shape does not have one. The next step is to create a directory that will be used to contain the shape data. The name of this directory is the shape identifier. After that, the script creates a subdirectory called *binGeo* in the shape directory. The *binGeo* directory will by used during the *MapReduce* phase to contain all the binary files that represent the shape.

Then, the script will integrate the shape within a basic X3D template to be able to view and render the shape independently. Simply, the template is just warping the *shape* node with a *X3D* and *Scene* nodes. After preparing the X3D file from the template, the script saves the file to the file system in the shape directory. The last step of the algorithm, is to replace the shape node in the original X3D document with an *inline* node. The *inline* node will point to the X3D file that was created in the previous step.

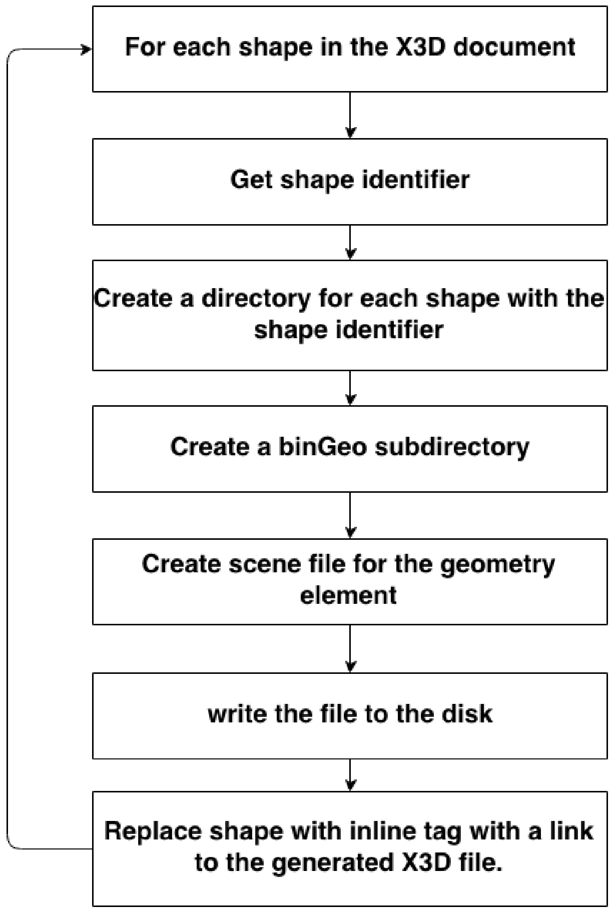
For each shape in the X3D document
Get shape identifier
Create a directory for each shape with the shape identifier
Create a binGeo subdirectory
Create scene file for the geometry element
write the file to the disk
Replace shape with inline tag with a link to the generated X3D file.

Figure 5.4.: Extract all shapes algorithm

5.3.2. Extract All Parts

The **Extract All Shapes** is performed before any task in the *Partitioning* phase for one reason. Parts represent the physical parts of the 3D model and not the geometries of these parts. Therefore, before extracting the parts, the geometries' representation should be removed from the X3D file first. This step is done using the **Extract All Shapes** script.

The **Extract All Parts** is also an XQuery script. This script aims to extract all the *Transform* nodes that represent the physical parts of the 3D model. These *Transform* nodes represent either a high level part such as the left wing, a sub part of the 3D model like the engine of the right wing, or a single geometry. All the *Transform* nodes that represent physical parts contain a **DEF** attribute. The value of this attribute is a simple description or name for the physical part that the node represents.

Figure 5.5 shows the steps of the **Extract All Parts** algorithm. The algorithm begins by preparing a set of all the *Transform* nodes that represent physical parts. In other words, only *Transform* nodes that have a **DFF** attribute will be included in the list. This list will be ordered in descending order based on the depth of each *Transform* node. The next step is to iterate over all the nodes in the prepared list and performing a set of tasks on each of these nodes. The first step is to create a directory that represents the physical part with the same name as the **DFF** attribute value. Then, extract the *Transform* and all its children and integrate it with a X3D template that is similar to the shapes template. The last step is to save the generated file to the file system under the *Transform* directory.

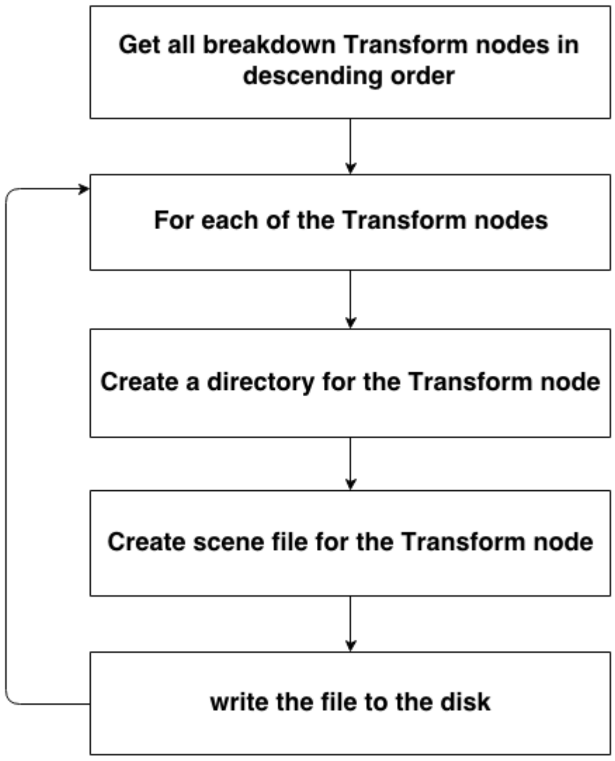
Get all breakdown Transform nodes in descending order
For each of the Transform nodes
Create a directory for the Transform node
Create scene file for the Transform node
write the file to the disk

Figure 5.5.: Extract all parts algorithm

5.3.3. Extract Main Part

The **Extract All Parts** will generate X3D files that represent the parts of the 3D model. These files can be used to render the individual parts of the 3D model in a stand alone mode. However, it is difficult to find out which file of these files represents the complete model based on the directory names. For this reason, the **Extract Main Part** script is implemented.

The **Extract Main Part** aims to extract the root *Transform* and save it to the file system with a predefined name. This way, it will be easier to find the part that represents the complete 3D model and it will also be feasible to reference this part from other documents. The algorithm used to extract the main part is almost the same as the algorithm used to extract all the parts.

The difference between these two algorithms is that the **Extract All Parts** algorithm prepares a set of *Transform* nodes in the first step, while the **Extract Main Part** algorithm only selects the first *Transform* node in the X3D document. All the *Transform* nodes that exist in the X3D document are children of this node. Therefore the first *Transform* node represents the root element of all the parts. The remaining steps in both algorithms are identical.

5.4. Phase III: MapReduce Jobs

After the *Partitioning* phase finishes successfully, the original dataset will be irrelevant for further processing. The next phase will operate on the generated files in the *Partitioning* phase. These generated files include the geometers of the 3D model represented by the **shapes** and the physical parts of the 3D model represented by the **parts**. The main goal of the *MapReduce* phase is to compress the geometries and to generate binary files for all the geometries that compose the 3D model.

As stated in Chapter 3, two major tasks should be done to achieve the goals of this phase. The first task is to build a hadoop cluster to be able to execute and evaluate the implemented MapReduce jobs. During this thesis, two types of clusters were built: the **local clusters** for the purpose of development of the MapReduce jobs, and the **EC2 clusters** for the evaluation of the MapReduce jobs on Amazon elastic compute cloud web service. More details regarding the hardware specification, installation steps, and configuration of these clusters can be found in Chapter 4.

The second task is the implementation of the MapReduce jobs that are responsible for transcoding the 3D data to binaries. MapReduce jobs can be developed using Java programming language. The developer creates a JAR file for the implemented MapReduce Job and executes it with the *hadoop jar* command directly. All codes that are needed for performing the MapReduce job is included in the JAR file. For example, the definition of the **Map** and **Reduce** functions will be included in the JAR file either as separate classes or as functions that belong to the same class.

The benefit of developing MapReduce jobs with Java programming language is that the developer will have a complete access to the Hadoop and MapReduce APIs provided by Hadoop framework. Moreover, Java is the programming language that is used in Hadoop development and therefore, there is a huge community that supports the development of MapReduce Jobs in Java. As a result, it will be easier and more comfortable to implement these MapReduce jobs using Java programming language.

In addition, MapReduce jobs can be developed in a wide range of programming languages such as Python, Ruby and Perl using a technology called Hadoop streaming. To implement a MapReduce job with this technology, the developer should implement the **Map** and **Reduce** in separate programs or files. Each of these functions should read the data from the standard input and write its output to the standard output. When the developer wants to launch the MapReduce job, he can use the same command used to launch Java MapReduce job. However, the developer needs to use the streaming JAR provided by Hadoop and he must specify the file names of the **Map** and **Reduce** functions in order to be able to execute the MapReduce Job on a Hadoop cluster.

Figure 5.6 shows a sample of the commands used to launch Hadoop streaming and Hadoop JAR MapReduce jobs. For the Java MapReduce jobs, the developer only has to provide the JAR file name and the name of the **main** class. While for the Hadoop streaming jobs, the developer needs to specify more information such as the input and output folders. Another advantage of the Java MapReduce jobs is that the developers can develop their JAR files to be configurable. For instance, a developer can create a MapReduce job that takes the input and output folders as arguments.

```
// Lunching a MapReduce Job Developed in Java
hadoop jar MyMapReduce.jar transcodeMe

// Lunching a Configrable MapReduce Job Developed in Java
hadoop jar MyMapReduce.jar transcodeMe /inputData /outputData

// Launching a Hadoop Streaming MapReduce job developed in Paython
hadoop jar hadoop-streaming.jar -file mapper.py  -mapper mapper.py
                                -file reducer.py -reducer reducer.py
                                -input /data/* -output /out-put
```

Figure 5.6.: Hadoop commands to launch MapReduce jobs

During this thesis, three MapReduce jobs were developed, one of them using the Hadoop streaming technology and the other jobs were developed using the custom JAR technology. These jobs are the **Hadoop Streaming** job, the **Hadoop Custom JAR**, and **HDFS Hadoop Custom JAR**. The reason behind developing three different MapReduce jobs that perform the same task is to be able to evaluate whether the technology, programming language or the architecture of the MapReduce job have an influence on the performance of the MapReduce job.

The rest of this section will focus on describing the details of each of the implemented MapReduce jobs. It will highlight the technologies, programming languages and the architecture of each of these Jobs.

5.4.1. Hadoop Streaming MapReduce Job

The first MapReduce job implemented in this thesis was the **Hadoop Streaming MapReduce** job. This MapReduce Job is implemented using Python programming language and it uses the Pydoop library for running commands on the HDFS file system. Pydoop[1] is a python library that provides the developers with an API for MapReduce Hadoop framework [37].

The main advantage of using Pydoop API in the development of MapReduce jobs is that it allows the developers to access all Hadoop standard library and third party modules using Python programming language [37]. Some of these modules and libraries can not be accessed using pure Python programs or some other Python libraries such as **SciPy** library.

In addition, Pydoop provides the user with an API for the HDFS. This API allows developers to run commands on the HDFS file system. For instance, using this API, the developers can connect to the Hadoop cluster file system and perform tasks such as read and write files as well as get information about these files.

The **Hadoop Streaming MapReduce Job** implementation consists of three python scripts: the **mapper** script, the **reducer** script, and the **driver** script. Each one of these scripts has its own goals and plays a major role in defining and launching the MapReduce Job. Below is a detailed description of each of these scripts and the tasks that these scripts perform during the MapReduce job.

The Driver Script goals are to prepare the HDFS file system for the MapReduce job and to launch the MapReduce job with the correct parameters. Three major tasks are performed by the **Driver** script. The first task is to connect to the web server where the dataset is stored, download a list of URL links to all the resources included in this dataset, and split this list into small chunks of files. Each of the chunks contain only **3** links. This value is called the *split size* value. The split size controls the number of the generated **Map** functions used in the transcoding process. The second task is to copy the generated chunks to the HDFS file system. These files will be the input dataset for the the MapReduce job. The last task is to launch the MapReduce job using the Hadoop streaming command and specify all the arguments required by the command.

The Mapper Script is the script that is executed by every **Map** function instance. The main purpose of this script is to transcode the X3D files to binary files and store them into the HDFS file system. Each of the generated **Map** functions will operate on a set of the chunks created using the **Driver** script. The number of the chunks that a single **Map** function can transcode is defined by the *split size*. The **Map** function instance will read the content of these chunks from the standard input stream.

[1]http://pydoop.sourceforge.net/docs/ Retrieved on 2014-10-24

Each of the **Map** instances begins by iterating over each line in the standard input stream and performs the following tasks on them. First, download the web resource from the web server to a temporary folder on the local machine. Next, use the *aopt* tool to transcode the X3D file to a binary file. Then, copy the results of the transcoding process back to the HDFS file system.

The **Map** instances will reconstruct the file hierarchy of the dataset based on the URLs of the X3D chunks. Finally, in the case of the transcoding process completed successfully, the script will write a successful message to the standard output stream, otherwise it will write an error message. After all the **Map** function instances finish their tasks, the results of transcoding the whole 3D model can be found on the HDFS file system on a file hierarchy similar to the hierarchy of the input dataset.

The Reducer Script is executed by each **Reduce** function instance. The goal of **Reduce** functions in MapReduce programming model is to sort, combine, and prepare the results of the MapReduce job. However, in our case, since the transcoded data is generated by the **Map** instances and there is no need to perform sorting or combining tasks on the dataset, we decided to use the **Reduce** functions for another purpose. This purpose is the validation of the transcoding process done by the **Map** instances. Each of the **Map** function instances will write to the standard output stream whether the transcoding process was successful or not. The **Reduce** function instances will receive these messages and it will count the successful transcoding process and the failed ones.

