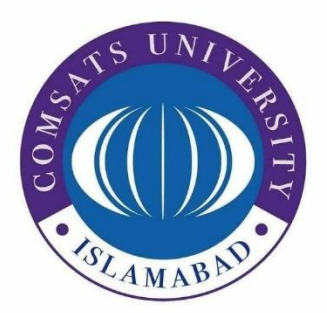
Thermal-Aware Data Center Work Load Simulator

Final Year Project

Session 2016-2020

A project submitted in partial fulfilment of the
COMSATS University Degree
of
BS in Computer Science (CUI)



Department of Computer Science
COMSATS University Islamabad, Lahore Campus

31 July 2020

Project Detail

Type (Nature of project)		[ ] **D**evelopment [ ] **R**esearch [✓] **R&D**		
Area of specialization		Distributed Computing, Cloud Computing		
Project Group Members				
Sr.#	Reg. #	Student Name	Email ID	*Signature
(i)	FA16-BCS-012	Usama Tahir	usamatahir00004@gmail.com	UT
(ii)	FA16-BCS-157	Bilal Sohail	bilalsohail000@gmail.com	BS

*The candidates confirm that the work submitted is their own and appropriate credit has been given where reference has been made to work of others

Plagiarism Free Certificate

This is to certify that, I am **Usama Tahir** S/o **Mirza Tahir Mehmood**, group leader of FYP under registration no CIIT/FA16-BCS-012/LHR at Computer Science Department, COMSATS Institute of Information Technology, Lahore. I declare that my FYP proposal is checked by my supervisor and the similarity index is ___18___% that is less than 20%, an acceptable limit by HEC. Report is attached herewith as Appendix A.

Date: 31 July, 2020 Name of Group Leader: Usama Tahir_ Signature: _____UT_____

Name of Supervisor: _Dr. Muhammad Hasan Jamal__ Co-Supervisor (if any):_Dr. Muhammad Tayyab Chaudhary

Designation: __Assistant Professor____ Designation: _____Assistant Professor ___

Signature: Hasan Jamal _____ Signature: _____MTC_____

HoD: _____

Signature: _____

Abstract

Servers are running around the clock that generates huge amount of heat and in out-turn it arises the temperature in datacenters. High temperature effects the other components like racks, power supply and cabling system in the datacenters. Cooling system are being used to control the temperature of datacenters that are very much costly. We introduced **thermal aware load balancing policies** in datacenters, which will keep the track of heat generated by servers and manage the load of data in such a way that servers will generate minimum amount of heat. It would not only keep the temperature of datacenters in control but also **reduces the cost** of coolants and cooling systems. It will keep the track for amount of heat generated by each server through visual representation using visualization tool. This would help to implement and choose the load balancing policies in datacenters for work load according to the temperature measures that best suits the respective datacenter.

Acknowledgement

*We are very much thankful to our supervisor, **Dr. M. Hasan Jamal** who gave us an opportunity to work on a research-based project. It is only through his guidance that we have done research work and introduced Thermal-Aware load balancing policies for Datacenters. This project helped us to learn new tools and technologies, to research and to enhance our skills which we have learned throughout our Alma mater. We are very grateful for having his expertise on our side that helped us to achieve this milestone.*

- *(**Usama Tahir, Bilal Sohail**)*

Table of Content

List of Tables

List of Figures

1 Introduction

1.1 Introduction

In the revolutionize world of Technology, giants of IT are shifting their data in bulk on servers which is the most essential asset for any organization. This arise the big challenge of maintenance of servers, and the most important of them is to control the temperature of data centers and the heat producing by the servers. Researches are trying to devise load balancing policies to efficiently manage the data coming in data centers. Many load balancing policies are being introduced up till now but none of them are thermal aware. Cooling system were introduced and are still in use to keep the datacenters cool, but this solution is too costly.

Considering the current problem of controlling the heat generated by the datacenters, we are proposing a **thermal aware load balancing policy** in datacenters which will keep the track of heat generated by servers and manage the load of data in such a way that servers will generate minimum amount of heat. Thus, it will reduce the cost spent on cooling systems and coolants. In our proposed system user will enter the data and details of virtual machine configurations (time, utilization, size and cores), server configurations (capacity, availability, memory, and cores), rack configurations (capacity, availability).

After putting the respective details, user will select the algorithm technique which he wants to apply on his data set and will press start simulation, data will be loaded to the java program. This program has the optimize algorithms techniques (e.g. Round-Robin) that would divide the data on servers in best efficient way according to the technique which the user selects.

We are using Kibana (open-source visualization tool for exploring logs). In Kibana we will apply different aggregation methods and it will generate the graph showing the **Average CPU Utilization** and the amount of **heat generated** by each server and how much **data load** on each server. This will help the user to have clear image of which technique should be applied on data so the load on each server would be managed efficiently which in result will generate minimum heat. This will help to reduce the energy and power consumption and the cost spent on the cooling systems.

1.2 Working of Simulator

Our Project has three phases:

1. Pre-processing of data
2. Simulation of Processed data
3. Visualization of data

1.2.1 Pre-processing

The data we are using is of "Ali baba cloud traces of data centers". This data is enormous in size (200 GB). We are using toy data of 5% of whole data through the script which is being written in java to fetch desired data in our simulation as, if we will use all data it will take days for only single simulation.

For error handling in pre-processing, we have also write a script in java to repair our data set so, that it doesn't create any errors during simulation. There are total 4 java codes running in the back end that are responsible for preparing data for the simulation.

1.2.2 Simulation of Processed Data

JAVA Simulator: Simulator GUI have two buttons Vm-configuration and server-configuration that will open a file selector so, user could locate the path of container Meta and machine Meta files from his system. Once the files will be selected, their paths would be stored in a variables and would read desired data from those selected files. From the main GUI user will select the following:

- Select total number of racks in a data center by using selection list (Select no of Racks)
- Set machines per rack by using (Machines per rack)
- From list of task dividing policy user will choose policy which he wants to implement on his data. So far we have implemented these policies:

Now, '**Start Simulation**' button would be used to start simulation. As soon as the user will press this button, it will trigger the java code and ELK stack in parallel.

1.2.2.1 *Simulation Techniques*

Round Robin across Machine: In this policy machines are assigned tasks sequentially. Each task has associated start time and end time with it. When time of simulation reaches at the start time machine is assigned with a particular task and when simulation reaches at the time stamp equals to end time of task then machine would be marked free and thus, the utilization decreases.

Round robin across rack: In this policy machines are assigned tasks sequentially. Initially, tasks will be assigned across different racks. Each task has associated start time and end time with it. When time of simulation reaches at the start time machine is assigned with a particular task and when simulation reaches at the time stamp