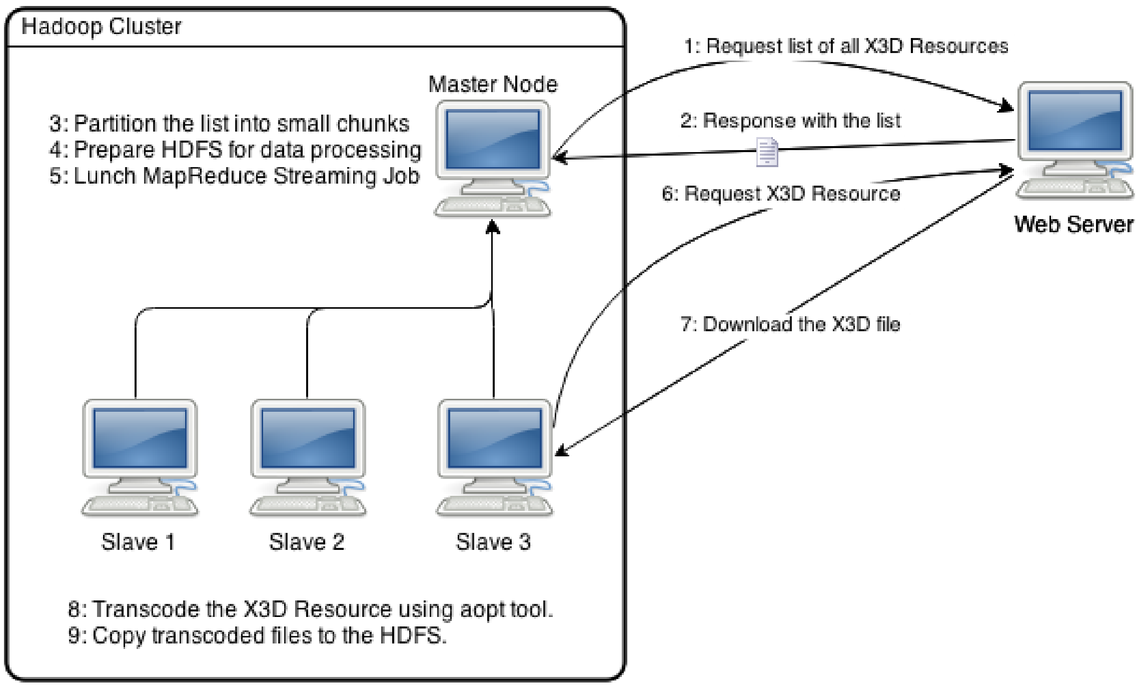
Hadoop Cluster
Master Node
3: Partition the list into small chunks
4: Prepare HDFS for data processing
5: Lunch MapReduce Streaming Job
1: Request list of all X3D Resources
2: Response with the list
6: Request X3D Resource
7: Download the X3D file
Web Server
Slave 1
Slave 2
Slave 3
8: Transcode the X3D Resource using aopt tool.
9: Copy transcoded files to the HDFS.

Figure 5.7.: Hadoop streaming job communication diagram

Figure 5.8 is a communication diagram that illustrates the components of the **Hadoop Streaming MapReduce Job** and the communications between these components while executing a MapReduce job. As shown in the figure, two high level components are composing the architecture of **Hadoop Streaming MapReduce Job**, which are the Hadoop cluster and the web server that host the partitioned dataset. The Hadoop cluster could be a single node cluster or multiple node cluster. The MapReduce job is executed and launched from the *master* node of the Hadoop cluster.

When the end user runs the *Driver* script, the following tasks are performed on the web server and Hadoop cluster. First of all, the master node of the hadoop cluster asks the web server for a list of URL links to all X3D resources contained on the partitioned dataset resulted from the *partitioning* phase. Next, the web server will respond with the requested list and send it to the *master* node.

After the *master* node completes downloading the requested list, the *master* node partitions the resources list into small chunks of files. These files will be used as an input dataset of the MapReduce job. Then, the *master* node prepares the HDFS file system and ensures that it is ready for launching the MapReduce job. For example, The *master* node checks if there is any naming conflicts between the new dataset and the data on the HDFS file system. The next step is to launch the MapReduce job. All the tasks described above are performed from the *Driver* script.

As soon as the MapReduce job is started, all the **DataNodes** of the Hadoop cluster will start executing the **Map** and **Reduce** functions. Each of the **Map** instances will request X3D resources from the web server, download them to the local file system, transcode these resources to binary files, and finally copy back the transcoding results to the HDFS file system.

The last step is important because Hadoop allows the local data processing only on temporary files; which means all the local files will be deleted as soon as the MapReduce job is completed. Another reason for copying back the transcoding data to the HDFS is that the data on the HDFS will be accessed as it was stored on one computer, even if it is distributed over multiple nodes. This way, it is easier to move the transcoded data from the HDFS to the deployment server in the next phase.

5.4.2. Hadoop Custom JAR MapReduce Job

The **Hadoop Custom JAR** job is the second implemented MapReduce job during this thesis. This job is implemented using the Java programming language and Hadoop custom JAR method. The components and scripts of the **Hadoop Custom JAR** job is very similar to those of the **Hadoop Streaming** MapReduce job. The only major difference between these two jobs is the technology and programming language used in the development of each of them.

As the **Hadoop Streaming** MapReduce job, the **Hadoop Custom JAR** job consists of three Java classes. The first class is the *JobDriver* class, which is responsible for preparing the HDFS, configuring the MapReduce job, and starting it. The second class is the *GeoMap* class which defines the behavior of **Map** instances. The third class is the *GeoReduce* class, this class defines the behavior of Reduce instances. The *JobDriver*, *GeoMap*, and *GeoReduce* provide an identical functionality to the functionality provided by the *Driver*, The *Mapper*, and the *Reducer* defined in the **Hadoop Streaming** MapReduce job.

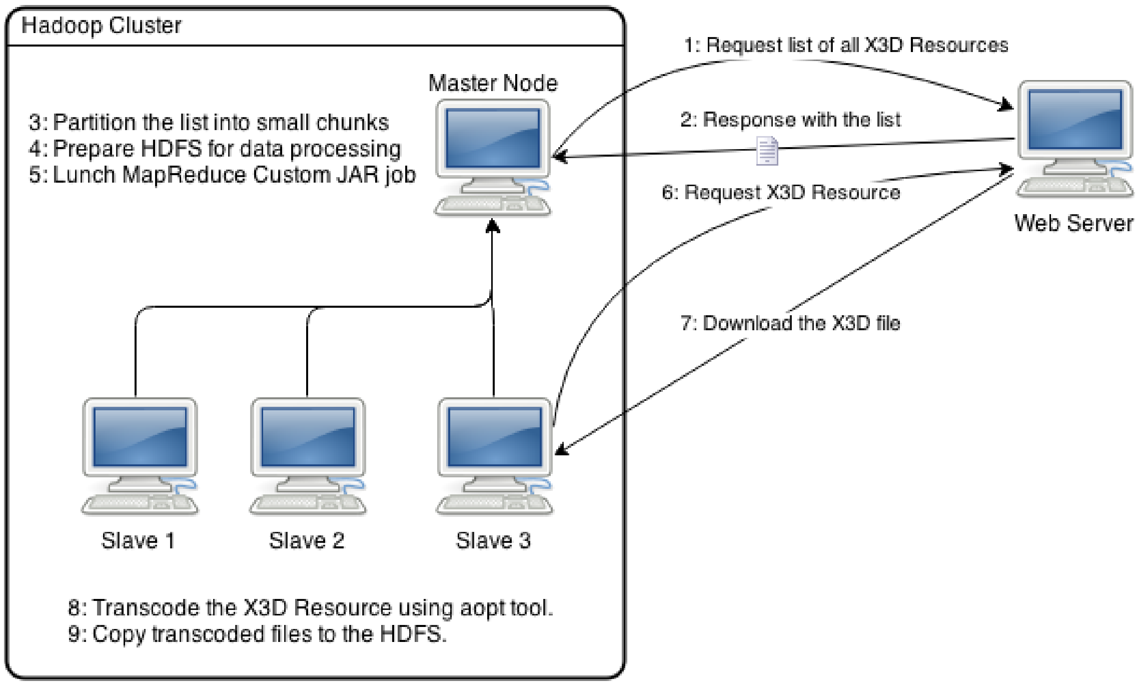
Hadoop Cluster
Master Node
3: Partition the list into small chunks
4: Prepare HDFS for data processing
5: Lunch MapReduce Custom JAR job
1: Request list of all X3D Resources
2: Response with the list
6: Request X3D Resource
Web Server
7: Download the X3D file
Slave 1
Slave 2
Slave 3
8: Transcode the X3D Resource using aopt tool.
9: Copy transcoded files to the HDFS.

Figure 5.8.: Hadoop custom JAR job communication diagram

Figure 5.8 is a communication diagram that describes the interaction and communications between the components of the **Hadoop Custom JAR** job. From the diagram, we can see that all the interactions of this job are identical to those interactions in the **Hadoop Streaming** MapReduce job. In addition, both of the jobs are composed of the same main components, which are the web server and the Hadoop cluster.

The only difference is that this MapReduce job is developed using Java programming language, while the other is developed using Python programming language and Pydoop API. All other steps of executing the MapReduce jobs are the same in both Jobs. The *master* node prepare the HDFS file system, prepare the input dataset and start the MapReduce job. The *DataNodes* of Hadoop cluster request X3D resources to transcode them to binaries and the web server responds to the requests from the *master* node and *slave* nodes.

The exact same MapReduce job was developed twice, by using two different technologies and programming languages to be able to evaluate the technologies used in the implementation of MapReduce jobs. This way, it is feasible to evaluate both the Hadoop streaming and Java Custom JAR technologies and find out which one gives us the highest performance in transcoding 3D data. This evaluation step will help us in developing a more efficient and effective MapReduce job by selecting the technologies which realize the lowest transcoding time.

5.4.3. HDFS Hadoop Custom JAR Job

The third MapReduce job is the **HDFS Hadoop Custom JAR** job. This job is implemented with the same technology used to implement the previous MapReduce job, which are the Hadoop custom JAR and Java programming language. This job performs almost all the tasks that the previous two jobs perform. However, the architecture and components interactions of this job are more different than the other jobs.

The **HDFS Hadoop Custom JAR** is composed of four components, the **Map** function which is defined in the *GeoMap* class, the **Reduce** function which is defined in the *GeoReduce* class, the **JobDriver** application which is defied in the *JobDriver* class; this class is responsible for configuring the MapRduce job and submitting it to the Hadoop cluster. The last component is the **HDFSTools**. This component includes utility functions that support the writing,reading and deleting files on the HDFS file system.

Figure 5.9 illustrates the components and the interactions of the **HDFS Hadoop Custom JAR** job. In the previous two jobs, the partitioned dataset were placed in an external web server. One of the tasks for these jobs is to connect to the web server to request the resources list and the individual resource to transcode them. However, it is clear that this job is only composed of one high level component which is the Hadoop cluster and it does not include any external web server.

The partitioned dataset is placed on the HDFS on the **HDFS Hadoop Custom JAR** job. This difference adds one more task for the MapReduce job, which is populating a list of all the X3D resources that is included on the dataset. As shown in Figure 5.9, when the end user launches a MapReduce job of this type, the job's first task is to connect the HDFS file system where the input dataset is stored. The next step is to iterate over all the subdirectories of the input directory to generate a list of paths for all the X3D resources included in the dataset. Then, the job will partition this list into small chunks. After that, the HDFS file system will be prepared for the data processing. This step includes creating the output folders and removing old files and directories that have any conflict with the new dataset.

By the time the HDFS the HDFS is ready, the MapReduce job will be configured and started processing the dataset. All the active *DataNodes* on the Hadoop cluster will participate in the transcoding process. Each one of these nodes will receive a bunch of input chunks created in the previous steps. These nodes will create the **Map** and the **Reduce** instances to transcode these resources and copy them back to the HDFS file system.

The reason behind removing the web server component from this MapReduce job is because requesting and downloading a huge amount of datasets from a web server. This will lead to an increase of the time required to transcode the dataset. Therefore, we were interested in finding the effects of eliminating this communication overhead which is caused by the huge number of HTTP GET requests from an external web server.

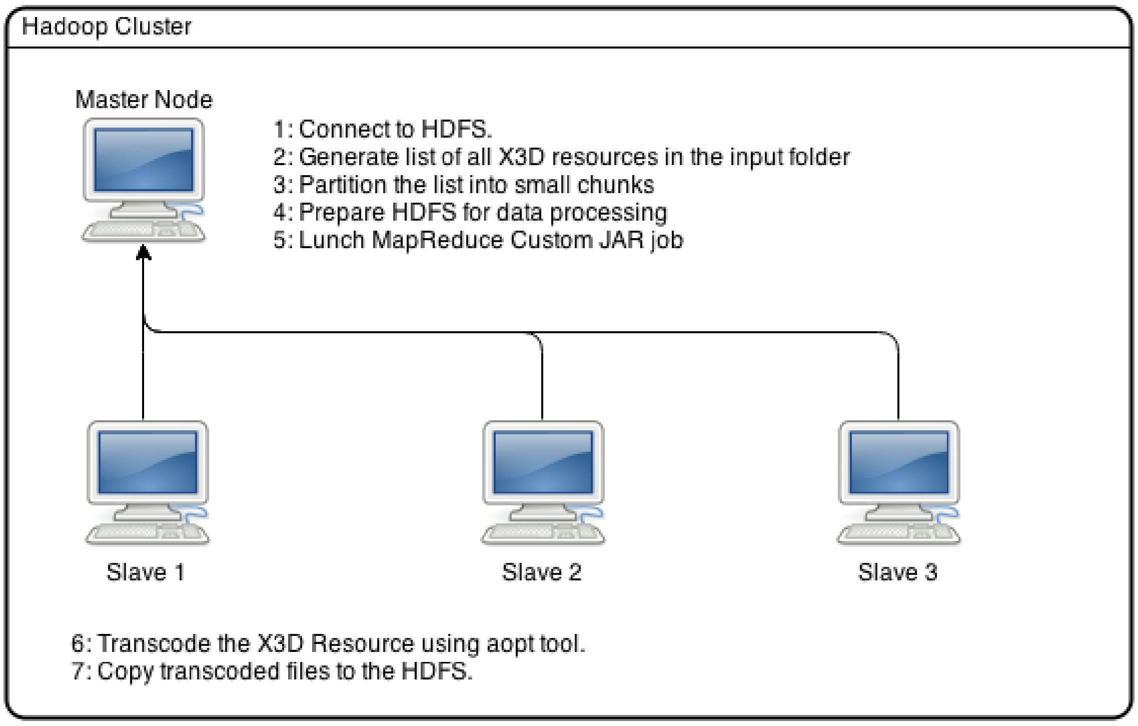
Hadoop Cluster
Master Node
1: Connect to HDFS.
2: Generate list of all X3D resources in the input folder
3: Partition the list into small chunks
4: Prepare HDFS for data processing
5: Lunch MapReduce Custom JAR job
Slave 1
Slave 2
Slave 3
6: Transcode the X3D Resource using aopt tool.
7: Copy transcoded files to the HDFS.

Figure 5.9.: HDFS Hadoop custom JAR job communication diagram

5.5. Phase IV: Deployment

The result of all three MapReduce jobs described in the previous section is a file hierarchy that contains all the transcoded X3D files. These resources are located on the HDFS file system. As a result, the end user can view the content of these files as text view, but he does not have a tool to view the transcoded resources in a web browser with a 3D viewer application. For this reason, the *Deployment* phases' main goals are to deploy the transcoded resources to a web server and to create a 3D viewer HTML page that allows the users to view and interact with the 3D model in a web browser.

The first step in achieving the goals of this phase is to build a web server that will host the generated HTML pages. For this purpose, we installed **Apache Tomcat** web server. We chose this server over the available web servers, because *Tomcat* supports Java server side scripting and in our case we need to perform some of the tasks on server side such as running the XQuery scripts. Once the web server is ready, deploying a transcoded 3D model consists of a set of tasks described below.

```
<!-- Link X3DOM framwork !-->
<script type='text/javascript' src='js/x3dom.js'></script>
<link rel='stylesheet' type='text/css' href='css/x3dom.css'></link>
<!-- Link JQuery library !-->
<script src='js/jquery-ui.custom.js' type='text/javascript'></script>
<link rel='stylesheet' href='css/jquery-ui.css'>
<!-- Link dynatree library !-->
<script src='js/jquery.dynatree.js' type='text/javascript'></script>
<link href='css/ui.dynatree.css' rel='stylesheet' type='text/css'>
```

Figure 5.10.: Integrating X3DOM, JQuery and DynaTree into the 3D viewer

1. Create a home directory for the transcoding model on the Tomcat web server. This directory will contain all the 3D model data that is required for rendering the model. The model data includes the parts directories resulted from the *Partitioning* phase, the transcoded shapes generated in the *MapReduce* phase, X3DOM framework, JavaScript libraries, CSS files, and a HTML page that provides 3D viewing functionality.

2. Copy the parts main directory to the home directory. The parts directory is located at the *master* node because the transcoding process is executed in this node.

3. Copy the transcoded shapes to the home directory of the web page. These shapes exist on the HDFS file system and not on any of the nodes that compose the cluster. The copy process can not be performed with the *cp* command, instead it is performed using the HDFS command : *hdfs dfs -copyToLocal /SourcePath /HomeDirectoryPath*.

4. Copy JavaScript and CSS libraries to the home directory. These files include the X3DOM JavaScript library, JQuery, DynaTree library, and CSS files. It also includes a user defined JavaScript library that provides utility functions to control the 3D model like hiding and viewing the individual parts of the 3D model.

5. Create the 3D viewer HTML web page. The first step in creating the 3D viewer is to integrate X3DOM with the HTML page, this goal can be achieved only by adding the required X3DOM files to the HTML page. Moreover, JQuery and DynaTree which is used to display the product breakdown tree should also integrate in the HTML document. Figure 5.10 shows the part of the HTML page which is responsible for integrating these libraries.

The second step in creating the 3D viewer web page is to integrate the product breakdown tree generated in the *Preparsing* phase into the HTML page. The last step is to configure the HTML page to view the main part of the 3D model. Figure 5.11 shows the X3D content of the HTML page. The X3D tags mainly defines two objects. The first one is the cursor used in the 3D viewer to point and select the parts of the 3D model. The second

```
<X\textsubscript{3}D id='x3dElement' >
 <scene>
  <Group>
   <Transform id='cursor' translation='0 0 0'>
    <Transform scale='50 200 50' translation='0 -200 0'>
     <Shape isPickable='false' DEF='float'>
      <Appearance>
       <Material diffuseColor='olivedrab' specularColor='peachpuff'></Material>
      </Appearance>
      <Cone></Cone>
     </Shape>
    </Transform>
   </Transform>
   <Group onmousemove='updateCursor(event);' >
    <inline id='viewer' url='' nameSpaceName ="gem" />
   </Group>
  </Group>
 </scene>
</x3d>
```

Figure 5.11.: The X3D content of the 3D viewer web page

object is the 3D viewer, this object is defined as *inline* tag. The value of the *href* attribute of this tag is defined by using a JavaScript function. This way, the user will be able to change the parts that he wants to view on the fly by only changing the *href* value. The default value for the *href* is the X3D file path of the main part.

In order to make the deployment phase easier, especially the creating of the 3D viewer, we implemented a HTML template file that contains the library integration part, the X3D content and a place holder for product breakdown tree. This template enables us to create the 3D viewer web pages automatically by only replacing the place holder of the product breakdown tree.

5.6. Transcoding Environment

Each of the previous phases is independent from each other and each of them is performed manually using the tools and applications used in the development process. For instance, the *Preparsing* and *Partitioning* phases are executed from the BaseX XML database engine, while the *MapReduce* and *Deployment* phases are executed from the Ubuntu terminal using Linux and Hadoop commands. At this stage transcoding a 3D model is done through a set of manual steps for performing each of the phases.

The main goal of the transcoding environment is to provide a web platform that allows the end user to perform the transcoding process in the most efficient way. For example, the implemented transcoding environment should provide an automated way for performing the phases of the data-parallel transcoding approach. In addition, the web application should provide the end user of a set of features to enhance the usability of the application and ensure the easiness of the transcoding process. A list of the most important features and functional requirements of the web environment are described in the list below.

1. Allows the end user to transcode their 3D models automatically. End user should not worry about starting Hadoop cluster or installing BaseX for running XQuery scripts.
2. The end user can create a new customized transcoding project. The transcoding projects can be configured, for instance, the user can specify the commands that are responsible for transcoding the shapes to binary files.
3. Users can list all their transcoding projects and they can delete any of these projects.
4. End user is notified when the transcoding process is completed.
5. The result of the transcoding process is a web page called 3D viewer. This web page allows the user to view the 3D model.
6. The 3D viewer provides the user with the basic functionality for viewing 3D models such as rotating the 3D model, zoom in and zoom out, hide and show the parts of the 3D model, etc.

In order to be able to build such an environment and combine all the phases of the data-parallel approach, we need to re-engineer the implemented MapReduce job and XQuery scripts to be able to provide the desired functionality. The reasons for re-engineering these applications is described below.

5.6.1. XQuery Re-engineering

BaseX is the software application that is used to develop and execute all the XQuery scripts implemented during these steps. Executing a script using BaseX is done in two phases. The first phase is to create a XML database for the dataset that the script will operate on. The second phase is to write the query or load the script and execute it. The results of the script will be shown on the BaseX application window.

To be able to develop the desired web application, we need to find a way that enables us to execute the XQurey scripts from the back-end of the web application, and to make these scripts configurable and can be used to preparse and partition different datasets. Fortunately, BaseX provides the user with an API for Java programming language. This API allows the developers to create XML databases, perform XQueries on these databases, executes predefined XQuery scripts and store the results to the file system.

All the implemented XQuery scripts in both the *Preparsing* and *Partitioning* phases have been rewritten in a way that these scripts can be configured or customized to transcode different datasets. For example, the *Extract All shapes* script defines a destination variable to define the path of the output destination directory. This directory will be used to save all the extracted shapes. On the other hand, calling and executing these scripts will be done from the transcoding environment back-end and not form the BaseX application.

5.6.2. MapReduce Re-engineering

Three different MapReduce approaches were developed during this thesis. The high performance approach is the *HDFS Hadoop custom JAR*. For the transcoding environment we can not use the three approaches therefore, we decided to use the high performance approach. The MapReduce job is simply a JAR file that contains all the information needed for the transcoding process such as the split size and the transcoding commands. This means that the JAR file is not suitable for transcoding any dataset and can not be customized. However, one of the features of the transcoding environment is to allow the end user to specify these parameters.

For the provided reasons and to be able to build the transcoding environment, we need to re-engineer the MapReduce JAR to be a configurable JAR file. The configurable JAR file accepts the following parameters. First the *split size* parameter which defines the number of X3D documents per **Map** function. The second parameter is the *input dataset path*. This path is a HDFS path for the shapes folder in the input dataset. The third parameter is the *aopt* commands. These commands are used by every **Map** function to transcode the shapes to binaries. The last parameter is the HDFS path of the output folder.

5.6.3. Transcoding Environment Architecture

Figure 5.12 shows the main components that compose the transcoding environment. These components are the Hadoop cluster, the Tomcat server and the client. The client is a simple web browser that is used by the end user to connect and interact with the web application. The Tomcat server is a web server and servlet container that allows the developer to build Java based web applications. The Tomcat server is hosting the developed web application and is responsible for performing the *Preparsing* and *partitioning* phases. The third component of the transcoding environment is the Hadoop cluster which is used to execute the MapReduce jobs.

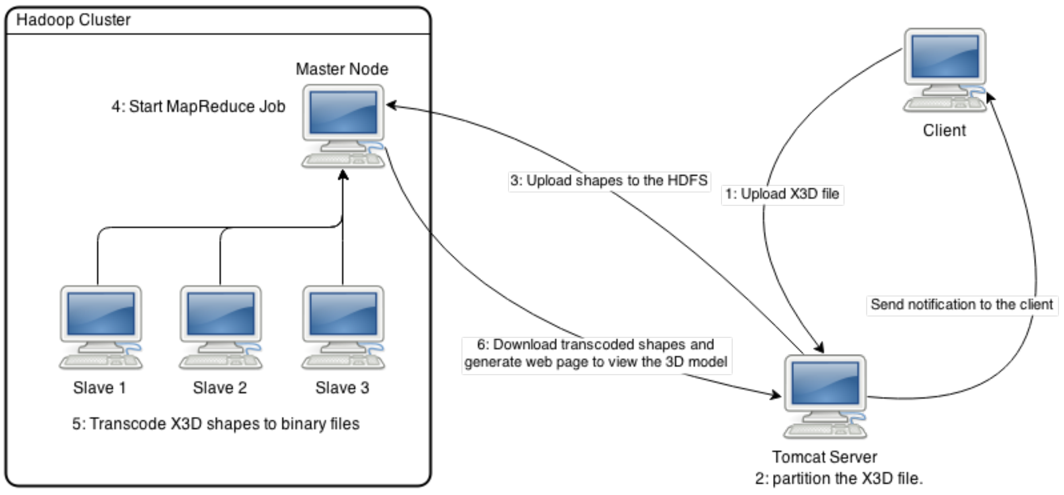
Hadoop Cluster
Master Node
4: Start MapReduce Job
Slave 1
Slave 2
Slave 3
5: Transcode X3D shapes to binary files
3: Upload shapes to the HDFS
1: Upload X3D file
Client
Send notification to the client
6: Download transcoded shapes and generate web page to view the 3D model
Tomcat Server
2: partition the X3D file.

Figure 5.12.: Transcoding environment

From the shown figure, the work-flow of the transcoding process is as follow. First, the user connects to the web application and creates a new transcoding project. In this step, the user needs to specify the parameter of the project such as the transcoding commands, X3D document, and the project name. The web application provides default values for some of the parameters of the project. For instance, the *aopt* commands used in the transcoding process is provided by default. Once the user creates the project, the X3D document will be uploaded to the Tomcat server and the user will be redirected to the projects list. From there, the user can start the phases of the transcoding approach, delete existing projects, download project files, and view transcoded models.

The next step in the transcoding system is the *Preparsing* and the *Partitioning* phases to prepare the data set for further processing. Both of these phases are performed on the Tomcat server. When these phases are complete, the user can start the *MapReduce* phase. In this step, all the shapes will be uploaded to the HDFS file system and then the MapReduce job will be launched to transcode the 3D data. Once the *MapReduce* phase is completed the *Deployment* phase will start automatically. This phases' main goal is to prepare a web page for viewing and controlling the transcoded 3D model. Finally, the end user will be notified when the transcoding process is completed and the 3D model is available for viewing.

Transcoding web application

The Tomcat web server hosts the web application for the transcoding environment. This web application is developed using the Java programming and Java Server Faces. The web application provides the end user with a simple platform that integrates the parts of the data-parallel transcoding approach. The platform is responsible for executing the *Preparsing* phase, the *Partitioning* phases, the *MapReduce* phase, and The *Deployment* phase.

The transcoding web application is connected to a simple *MySql* database that consists of two tables.The first table is called **tblUsers**. This table stores the users login credentials and profile information such as the email and the user's full name. The second table is the **tblProjetcs** table. This table stores project details like the path of the project on the Tomcat server, the project name, the *aopt* commands used by the project, and the project status. In addition, this table links the projects to the users who created these projects.

The transcoding web application is composed of three Java packages, the *dataaccess* package, *common* package, and *transCoder* package. The *dataaccess* package includes classes that define the database connection and function to open the connection , close the connection and execute *MySql* queries on the database. It also includes a database handler class for each of the tables. These classes include functions to create new records, update existing records and delete records from the tables.

The *common* package includes classes that provide global or shared functions. For instance, this package includes classes that are responsible for filtering the requested URLs and ensure that the requested user has access permissions to these URLs. Moreover, this package includes classes that provide utility functions like the function for controlling and managing the session variables of the application.

The *transCoder* package provides the main functionality of the web application. First, it provides the business logic for creating the projects and login functionalities. Second the business logic of the *Preparsing* and *Partitioning* is defined in the class called *XQuery-Partitioner*. This class is using the BaseX API to execute XQuery scripts for preparing and partitioning the X3D document. Third, the class *TransCoder* is responsible for performing all the tasks in both the *MapReduce* and *Deployment* phases.

5.7. Summary

The main goal of Chapter 5 is to describe the implantation details of the proposed data-parallel approach described in Chapter 3. The chapter began by presenting the details of each of the approach phases, then it described the web transcoding environment that integrates all the phases of the approach and provide the end users with a web interface for creating and executing transcoding processes. Below is a brief summary of all the phases and the transcoding environment.

The Preparsing phase is responsible for preparing the input X3D file for further processing in the next phase. Three main tasks are done in this phase to remove the unwanted nodes, replace USE in the X3D document, and create the product breakdown tree. All these tasks are developed using XQuery scripts and BaseX XML database engine. For each of these tasks, an independent XQuery script was developed to achieve the task objectives and goals.

The Partitioning phase is also implemented using BaseX and XQuery scripts. The goal of this phase is to partition the X3D fine in a way that is suitable for data-parallel processing. The proposed method for partitioning X3D files involved three tasks. The first task is to extract all the shapes from the X3D document and replacing them with *inline* tags. The second task is to extract all the parts included in the X3D document. The last task is to extract the main part that represents the complete 3D model from the X3D document.

The MapReduce phase is responsible for transcoding the shapes into binaries. During this phase three different transcoding approaches were implemented. These approaches are the Hadoop streaming, the Hadoop custom JAR, and the HDFS Hadoop custom JAR. The implemented approaches were developed using two technologies, the Hadoop streaming technology and the Java Custom JAR technology.

The Hadoop streaming technology is an alternative to the native method of developing MapReduce jobs. This method allows developers to build MapReduce jobs with a wide range of programming languages like Ruby and Python. The only restriction on the jobs implemented using the Hadoop streaming technology is that both the **Map** and **Reduce** function should read their input from the standard input and write the output to the standard output.

The Deployment Phases' main goal is to create a web 3D viewer for the transcoded 3D model from an HTML template. This phase involves the creation of the web page for the 3D viewer on a Tomcat server and copy all the files needed for rendering the 3D model to the *home* directory of the web page. The 3D viewer template integrates all the rendering required JavaScript libraries and CSS files such as the X3DOM, JQuery and their CSS documents. Moreover, the 3D viewer template integrates the product breakdown tree and the 3D scene of the transcoded model.

In addition, this chapter described the web transcoding environment. The aim of this environment is to provide a web interface that allows the end user to create and transcode 3D models without using terminal commands in an automated way. Building this environment involved the re-engineering of both the XQuery scripts and the MapReduce approach. The goal of the the re-engineering step is to make these scripts customizable and configurable to transcode any X3D dataset. The transcoding web application allows users to upload X3D files and runs the phases of the proposed data-parallel approach, and to view the transcoded 3D model in a web browser.

Part III.
Evaluation and Performance Analysis

6. Evaluation and Performance Analysis

6.1. Overview

System evaluation is an important and necessary step in the development of any system. The evaluation process aims to investigate the performance of the functional and non-functional requirement of the evaluated system. It helps engineers and users in building a clear understanding of the system performance, effectiveness, efficiency, and how the environment parameters affect the system. As a result of this understanding, engineers, researchers and even clients can make decisions based on the evaluation results. For instance, they can decide whether the system needs more investing and improving or that the system is not performing well and cannot be improved anymore, or that the system performance is acceptable and can be used in a production environment.

The evaluation process requires a lot of resources and experiments in order to get meaningful results. The evaluation criteria or evaluation setup may affect the evaluation results and their implications. In our case and in order to perform a valuable evaluation, we decided to build multiple Hadoop clusters with different specifications and test the implemented MapReduce jobs on these different clusters. Performing such an evaluation requires several computers, as well as it also requires a lot of time to install and setup Hadoop clusters on these machines. Thus, the Amazon Elastic Compute Cloud web service is selected to be used as an evaluation platform. This web service will enable us to build and customize Hadoop cluster in a more efficient way than building these clusters on local machines. In addition, it provides us with much more powerful machines than we can get locally.

However, due to the high expenses of using of using the Amazon elastic compute cloud web service (EC2), we decided to only evaluate one of the implemented MapReduce approaches during the thesis and not all of them. Therefore, in order to lower the costs of the evaluation process, we decided to split the evaluation process into two phases: the **local phase** and the **EC2 phase**.

The local phase aims to evaluate the performance of the three implemented approaches: *Hadoop python streaming script*, *Hadoop custom JAR*, and *HDFS Hadoop custom JAR*. This phase will be performed on two Hadoop clusters and the result of this phase will determine the approach that will be used in the next evaluation phase (**EC2 phase**).

The EC2 phase aims to evaluate the high performance MapReduce approach using EC2 web service. To evaluate this script properly, we intend to evaluate the effects of the Hadoop environment, as well as the effects of the system parameters on the performance and efficiency of the implemented approach. Below is a brief description of the evaluation criteria and cases that we intend to perform during the evaluation process.

6.1.1. Evaluated Environment Parameters

Basic geometry evaluation: aims to watch the performance of the MapReduce approach for transcoding different basic geometries.

Split size: aims to present the effects of manipulating the split size for each **Map** function in regards to the processing time.

Shapes number: aims to find the relationship between 3D model shapes number and the time required for transcoding the model into binaries.

Files number: aims to deduce the reflex of increasing the file number that represent the 3D model.

6.1.2. Evaluated System Parameters

Cluster size: aims to conclude the effects of cluster size (number of alive nodes in a cluster) on the performance of the MapReduce parallel processing.

Memeory capacity: aims to conclude the effects of increasing the cluster memory capacity on the performance of the MapReduce job.

Virtual CPUs number: aims to conclude the effects of increasing the virtual CPUs per node in the cluster on the performance of the MapRduce jobs.

The rest of this chapter will present a detailed description of the evaluation setup, goals and results of each of the evaluation cases listed above. The chapter will begin with a motivation to perform the evaluation of the system in section 6.2. After that, section 6.3 will present the evaluation of the three MapReduce approaches during the local evaluation phase, and then section 6.4 will present the results of evaluating the high performance approach. Finally, the chapter will conclude with the results of the evaluation cases, along with a summary of the evaluation process.

6.2. Sequential Script Performance Motivation

One root cause or need of this thesis is the difficulty of transcoding large 3D models in a single machine. Thus, one of the goals of the thesis is to propose and implement a simple and effective data-parallel approach for transcoding the 3D models. Currently, researchers at AirBus Group rely on a sequential script written in Python programming language for transcoding 3D models to binary files. We tested this sequential script on two computers at AirBus Group. The computers used in this experiment are considered to be very powerful computers, their hardware specification is presented in Table6.1 .

Hard Disc	500 GB
RAM	8 GB
CPU	Intel (R) Xeon(R) CPU X3470@ 2.93GHz
Operating System	Windows 7, 64-bit

Table 6.1.: Evaluation computers specification

This experiment involved the transcoding of a simple and small 3D model that contains only **879** shapes in its structure, along with the size of its X3D representation which was relatively small at **84** MB. Although, the computers used in the experiment were very powerful, the model used in the transcoding process was also small in size and shape number, both computers took more than **36** hours to transcode the whole 3D model into binary files. Moreover, one main drawback of the sequential script was that it is very difficult to scale up or to use resources from other machines. The performance of the sequential Python script was one of the main motivational reasons for using the MapReduce programming model and it is also one of the main reasons for performing the evaluation of the system. Comparing the performance of the MapReduce with the sequential script is very helpful in understanding the advantages and drawbacks of the MapReduce approach compared to other approaches.

6.3. Local Evaluation Phase

This section describes the setup of the local evaluation environment and the results of the evaluation of the different MapReduce approaches. The main goal of this evaluation phase is to find out which of the implemented approaches have the highest performance for transcoding 3D models.

6.3.1. MapReduce Scripts

During this thesis, three different MapReduce scripts were implemented with different technologies and architectures. Chapter 5 provides a complete and detailed description of these three approaches and the differences between them. The first approach is implemented using a technology called *Hadoop streaming*. The **Map** and **Reduce** functions are written in python and it uses Pydoop[1] as an API for the Hadoop distributed file system . The second and third approaches are built using the same technology and programming language which are Java custom JAR and Java programming language respectively .

The main difference between these two different approaches is the location of the dataset and the input of each approach. The Hadoop custom JAR approach takes a list of X3D web resources as input, also the dataset is located on a web server outside the Hadoop cluster. While the HDFS Hadoop custom JAR stores the 3D model data on the HDFS instead of storing it on a remote web server and then takes the HDFS dataset path as input.

[1]Python API for Hadoop.

System Parameters	Ariane5 Model	Mercedes Model
Model Size (X3D file)	84 MB	281 MB
Shapes Number	879	125
Split Size	3	3
Maps Count	293	42

Table 6.2.: Local phase models parameters

6.3.2. Environment Description

For evaluating these three scripts, two different clusters were built as described in Chapter 4: **Hadoop Test Environments**. The first environment is a Hadoop single node cluster, this Hadoop cluster consists of only one machine that runs Ubuntu as the operating system and the rest of the specification of this machine is presented in Table 6.1. The other environment is a Hadoop dual node cluster that consists of two machines. Due to the lack of the resources at AirBus Group, the dual-node cluster could not be built with equivalent nodes. One of the machines has the same specification as the machine used for the single node cluster. However, the other one has less CPU power and only contributed **2** GB of RAM for the cluster. More details about the specification of the cluster are provided in Chapter 4.

6.3.3. Evaluation Dataset

The dataset used in the evaluation process of these three scripts consists of two 3D models. The first model is a simple 3D model for the cylinder of the *Ariana5*[2]. This 3D model is provided by AriBus Group and it is not a public case. Therefore, sharing or even transcoding the model using EC2 web service is not allowed. The other model is a public case model provided by the Mercedes-Benz company for one of its trucks. Table 6.2 shows the basic parameters that describe both of the models. These parameters include the size of the original X3D file, the number of the shapes included in each of the models, the split size used in the transcoding process; which is a user defined value that determines the number of transcoded shapes per **Map** function, and finally the **Map** functions count which is the number of **Map** functions used to transcode the whole model to binary files.

6.3.4. Experiment Setup

The implemented scripts used to transcode both models on both of the Hadoop clusters are single node cluster and dual-node cluster. The transcoding process is performed **60** times for each of the 3D models on every environment setup. For instance, the Ariane5 model is transcoded on both of the clusters (single and dual) and on each of these clusters it is transcoded using the three implemented approaches. After the completion of the transcoding process, the average transcoding time of the transcoding process is calculated.

[2]http://www.arianespace.com/launch-services-ariane5/ariane-5-intro.asp Retrieved on 2014-10-24

6.3.5. Results

Figure 6.1 and Figure 6.2 present the results of this experiment in a bar graph style. From the results shown in these graphs, it is obvious that the *HDFS Hadoop custom JAR* approach is performing better than the other two approaches in all the cases. The *HDFS Hadoop custom JAR* is faster than *Hadoop Streaming* and *Hadoop custom JAR* on both of the clusters, the single node and the dual node clusters. In addition, both of the figures showed that there is a big difference in performance between *Hadoop Streaming* approach and the other two approaches.

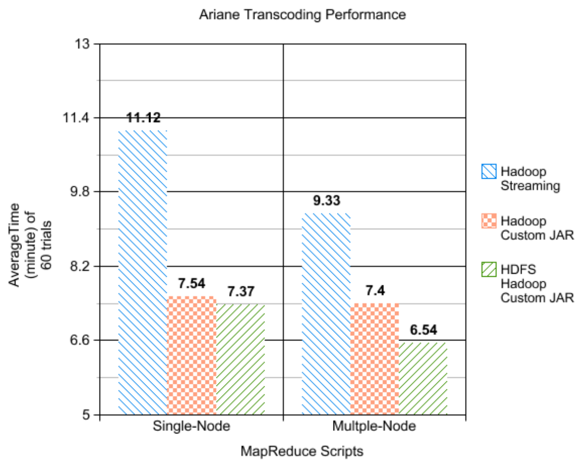

Figure 6.1.: Performance of transcoding Ariane5 model on local clusters

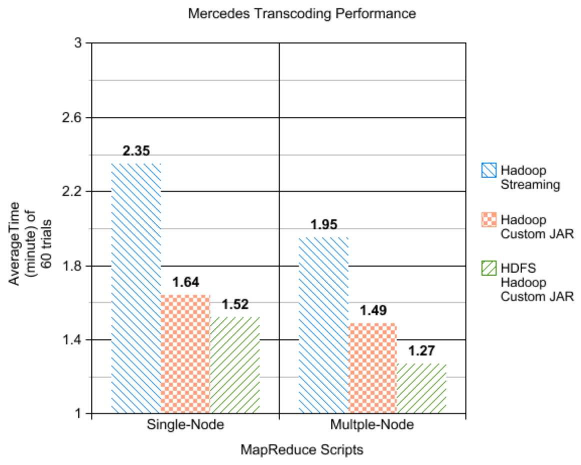

Figure 6.2.: Performance of transcoding Mercedes model on local clusters

The intended aim of this phase was to find out which of the implemented approaches is the highiest performance approach. However, the results of the evaluation showed more than this information. Combining the results of this experiment with the information provided in Table 6.2 about the evaluated 3D models, we found that the *Mercedes* model took less time to completely transcode using all scripts and using both of the clusters. Despite the fact that the *Mercedes* model is larger than the *Ariana5* model in size and the difference between both of the models is **197** MB. In addition, this evaluation case showed that the *Ariana5* model is smaller than the *Mercedes* model and the *Ariana5* model has more shapes included in its structure than the *Mercedes* model.

To conclude, the results of this experiment was beneficial for selecting the best transcoding approach which is the third approach: the **HDFS Hadoop custom JAR** approach. Furthermore, it is clear now that the original X3D file size does not affect the transcoding process and there is no clear relation between the model size and the time required for the transcoding process.

6.4. EC2 Evaluation Phase

The main goal of the EC2 evaluation phase is to describe the evaluation of the HDFS Hadoop custom JAR approach using EC2 web service. As stated in the Overview, there are seven evaluation cases in this section and they can be classified under two categories. The first is the system parameters category, which includes evaluating the effects of shapes number, split size, shape type, and file number on the transcoding time. The other category is the environment parameters category which includes the evaluation of the effects of the cluster size, memory capacity, and virtual CPUs numbers on the performance of the MapReduce jobs.

This section will describe the environment, dataset, experiment setup and the results for each of the evaluation cases. The rest of this section is organized as follows: first, a full description of the environment that is used to run the system parameter evaluation cases will be presented. Since all the evaluation cases in the system parameters category share the same environment, they intended to evaluate the effects of various system parameters only. Then, each of the cases that belong to the system parameter category will be described in details. After that, the environment parameter category evaluation cases are described.

6.4.1. System Evaluation Environment description

In order to perform the evaluation cases from the system parameter category, the preparation and building of powerful Hadoop clusters using EC2 web service is needed. Table 6.3 shows the specification of the nodes used to construct the Hadoop cluster used in the system parameters evaluation. The Hadoop cluster used to run the evaluation cases of this category consist of four nodes with an identical specification. All the nodes contribute with the same amount of resources to the cluster. For instance, each of the nodes contributes **8** GB of RAM to the cluster, which means that the cluster RAM capacity is **32** GB. One of the nodes serves as a *master* node and the rest of the nodes are *slaves* on the cluster.

EC2 Family	General purpose
EC2 Model	m3.2xlarge
Storage Capacity	2 x 80 GiB
RAM	8 GB
vCPUs[3]	8 vCPUs, 2.5 GHz, Intel Xeon E5-2670v2
Operating System	Ubuntu Server 14.04 LTS (HVM), 64-bit

Table 6.3.: The specification of the computers used in the system parameters evaluation

6.4.2. Shape Number Evaluation

The aim of this evaluation case is to find the relationship between the number of the shapes that compose the 3D model and the time required by the transcoding approach to completely transcode all the shapes included in the 3D model. The initial expectation of this evaluation case is that there is a positive relationship between the number of shapes and the transcoding time. This evaluation case is performed on the EC2 Hadoop cluster mentioned in the previous section.

Evaluation dataset

This evaluation case includes the transcoding of eleven different datasets. Each of these datasets consists of a different number of shapes. For instance , the smallest dataset consists of only one shape, while the largest dataset consists of two thousand shapes. The datasets were prepared in this way to be able to recognize the effects of increasing the number of the shapes included in the model on the required transcoding time.

Experiment setup

Each of the prepared datasets is transcoded once using the HDFS Hadoop custom JAR approach on the EC2 cluster. In all the transcoding trials the *Split Size* was fixed to the value **three**. After the transcoding process completes, the corresponding time for each of these datasets is captured and recorded.

Results

Figure 6.3 shows the results of transcoding the eleven different datasets in a line graph style. The results of the experiments shown in the graph meet the initial expectations, and prove that there is a positive relationship between the shapes number and the needed time for transcoding the 3D model. In addition, the graph shows that the relationship between the transcoding time and shapes number is a linear relationship. The more shapes included in the 3D model the more time is spent during the transcoding process.

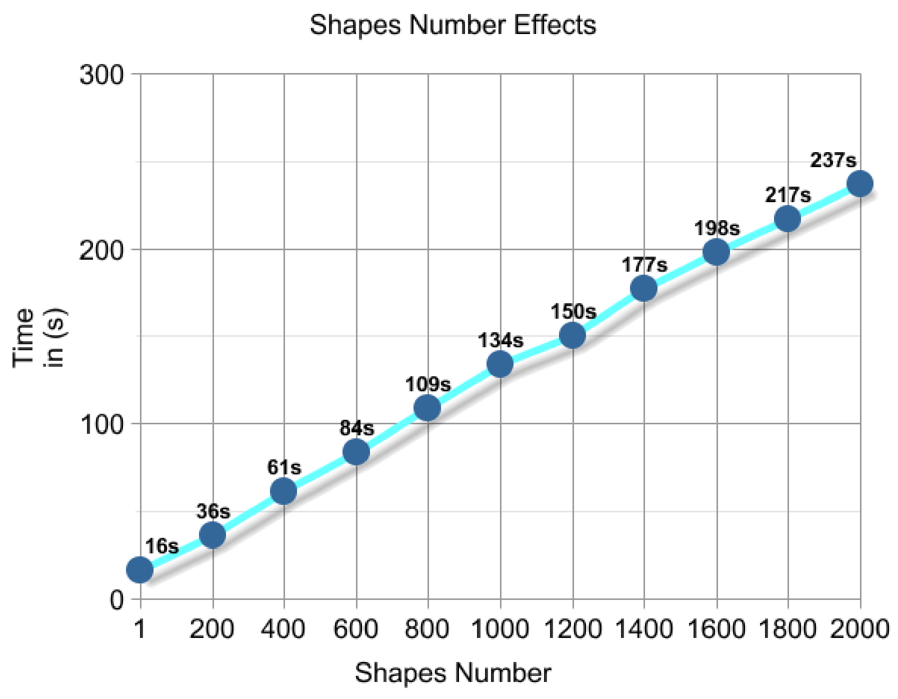

Figure 6.3.: Shape number effects on the transcoding time

6.4.3. Split Size Evaluation

Split size is a numerical configuration parameter used by the MapReduce jobs to configure the **Map** functions. The main purpose of the split size is to define the number of files that each **Map** function can process. For instance, if the transcoded model has a **100** shapes, then by using the split size the number of **Map** functions that is used in the transcoding process can be defined. In cases where the split time value is set to **2**, then **50 Map** functions will be used in the transcoding process. Each of these functions will be responsible for transcoding only **2** shapes and terminates after that.

The goal of the split size evaluation case is to figure out if there is a relationship between the split size and the required transcoding time. The expected result of this evaluation case is to find a positive relation between these two variables. The main reason behind this expectation, is that when the split size increases the Hadoop processing balance will be affected and therefore, not all the resources on the cluster will be used during the transcoding process. As a result, the transcoding time will also increase.

To illustrate this argument, imagine that we have a cluster that consists of **three** nodes and we want to transcode a model that contains **100** shapes; if the split size value is set to a value that is less than **50**, then the number of **Map** functions will be at least **3** and therefore, all the nodes in the cluster will participate in the transcoding process. However, if the split size value is set to a value that is equal or larger than **50**, then the generated **Map** function are **2** functions or **1** function. As a result, only two nodes will be used in the transcoding process. For this reason, the transcoding time will be affected by the split size and it will increase as the split size increases. This evaluation case uses the same Hadoop EC2 cluster as the one mentioned in section 6.4.1.

Evaluation dataset

Unlike the previous evaluation case, the split size evaluation case is performed using only one dataset. The used dataset consists of **2000** shapes. To prepare this dataset, a basic geometry shape had been extracted from a 3D model and duplicated **2000** times. This way, all the shapes that compose the dataset are similar and that the size of these shapes does not influence the transcoding process.

Experiment setup

To get precise and helpful results, this experiment is performed in the following way: dataset is transcoded using the HDFS Hadoop custom JAR **11** times. Each of these transcoding trials were launched to transcode the same dataset, but each of them has a different split size value. The first experiment in this evaluation case is performed with a split size value equal to **1**. The remaining experiments are performed with different split size values. The split size values used in these experiments are in the form of power of **2** values like **4, 8, 16, and 32**.

Results

Figure 6.4 illustrates the results of transcoding the dataset that consists of **2000** shapes **11** times with different split sizes. The first part of the graph - from split size **1-32** - was not expected, it was not clear that for a very small split size value the transcoding time will not be the shortest transcoding time. The initial expectations of this case were that small split sizes will generate many **Map** functions and therefore, the balance of processing the data will be more flexible. As a result, this way all the resources on the cluster are used in the transcoding process.

On the other hand, the rest of the graph - after split size value **32**- met the initial expectation and shows that there is a positive relation between the split size and the required transcoding time. The graph also shows that the best split size value for the used dataset was in the range of **16-64**. In this range, the transcoded time was the shortest transcoding time for all the trials which was **81** seconds.

The initial explanation for the results of this evaluation case is that for the small split size values, the MapReduce job introduces more processing overhead on the Hadoop cluster, since each of the **Map** functions needs to be initialized and terminated, this requires an increase in the overall startup time for all **Map** functions. To illustrate, with a split size value of **1** a **2000 Map** functions are generated, while with split size value of **2** only a **1000 Map** functions are generated. The presented graph shows that there is a big enhancement on the transcoding time between these two points.

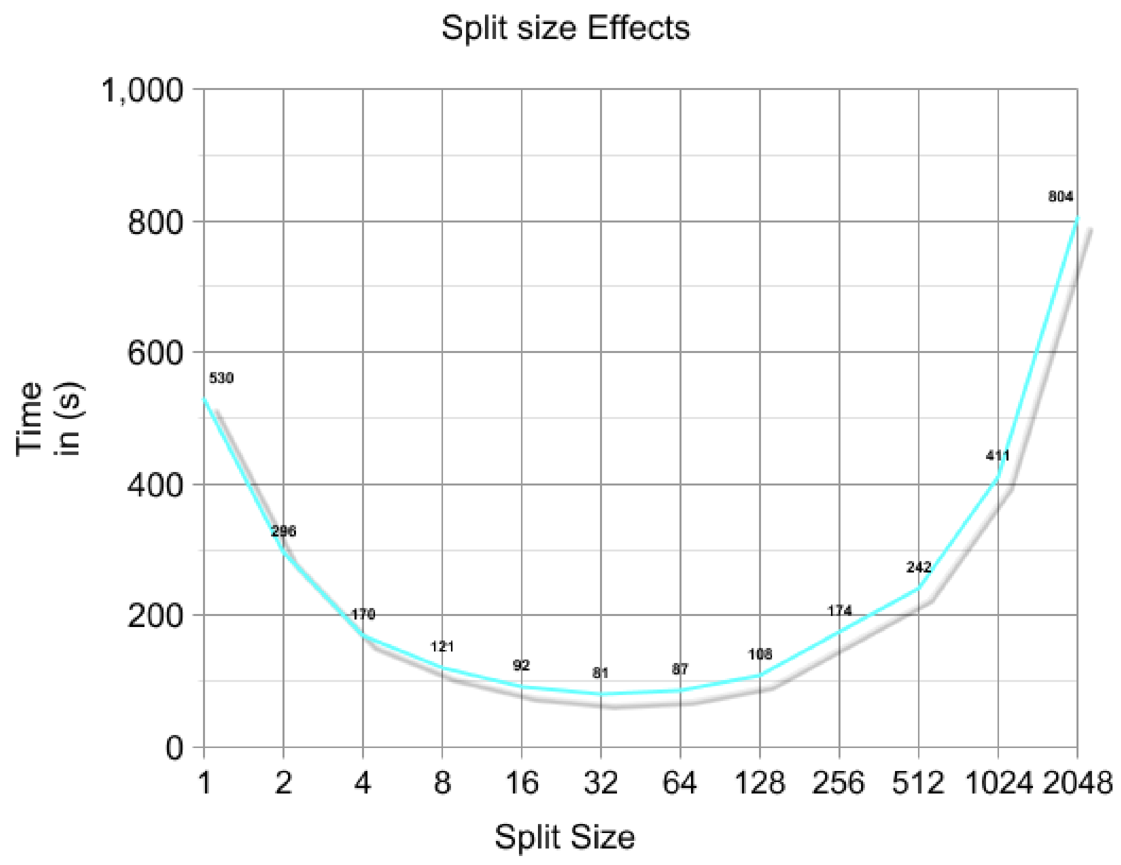

Figure 6.4.: Split size effects on the transcoding time

Then, the transcoding time keeps decreasing, while the split size value increases till it reaches a point where the increase of the split size will stop affecting the processing overhead time and it starts to affect the loading balance of **Map** functions on the nodes of the clusters. At that point the transcoding time begins to increase each time the split size is incremented.

6.4.4. Basic Geometry Evaluation

Basic geometry evaluation case aims to find the effects of transcoding different geometric shapes on the transcoding time. In this case, the nature or the definition of the shape itself is evaluated to find out the relationship between transcoding time and the shape's type. In other words, this evaluation case will answer the following questions; is there shapes that require more transcoding time than the other shapes? In case that the shape type affects the transcoding time; what are the key parameters that increase the transcoding time? Is it the shape file size?. The initial expectation of this phase is that there is a difference in the transcoding time for transcoding different shape types.

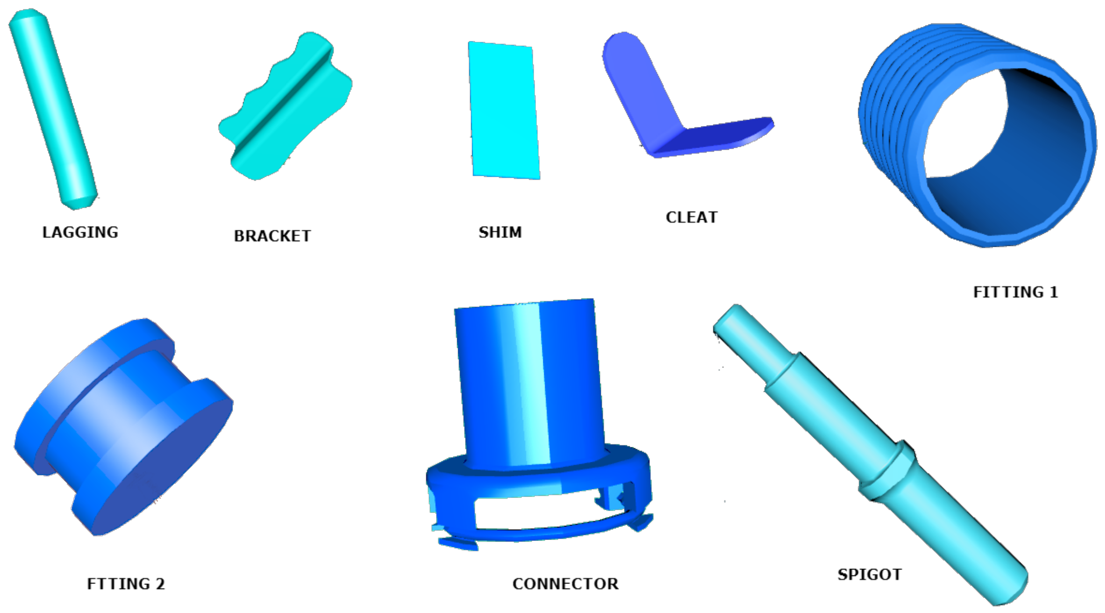

Figure 6.5.: Geometry shapes used in the basic geometry evaluation

Evaluation dataset

For the purpose evaluating the effects of shape type on the transcoding time, **8** different basic geometry shapes have been extracted from a 3D model for the **Eurofighter**[4] . These shapes are presented in Figure 6.5. As shown in the figure, all the shapes are simple geometry shapes, however, not all of them have the same complexity and size. For instance, **FITTING 1**, and **CONNECTOR** are considered to be more complex than **CLEAT** and **SHIM**. Table 6.4 shows the used shapes and their X3D file size. Also, it is clear from the table that these shapes have different X3D file sizes.

For each of these basic geometries, a separate dataset is created. Each of the generated datasets consist of the same shapes number, which is **2000**. The reason for using a large number of shapes in these datasets is to be able to capture the difference in the transcoding time. From the shape number evaluation case, it is known that the transcoding time for datasets that includes one shape is very small and therefore, it is not possible to capture the difference. On the other hand, using a large number of shapes will help us in capturing the difference even if it is a small difference.

Experiment setup

During this evaluation case, each of the mentioned datasets are transcoded on the Hadoop EC2 cluster three times in a row. After that, the average of the transcoding time is computed and recorded. The reason for performing **3** trials for each dataset and using the

[4]http://www.eurofighter.com/ Retrieved on 2014-10-30

Shape	Shapes Number	Shape size
LAGGING	2000	28 kb
BARCKET	2000	18 kb
SHIM	2000	2 kb
CLEAT	2000	7 kb
FITTING 1	2000	189 kb
FITTING 2	2000	15 kb
CONNECTOR	2000	66 kb
SPIGOT	2000	18 kb

Table 6.4.: Basic geometry datasets description

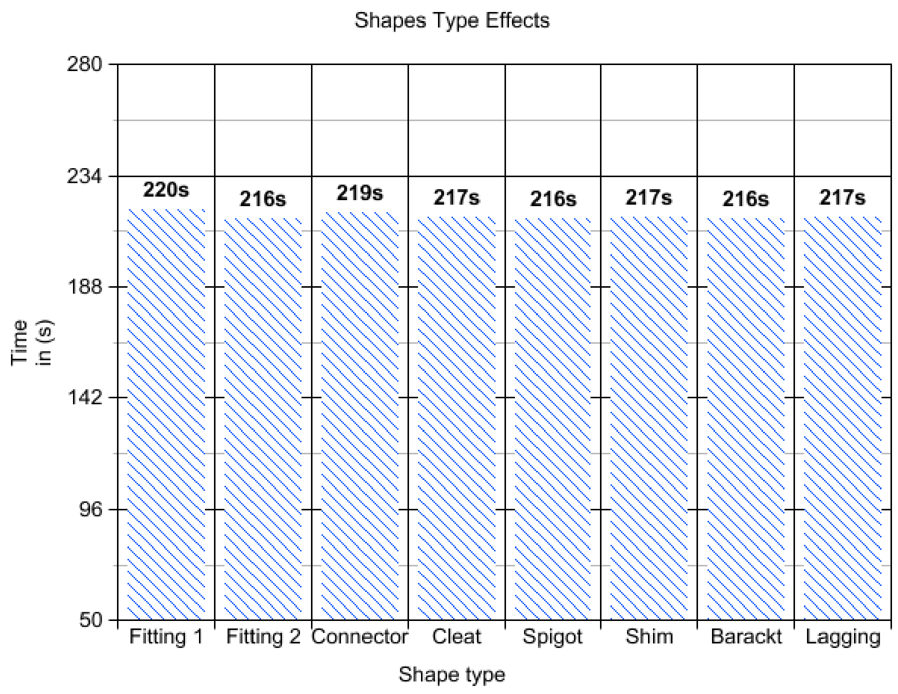

Figure 6.6.: Sahpe type effects on the transcoding time

average of these trials instead of a single transcoding process, is to get more precise results and to make sure that the transcoding time for the same dataset is very close in the three trials. All the transcoding trials in this phase are launched with a split size value equal to **3**.

Results

Figure 6.6 shows the average transcoding time of three trials for transcoding each of the **8** datasets presented in Figure 6.5. As stated above, the initial expectation of this case is that there would be a big difference in the transcoding time for different shapes with different file sizes or complexity. However, the results of this evaluation case shows that there is no noticeable difference in the transcoding time required to transcode different basic shapes, and that the file size of these shapes does not influence the transcoding time in any way.

6.4.5. File Number Evaluation

The partitioning process divides the X3D input file into small chunks and then these chunks are transcoded to binary. In the split size, shapes number, and basic geometry evaluation cases, each of these chunks contains only one shape. The intend of this evaluation case is to find out whether the file number that represents the 3D model influences the required transcoding time or not.

Since the transcoding process is simply iterates over a list of X3D files and converts them to binaries, The initial expectation of this evaluation case is that there is a positive relationship between the processing time and the number of files used to represent the 3D model. Increasing the file number will lead to an increase of the transcoding time, because the file number increment introduces more data processing during the transcoding process.

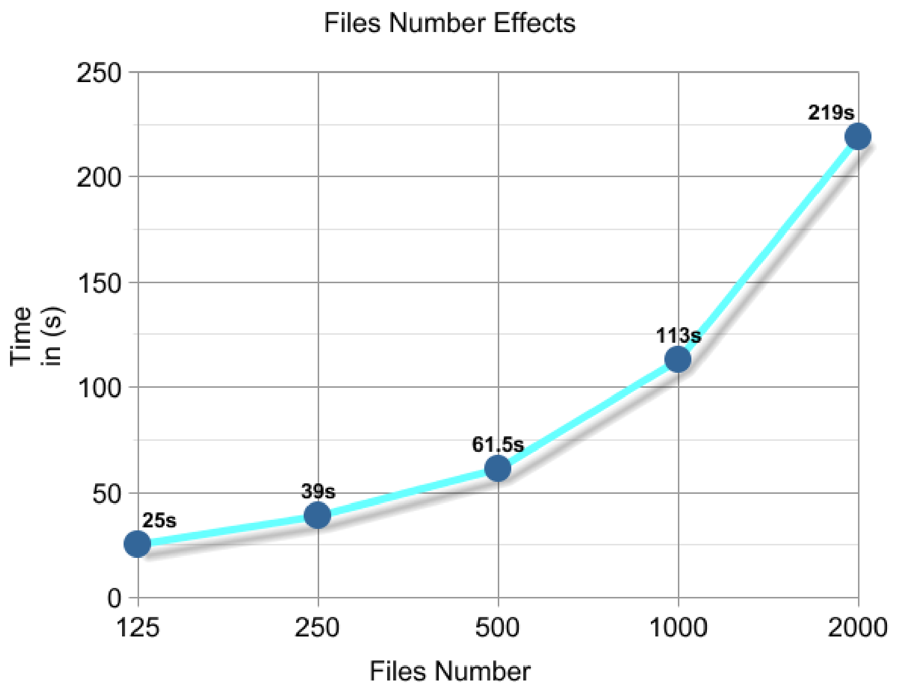

Figure 6.7.: File number effects on the transcoding time

Evaluation dataset

For the evaluation purposes of this case, **5** different datasets were prepared. All the five datasets include the same number of shapes, which is **2000** shapes. The only difference between these datasets are how the shapes are represented in these files. Table 6.5 presents the details of the five datasets in terms of shapes number and files number. To illustrate,

Dataset	1	2	3	4	5
File Number	125	250	500	1000	2000
Shapes Number	2000	2000	2000	2000	2000
Shapes per file	16	8	4	2	1

Table 6.5.: File number evaluation dataset

the first dataset contains **125** files and each file consists of **16** shapes. While the last dataset consists of **2000** files, where each file contains only **1** shape.

Experiment setup

This experiment performs the transcoding of the five different datasets on the Hadoop EC2 cluster mentioned in section 6.4.1. Each of the datasets is transcoded three times. After that, the average of the three trials is computed and used in creating the result graph. To get more accurate results for this evaluation case, the average of the trials is used instead of a single transcoding process time. All the transcoding trials in this phase are launched with a split size value equal to **3**.

Results

Figure 6.7 shows the results of the file number evaluation case. From the shown graph, it is clear that there is a strong positive relationship between the file number and the required transcoding time. For instance, the transcoding of the model represented by **2000** files took **209** seconds, while transcoding the same model, but represented by only **125** files took only **25** seconds. The results of this case show that the transcoding process time can be enhanced dramatically by decreasing the number of files that represent the 3D model.

One of the main goals of the transcoding process, is to be able to create web resources for each individual shape in the 3D model. Representing a 3D model with a less number of files will lead to grouping and converting of all the shapes included in one single file to one binary file. As a result, it is not feasible to create web resources for all the individual shapes in that 3D model. However, in some cases engineers are not looking for the details of the 3D model, but they are looking for a specific level of granularity. In these cases, the number of the files that represent the 3D model can be changed to enhance the transcoding time. This change will lead to loss of some of the details of the 3D model, but no one is interested in viewing these details in this case.

6.4.6. Memory Capacity Evaluation

The memory capacity evaluation case is the first evaluation case that evaluates one of the environment parameters instead of the system parameters. This evaluation case intends to investigate the effects of increasing the RAM capacity of the Hadoop cluster, and capture how the transcending time is influenced by this increment. This evaluation phase is performed on multiple Hadoop clusters with different hardware specifications. The initial

EC2 Family	Memory Optimized
EC2 Model	r3.4xlarge
Storage Capacity	1 x 320 GiB
vCPUs	16 vCPUs, 2.5 GHz, Intel Xeon E5-2670
Operating System	Ubuntu Server 14.04 LTS (HVM), 64-bit

Table 6.6.: The specification of the computers used in the memory capacity evaluation

expectation for the relationship between the memory capacity and the required transcoding time is as follow, adding more memory capacity to the nodes of the cluster will decrease the transcoding time, however, it is expected that after a given point, increasing the memory capacity will not affect the required transcoding time and that this time will become almost constant. Below is a brief description of the environment, dataset, experiment setup, and the results of this evaluation case.

Environment description

The memory evaluation case experiments are performed on a set of Hadoop clusters. Each of these clusters has the same hardware specification except for the memory capacity. Table 6.6 shows the shared hardware specification between all the clusters used in this evaluation case. Each of the clusters consists of four nodes; one node is the *master* node and the rest are *slave* nodes. All the four nodes that compose the cluster are participating in the transcoding process .

During this evaluation case, seven Hadoop clusters were built. Each of these clusters has a different memory capacity. For instance, in the first cluster built, each of the nodes contributes **1** GB of RAM for the cluster. That means the total cluster memory capacity is **4** GB. On the other hand, each of the nodes in the last cluster contributes a **64** GB of RAM to the cluster, which leads to a Hadoop cluster with **256** memory capacity. This way, transcoding 3D data on these clusters will enable us to demonstrate the effect of memory capacity on the transcoding time.

Evaluation dataset

The dataset used in the memory capacity evaluation case consists of a 3D model that contains **2000** shapes. During this evaluation case, all the experiments performed on the set of Hadoop clusters were operated on the same datasets. For this case, using the same datasets in all the experiments will eliminate any affects that the datasets has on the transcoding time, and will generate more accurate results.

Experiment setup

This evaluation case involves the transcoding of the described dataset seven times. Each time on one of the Hadoop clusters built during the evaluation case. The capacity of the used clusters range from **4** GB to **256** GB.

Results

In regards to the effects of the increasing of the memory capacity, the results of this evaluation case are very similar to the initial expectations. For the evaluation, Figure 6.8 shows a line graph that represents the results of the transcoding of the dataset on the seven Hadoop clusters. The figure shows that the transcoding time enhances if the cluster memory capacity increases. However, after a given point, in our case the point where the memory capacity is equal or larger than **32** GB, the transcoding time becomes almost constant, and increasing the cluster memory capacity does not enhance or improve the transcoding time any more.

An explanation for these results is that the transcoding process of any dataset requires a specific amount of memory for executing all the **Map** functions. In a case where the cluster has less memory capacity, Hadoop will manage a queue for the **Map** functions and it will execute a set of these functions at a given time. If the memory capacity of the cluster is increased, the Hadoop framework will execute more **Map** functions at the same time. However, if the memory capacity is larger than the memory needed for the transcoding process, Hadoop will use only a subset of the memory and the rest will not affect the performance of the transcoding approach.

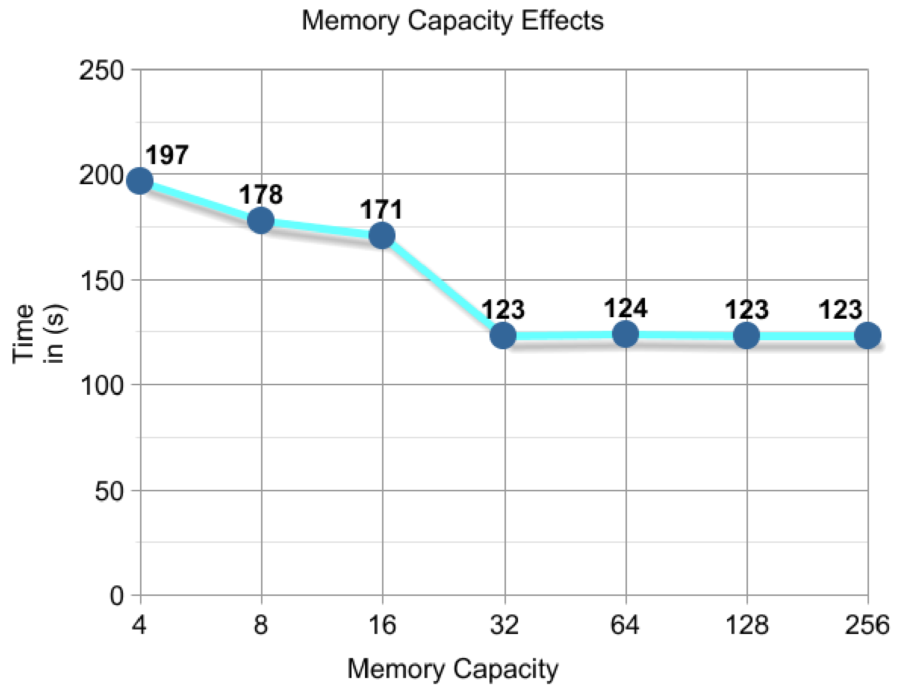

Figure 6.8.: Memory capacity effects on transcoding time

EC2 Family	Compute optimized
RAM	28 GB
vCPU	2.8 GHz, Intel Xeon E5-2680v2
Operating System	Ubuntu Server 14.04 LTS (HVM), 64-bit

Table 6.7.: The specification of the computers used in the virtual CPUs number evaluation

6.4.7. Virtual CPUs Number Evaluation

Hadoop is an open source framework for powerful distributed processing of data using commodity clusters. In other words, Hadoop clusters could be composed of computers that are not very powerful or have high hardware specifications. However, building Hadoop clusters using powerful computers makes these clusters more powerful and therefore more productive in processing large amounts of data.

For the above reason, the aim of this evaluation case is to find out the effects of adding more processing power on the Hadoop clusters nodes. The virtual CPUs number evaluation case evaluates one of the environment parameters and therefore, this case involves the creating of multiple Hadoop clusters to perform the experiments that are intended to be performed to get the results. The virtual CPUs number evaluation case aims to find the relation between the transcoding time and the number of virtual CPUs for the instance.

As for the memory capacity evaluation case, the expectation of this evaluation case is that the increase of the virtual CPUs number will lead to an increase of the performance of the transcoding process, and this will lead to a decrease in the required transcoding time.

Environment description

The virtual CPUs number evaluation cases involved the building of four different Hadoop clusters. Each of these clusters consists of four nodes from the *Compute Optimized* EC2 family. Table 6.7 shows the shared hardware specifications between the four clusters. The only difference between these four clusters is the number of used vCPUs in each of the nodes that composes the clusters. The first cluster consists of nodes that have **4** vCPUs for each of them; on the other hand, the second cluster is composed of nodes that have **8** vCPUs and each of the nodes that composes the third cluster have **16** vCPUs. Each of the nodes that composes the last cluster have **32** vCPUs.

Evaluation dataset

The dataset used in the virtual CPUs Number evaluation case consists of a 3D model that contains **4000** shapes. All the experiments performed on the set of Hadoop clusters during this evaluation case operated on the same dataset. The same dataset in all the experiments to eliminate any effects of the dataset on the transcoding time.

Experiment setup

This evaluation case involved the transcoding of the described dataset four times. Each time on one of the Hadoop clusters built during the evaluation case. All the experiments performed during this evaluation case used the same system parameters. For instance, all the transcoding processes are performed with a split size value equal to **3** . Moreover, all the transcoding experiments operated on the same dataset, which is composed of **4000** shapes.

Results

Figure 6.9 presents the results of transcoding the dataset on Hadoop clusters described above. From the presented graph, two major points can be concluded. The first point is that the virtual CPUs Number has a huge influence on the required transcoding time. Each time the virtual CPUs number in the nodes of the cluster is increased, the transcoding time decreased significantly. For instance, the difference in the transcoding time between the first cluster with **4** virtual CPUs per node and the second cluster with **4** virtual CPUs per node is more than five minutes. In addition, the difference in the transcoding time between the second and third cluster is almost three minutes.

The second point that can be concluded from the graph is that the enhancement in the required transcoding time is not a fixed value. For instance, the enhancement of the transcoding time between the first and the second cluster is five minutes and thirty seconds while the enhancement in the transcoding time between the third and the fourth cluster is only thirty four seconds.

To conclude, composing clusters from nodes that have a high number of virtual CPUs will enhance the transcoding time dramatically. However, at a given point, this enhancement will become very close to zero. In addition, what applies to the virtual CPUs also apply to the CPUs and the number of cores contained in the CPUs.

6.4.8. Cluster Size Evaluation

Hardware specification of the individual nodes that composes a Hadoop cluster is not the only factor that influences the processing power of the cluster. The number of the nodes that compose the cluster is also affecting the cluster's overall power. Adding nodes to a Hadoop cluster will lead to an increase in the resources that are available for the cluster and therefore, the processing power of the cluster will be enhanced.

The main purpose of this evaluation case is to figure out the relationship between the number of the active nodes in a cluster and the required transcoding time. As stated above, The initial expectation of this case is that the transcoding time will decrease each time the active nodes number in the cluster increases. However, The goal of this case is to check if the number of nodes will have a linear negative relation to the transcoding time or not. For example, does the transcoding time becomes constant after a specific number of nodes.

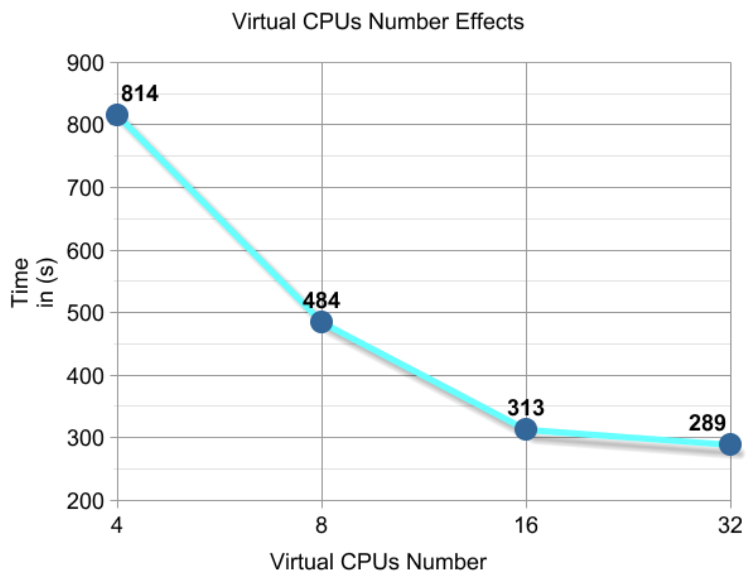

Figure 6.9.: Virtual CPUs number effects on transcoding time

EC2 Family	General purpose
EC2 Model	m3.2xlarge
Storage Capacity	2 x 80 GiB
RAM	7 GB
vCPUs	8 vCPUs, 2.5 GHz, Intel Xeon E5-2670v2
Operating System	Ubuntu Server 14.04 LTS (HVM), 64-bit

Table 6.8.: The specification of the computers used in the cluster size evaluation

An important note to mention here is that increasing the nodes number in a cluster will increase the computation power of the clusters, and it will increase the available resources for the Hadoop cluster. For example, adding a new node to a cluster will increase the memory capacity of the cluster and it will also add more processing power to the cluster since the CPU of the new node will also be used in the data processing.

Environment description

In order to be able to evaluate the effects of the cluster size, six clusters with the same hardware specifications were build during this evaluation case. These clusters differ from each other in the number of the nodes that composes each of them. Table 6.8 shows the hardware specification of all the nodes used in this evaluation case. The first cluster used in this evaluation case consists of **2** nodes, each of the remaining clusters consist of **4**, **8**, **16**, **32**, **64** nodes. This way will guarantee that more accurate results for the effects of the cluster size on the required transcoding time are generated.

Evaluation dataset

The dataset used in the cluster size evaluation case consists of a 3D model that contains **8000** shapes. All the experiments performed on the set of Hadoop clusters during this evaluation case operated on the same dataset. The same dataset is used in all the experiments to eliminate any effects of the dataset on the transcoding time.

Experiment setup

During this evaluation case a set of powerful Hadoop clusters had been built. These clusters consist of **2** to **64** active nodes. On the built cluster, the cluster size evaluation case involved transcoding the described dataset six different times. Each transcoding process was done on one of the built clusters and thus each transcoding process was done using a different number of active nodes. For instance, the first transcoding process is performed using only **2** nodes while the last transcoding process is done using **64**. All the transcoding experiments done in this evaluation used the exact same dataset and were transcoded using a split size value equal to **3**.

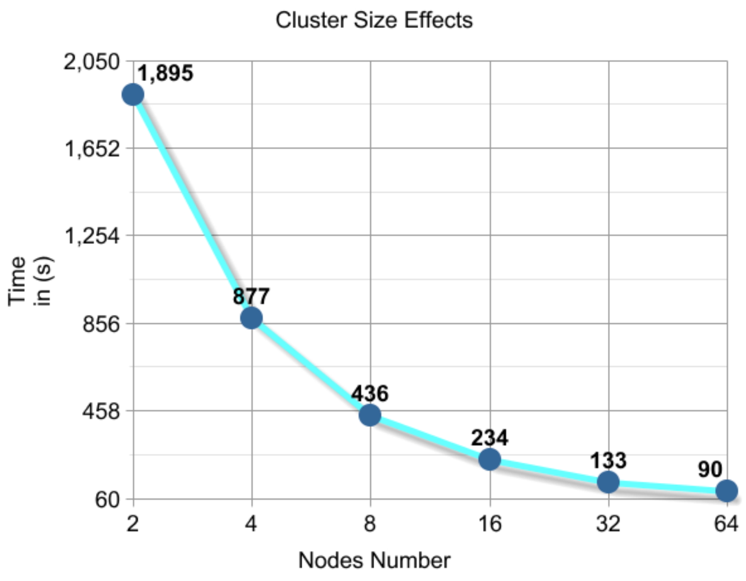

Figure 6.10.: Cluster size effects on transcoding time

Results

Figure 6.10 presents the results of the cluster size evaluation case. From the shown graph, it can be concluded that the number of active nodes in a cluster has a big influence on the required transcoding time. For instance, transcoding the dataset on a cluster that consists of two active nodes took more than **30** minuets, while transcoding the same dataset on a cluster that contains **64** nodes took only **one minute** and **30** seconds.

Another thing that can be concluded from the graph, is that the enhancement of the transcoding time is very large at the beginning, but after a while it slows down till it reaches a point where there is no more enhancement on the transcoding time. The results of this evaluation case is very similar to the results of the *Memory Capacity* and *Virtual CPU Number* evaluation cases. The same reasons that are used to explain the results of these cases can be used to explain the results of this case.

Cluster nodes represent a processing power in both memory and CPU terms. The transcoding process requires a specific amount of these resources. Adding a node to the cluster will increase the available resources and therefore, the transcoding process will take the advantage of these resources. However, if the available resources are more than the resources that are required by the transcoding process, the extra resources will not by used and therefore, they will not affect the transcoding process performance.

6.5. Summary

The main goal of this chapter is to describe the evaluation and performance analysis process used to evaluate the implemented system. System evaluation helps engineers to get a clear understanding of the behavior of the system that is under the evaluation. Evaluation process highlights some of the weaknesses or drawbacks of the evaluated system. Moreover, evaluation results are considered as a basis for many decisions. For instance, engineers can decide whether further investment and improvement of the system is needed or not.

The evaluation process of the MapReduce jobs consists of two phases, the *Local* phase and the *EC2* phase. The main goal of the local phase is to select the MapReduce approach with the highest performance to evaluate it using the EC2 web service. This phase included the transcoding of two different models, the **Ariane5** and the **Mercedes** models. Both of these models transcoded using the three MapReduce approaches of the transcoding process to find out which is the best approach among these approaches. The result of the *Local* phase evaluation showed that the HDFS Hadoop custom JAR is the highest performance approach and therefore, this approach is selected for further evaluation on the *EC2* evaluation phase.

The *EC2* evaluation case aims to evaluate the system and environment parameters of the selected MapRedcue Job on the Amazon EC2 web service. To achieve this goal, the *EC2* phase performed seven different evaluation cases in order to measure the effects of the system and environment parameters on the transcoding time. The performed evaluation cases are briefly described below.

Shape Number Evaluation Case: This evaluation case aims to find the relationship between the transcoding time and the number of the shapes included in the 3D model. The results of the case showed that 3D models that contain more shapes will need more time to be transcoded completely.

Split Size Evaluation Case: The aim of this evaluation case is to find the effects of changing the split size for the transcoding process on the transcoding time. This evaluation case showed that the split size influences the transcoding time in two ways. First, for small split size values, increasing the split size value will lead to a decrease in the transcoding time. Second, for large split size values, increasing the split size value will lead to an increase in the transcoding time. The results showed that there is a specific point where the affects of the split size is changes from decreasing the transcoding time to increasing it.

Basic Geometry Evaluation Case: This case investigates the difference between transcoding different geometry shapes and whether the shape type influences the transcoding time. Depending on the results of this evaluation case, the conclusion that there is no big difference in the transcoding time for transcoding different geometric shapes that diverse in the X3D and the complexity of the shape can be drawn.

File Number Evaluation Case: The main goal of this case is to find out if the number of the files used to represent the 3D model affects the transcoding time. The evaluation case showed that the representation of the X3D affects the transcoding time. For instance, transcoding the same dataset with a different file number will have different transcoding time. If the 3D model is represented by less number of files, the transcoding time will enhance and it will be smaller if the model is represented by a larger number of files.

Memory Capacity Evaluation Case: This evaluation case aims to find the effects of increasing the memory capacity of the cluster nodes on the transcoding time. Some points are noticed in this evaluation case. The first is that the memory capacity influences the transcoding time, and the second is that the increase of the capacity will lead to an enhancement of the transcoding time, till a particular point where the part of the memory capacity will not be used for the transcoding process.

Virtual CPUs Number Evaluation Case: The aim of this case is to investigate the influences of the Virtual CPUs Number on the transcoding time. Relying on the results of this phase, it is obvious that the increase of the virtual CPUs number will lead to a decrease in the required transcoding time. However, the enhancement of the transcoding time is not linear and after a specific point the enhancement becomes close to zero.

Cluster Size Evaluation Case: The goal of this case is to figure out if the number of nodes that composes the Hadoop cluster affects the transcoding time. This case showed that the increase of the nodes number will lead to the decrease of the transcoding time . It illustrated that the enhancement of the transcoding time is big, till a specific point where the increase of the node number will lead to adding extra resources that are not needed for the transcoding process. At that point the transcoding time almost becomes a constant.

7. Conclusion and Outlook

7.1. Summary

AirBus Group aims to develop a web based collaboration environment for managing and exchanging 3D models. The key entry point for this environment is the DMU of the modeled object. The main objective of the environment is to allow the various stakeholders of the 3D modeling process to view the DMU, exchange or share the DMU, and attach analysis information to the parts of the DMU. Moreover, the environment allows the engineers to have more than one view for the same DMU. One advantage for this environment is that it provides these features without requiring any tool licenses and therefore, this environment is an alternative for the expensive CAD tools in viewing 3D models.

Developing such an environment is not an easy task and there are many problems and challenges to accomplish this vision. One of these problems is how to bring the DMU to a web browser and render it efficiently. The current technologies such as X3DOM and HTML support rendering 3D models in web browsers. However, for large 3D models the rendering process is very slow and it may lead to crash the web browser. One solution for this problem is to transcode these models to binaries to reduce their sizes. The sequential transcoding process is time consuming, and therefore, the need for another method for transcoding the 3D models is raised.

This thesis contributes in solving this problem by focusing on defining and implementing a data-parallel transcoding system for X3D/XML data. The proposed system provides the user with an alternative method for transcoding the X3D data. The main objective of this system is to reduce the transcoding time required to transcode a 3D model.

The MapReduce programming model influences the proposed data-parallel approach. The approach is simply a pipeline of four phases. Each of these phases operates on the 3D data set and passes it to the next step for further processing. The first phase is the *Preparsing* phase, which is responsible for preparing the original X3D file for the *Partitioning* phase, and to make sure that each of the geometries in the X3D document is independent from other geometries. The second phase is the *Partitioning* phase, this phase partitions the original X3D file to several chunks of files to prepare the dataset for parallel transcoding. The third phase is the *MapRduce* phase, which is responsible for transcoding the partitioned X3D documents in parallel. The *Deployment* phase is the last phase in the proposed approach; this phase is responsible for deploying the 3D transcoding model to a web server that allows the end users to interact with the 3D model.

During the implementation of the proposed data-parallel approach several programming languages and technologies were used. For the tasks of both the *Preparsing* and *Partitioning* phases, XQuery and BaseX were used to prepare the X3D document and partitioning it. The *MapReduce* phase involves the development of three different MapReduce jobs to transcode the X3D data. These jobs are developed using different technologies such as Hadoop Streaming and Hadoop Custom JAR. In the *Deployment* phase, a web environment was built to enable the end users to transcode 3D models and view them. The main objective of this web application is to provide the end user with a data-parallel transcoding system with a interface.

In addition, this thesis performs an experimental evaluation and performance analysis of the implemented system. The goals of this evaluation are to measure the effectiveness of the proposed system, and to find out the effects of the system and environment parameters on the transcoding time. The evaluation process is split into two phases. The first phase is the local phase, which aims to evaluate the performance of the implemented MapReduce jobs to find out the highest performance MapReduce job. The evaluation cases of this phase are performed on local clusters at AirBus Group.

The second evaluation phase is the EC2 evaluation phase. The goal of this phase is to figure out the effects of the Hadoop environment and the Mapreduce system parameters of the transcoding time. The evaluation cases included in this phase are the effects of shapes number, shapes type, file number, split size, memory capacity, virtual CPU number, and cluster node on the transcoding time. During this phase, multiple Hadoop clusters were built to perform the evaluation cases in this phase. These clusters have different hardware specifications and node numbers.

The results of the local evaluation showed that the HDFS Hadoop custom JAR is the highest performance MapReduce job among the implemented jobs on both the single and dual Hadoop cluster. Also, all the cases showed that the Hadoop streaming technology has worse performance than the custom JAR technology. In addition, this phase showed that the X3D files size in not a key factor of the transcoding process and it does not influence the transcoding process in any way.

On the other hand, the EC2 evaluation cases show that the shapes number and file number have a positive linear relationship with the transcoding time. Each time the shapes number or the files number increases the transcoding time will increase linearly. Another observation from the EC2 evaluation cases is that the shape's type does not influence the required transcoding time in any way. Furthermore, memory capacity, number of virtual CPUs, and cluster size have the same effects on the transcoding time.

Enhancing the Hadoop cluster hardware specification by adding more resources of these proprieties, will lead to enhancing the performance of the transcoding process and decreasing the required time for transcoding the 3D model. However, at a specific point the enhancement becomes close to zero and the transcoding time becomes almost a constant. Finally, the split size influences the transcoding time in two ways. Incrementing small split size values – less than **40** - will lead to an enhancement of the transcoding process while incrementing large split size values will lead to an increase in the transcoding time.

To conclude, the proposed data-parallel transcoding approach suggested another method for transcoding 3D geometries into binaries. Using this approach the transcoding process is quicker and more efficient, and the transcoding time is dramatically reduced. For instance, the Ariane5 3D model took more than **36** hours to be transcoded using the sequential script, while it took around **7** minuets to be transcoded completely using the HDFS Hadoop custom JAR. In addition, the required transcoding time is influenced by some system and environmental properties such as the shapes number and the cluster size.

7.2. Future Work

The proposed data-parallel transcoding approach helps us in reducing the transcoding time and in achieving our objectives. Nevertheless, there are still open issues and problems in achieving our vision in building an efficient and effective web collaborative environment. These issues and problems should be addressed in the future and resolved in order to build a high performance and effective collaboration environment.

Even with the transcoding process, the loading of very large 3D models still took some time. Therefore, one of the future steps is to investigate the possibility of controlling the loading of the 3D geometries on the web browsers. If it is possible to control the loading of 3D models, then it is possible to enhance the time performance of the collaborative web environment by loading only a sub set of the 3D geometries. This way, we will be able to implement loading strategies that control the loading of the 3D model geometries based on the user interaction. For instance, in a case where we have a 3D model for an aircraft, the outer body of the aircraft will be loaded first. Then, if the user zooms in or navigates to a specific part of the model, the geometries of this model will be loaded.

Another future step is to enhance the mini 3D web viewer template. The current version of this viewer allows users to view, hide/show, and to rotate the 3D model and its parts. Adding a new 3D functionality can enhance the 3D web viewer. For example, features such as picking, selecting and attaching information to specific parts, can enhance the web collaboration environment, as well as enrich the users experience using this environment.

Currently, the 3D information such as the hierarchal structure or connection between the parts of the 3D model is stored only in the X3D files, and files hierarchy of the partitioned dataset. Using a database to store this information and other 3D information such as the parts names, parts description, parts analysis results, and engineer's comments will enhance the usability of the collaboration environment, and it will allow the developers to implement dynamic web pages that represent the individual parts or shapes of the 3D model.

The last important open issue here is to determine the solution type that Airbus Group wants to adopt. The transcoding system can be developed in two different ways. The first, is to build the system locally using Airbus resources such as the network and the computers that compose the Hadoop clusters. The alternative to this solution is to develop the transcoding system using Amazon EC2 web service.

Building a local Hadoop cluster is not an easy task and it requires a lot of time to install and setup the cluster. Adding nodes to the cluster also requires a long time, because these installation and setup steps have to be performed in every single node in the cluster as there is not an automated way to do this process. Furthermore, building the cluster and adding nodes to the cluster is very expensive. Even if there is a budget for such resources, the purchase process is a slow process and it may take up to a month. However, the main advantage of local clusters is that we have the full control over all the nodes of the cluster and the data stored on these clusters. These clusters are secure clusters and AirBus Group is the only entity that accesses these clusters.

On the other hand, EC2 clusters are very easy to construct especially if a Hadoop cluster image is created. This image can be used to create new clusters in seconds, using this technology the installation and setup of Hadoop and MapReduce is done only once, which is while preparing the Hadoop image. EC2 web service provides the user with access to a powerful set of computers with cheap prices. Amazon will charge the users for only what they use without a minimum fee. Amazon has rates per hour for using every single virtual machine type.

In this case, the major drawback of EC2 web services is that the 3D models will get stored in Amazon servers rather than on Airbus Group computers. This way, there is a security risk that these 3D models can get stolen or a third party could have access to them. Even if these models are deleted from the servers and the clusters are terminated, Airbus Group does not have any clue whether the data actually gets deleted from the servers or not.

Bibliography

[1] F. N. AFRATI and J. D. ULLMAN, *Optimizing joins in a map-reduce environment*, in EDBT 2010, 13th International Conference on Extending Database Technology, Lausanne, Switzerland, March 22-26, 2010, Proceedings, 2010, pp. 99–110.

[2] E. S. AGENCY, *Adapted Ariane 5 ME. Retrieved on 2014-08-30, from http://www.esa.int/Our_Activities/Launchers/Launch_vehicles/Adapted_Ariane_5_ME*.

[3] T. ALPCAN, C. BAUCKHAGE, and E. KOTSOVINOS, *Towards 3D Internet: Why, What, and How?*, 2013 International Conference on Cyberworlds, 0, 2007, pp. 95–99.

[4] I. AMAZON WEB SERVICES, *Amazon EC2 Product Details. Retrieved on 2014-08-30, from http://aws.amazon.com/ec2/details/*.

[5] A. ANAND, *Scaling Hadoop to 4000 nodes at Yahoo!. Retrieved on 2014-09-14 http://goo.gl/8dRMq*.

[6] L. O. ANURAG GHOSH, *What is 3D Modeling?. Retrieved on 2014-08-30, from http://www.wisegeek.com/what-is-3d-modeling.htm*.

[7] J. BEHR, P. ESCHLER, Y. JUNG, and M. ZÖLLNER, *X3DOM: a DOM-based HTML5/X3D integration model.*, in Web3D, S. N. Spencer, D. W. Fellner, J. Behr, and K. Walczak, eds., ACM, 2009, pp. 127–135.

[8] J. BEHR, Y. JUNG, T. DREVENSEK, and A. ADERHOLD, *Dynamic and interactive aspects of X3DOM*, in 3D Technologies for the World Wide Web, Proceedings of the 16th International Conference on Web 3D Technology, Web3D 2011, Paris, France, June 20-22, 2011, pp. 81–87.

[9] J. BEHR, Y. JUNG, J. KEIL, T. DREVENSEK, M. ZÖLLNER, P. ESCHLER, and D. W. FELLNER, *A scalable architecture for the HTML5/X3D integration model X3DOM*, in 3D technologies for the World Wide Web, Proceedings of the 15th International Conference on Web 3D Technology, Web3D 2010, Los Angeles, California, July 24-25, 2010, pp. 185–194.

[10] V. R. BENJAMINS, J. CONTRERAS, O. CORCHO, and A. GÓMEZ-PÉREZ, *Six Challenges for the Semantic Web*, in In KR2002 Semantic Web Workshop, 2002, p. 2004.

[11] H. S. BLOG, *MapReduce Patterns, Algorithms, and Use Cases. Retrieved on 2014-09-14 http://highlyscalable.wordpress.com/2012/02/01/mapreduce-patterns/*.

[12] T. C. BLOG, *The future of O3D. Retrieved on 2014-09-14 from http://blog.chromium.org/2010/05/future-of-o3d.html*.

[13] S. E. BROWN, *Tesla shows off sedan at Autodesk gallery. Retrieved on 2014-09-09, from http://www.bizjournals.com/sanfrancisco/stories/2010/09/20/daily9.html*.

[14] D. BRUTZMAN and L. DALY, *X3D: Extensible 3D Graphics for Web Authors*, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[15] J. CHANDAR, *Join Algorithms using Map/Reduce*, Master's thesis, University of Edinburgh, School of Informatics, 2010.

[16] C. CHU, S. K. KIM, Y. LIN, Y. YU, G. R. BRADSKI, A. Y. NG, and K. OLUKOTUN, *Map-Reduce for Machine Learning on Multicore*, in Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006, pp. 281–288.

[17] T. W. CONSORTIUM, *What is X3D. Retrieved on 2014-08-30, from http://www.web3d.org/what-x3d*.

[18] T. W. CONSORTIUM, *X3D and VRML, The Most Widely Used 3D Formats. Retrieved on 2014-08-30, from http://www.web3d.org/x3d-vrml-most-widely-used-3d-formats*.

[19] M. H. DEA, CARL, G. GRUNWALD, and S. PHILLIPS, *JavaFX 8: Introduction by Example*, 2014.

[20] J. DEAN and S. GHEMAWAT, *MapReduce: simplified data processing on large clusters*, Commun. ACM, 51(1), 2008, pp. 107–113.

[21] J. DEAN and S. GHEMAWAT, *MapReduce: a flexible data processing tool*, Commun. ACM, 53(1), 2010, pp. 72–77.

[22] J. DEJUN, G. PIERRE, and C.-H. CHI, *EC2 Performance Analysis for Resource Provisioning of Service-Oriented Applications*, in Proceedings of the 3rd Workshop on Non-Functional Properties and SLA Management in Service-Oriented Computing, nov 2009.

[23] C. W. EVERITT and M. J. KILGARD, *Practical and Robust Stenciled Shadow Volumes for Hardware-Accelerated Rendering*, CoRR, cs.GR/0301002, 2003.

[24] EXIST DB.COM, *Learning XQuery and eXist-db. Retrieved on 2014-09-09, from http://exist-db.org/exist/apps/doc/learning-xquery.xml*.

[25] A. S. FOUNDATION, *HDFS Architecture Retrieved on 2014-09-14 http://hadoop.apache.org/docs/r2.2.0/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html*.

[26] A. S. FOUNDATION, *Welcome to Apache Hadoop Retrieved on 2014-09-14 http://hadoop.apache.org/*.

[27] T. GLANDER, A. MORENO, M. ARISTIZÁBAL, J. CONGOTE, J. POSADA, A. GARCÍA-ALONSO, and O. E. RUIZ, *ReWeb3D: enabling desktop 3D applications to run in the web*, in Web3D13, 2013, pp. 147–155.

[28] GOOGLE, *O3D has changed . Retrieved on 2014-08-30, from https://code.google.com/p/o3d/*.

[29] J. GRAAUMANS, *Usability of XML Query Languages*.

[30] O. C. GRAPHICS, *PolyTrans—CAD+DCC Pro Translation System. Retrieved on 2014-08-30, from http://www.okino.com/conv/conv.htm*.

[31] W. HADOOP, *Hadoop wiki - powered by. Retrieved on 2014-09-14 https://wiki.apache.org/hadoop/PoweredBy*.

[32] A. INC, *3D Modeling and Rendering Software — 3ds Max — Autodesk. Retrieved on 2014-09-09, from http://www.autodesk.com/products/3ds-max/overview*.

[33] A. INC, *Compare 3ds Max 2015 vs 3ds Max 2014 or 2013 — Autodesk. Retrieved on 2014-09-09, from http://www.autodesk.com/products/3ds-max/compare/compare*.

[34] A. INC, *Maya 2015 vs. 2014 and previous versions. Retrieved on 2014-09-09, from http://www.autodesk.com/products/maya/compare/compare-releases*.

[35] H. KASIM, V. MARCH, R. ZHANG, and S. SEE, *Survey on Parallel Programming Model*, in Proceedings of the IFIP International Conference on Network and Parallel Computing, NPC '08, Berlin, Heidelberg, 2008, Springer-Verlag, pp. 266–275.

[36] K. LEE, Y. LEE, H. CHOI, Y. D. CHUNG, and B. MOON, *Parallel data processing with MapReduce: a survey*, SIGMOD Record, 40(4), 2011, pp. 11–20.

[37] S. LEO and G. ZANETTI, *Pydoop: a Python MapReduce and HDFS API for Hadoop*, in Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC '10, New York, NY, USA, 2010, ACM, pp. 819–825.

[38] J. LIN and C. DYER, *Data-Intensive Text Processing with MapReduce*, Synthesis Lectures on Human Language Technologies, Morgan & Claypool Publishers, 2010.

[39] A. MAGAZINE, *BIM and the Freedom Tower 2*.

[40] D. MINER and A. SHOOK, *MapReduce Design Patterns : [building effective algorithms and analytics for Hadoop and other systems]*, O'Reilly, Beijing, Köln, u.a., 2013. DEBSZ.

[41] MOZILLA, *Canvas:3D. Retrieved on 2014-09-14 https://wiki.mozilla.org/Canvas:3D*.

[42] L. D. PAULSON, *Building rich web applications with Ajax*, Computer, 38(10), Oct. 2005, pp. 14–17.

[43] N. E. PIUS, L. QIN, F. YANG, and Z. H. MING, *Optimizing Hadoop Block Placement Policy and Cluster Blocks Distribution*, 6(10), 2012, pp. 1220 – 1226.

[44] POWERCATIA, *About CATIA. Retrieved on 2014-09-09, from http://www.powercatia.com/Pages/aboutcatia.aspx*.

[45] S. RÖTTGER, M. KRAUS, and T. ERTL, *Hardware-accelerated volume and isosurface rendering based on cell-projection*, in IEEE Visualization, 2000, pp. 109–116.

[46] S. Seo, E. J. Yoon, J. Kim, S. Jin, J. Kim, and S. Maeng, *HAMA: An Efficient Matrix Computation with the MapReduce Framework*, in Cloud Computing, Second International Conference, CloudCom 2010, November 30 - December 3, 2010, Indianapolis, Indiana, USA, Proceedings, 2010, pp. 721–726.

[47] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, *The Hadoop Distributed File System*, in Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST), MSST '10, Washington, DC, USA, 2010, IEEE Computer Society, pp. 1–10.

[48] M. Solutions, *CATIA. Retrieved on 2014-08-30, from http://www.mecanicasolutions.com/index.php/plm-product-services/catia/.*

[49] G. Stockman and L. G. Shapiro, *Computer Vision*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st ed., 2001.

[50] Technia, *CATIA Highlights and benefits. Retrieved on 2014-09-09, from http://www.technia.com/cad/catia.*

[51] Techopedia, *What is AutoCAD? - Definition from Techopedia. Retrieved on 2014-09-09, from http://www.techopedia.com/definition/6080/autocad.*

[52] J. Tordable, *MapReduce for Integer Factorization*, CoRR, abs/1001.0421, 2010.

[53] T. W. W. W. C. W3C, *Extensible Markup Language (XML) 1.0 Retrieved on 2014-09-14 http://www.w3.org/TR/REC-xml/.*

[54] T. W. W. W. C. W3C, *XQuery 1.0: An XML Query Language Retrieved on 2014-09-14 from http://www.w3.org/TR/xquery/.*

[55] T. White, *Hadoop - The Definitive Guide: Storage and Analysis at Internet Scale (3. ed., revised and updated)*, O'Reilly, 2012.

[56] Wikipedia, *Elements of mesh modeling. Retrieved on 2014-09-10, from http://en.wikipedia.org/wiki/Polygon-mesh.*

[57] WisegGeek, *What Is AutoCAD?. Retrieved on 2014-09-09, from http://www.wisegeek.org/what-is-autocad.htm.*

[58] www.khronos.org, *OpenGL ES 2.0 for the Web. Retrieved on 2014-09-14 http://www.khronos.org/webgl/.*

[59] X3DOM, *Welcome to X3DOM Retrieved on 2014-09-14 from http://x3dom.org/download/1.4/docs/singlehtml/.*

[60] x3dom.org, *About X3DOM. Retrieved on 2014-08-30, from http://www.x3dom.org/?page_id=2.*

[61] H. Yang, A. Dasdan, R. Hsiao, and D. S. P. Jr., *Map-reduce-merge: simplified relational data processing on large clusters*, in Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007, 2007, pp. 1029–1040.

[62] E. YARES, *AUTOCAD'S ANCESTOR. Retrieved on 2014-09-09, from http://www.3dcadworld.com/autocads-ancestor/.*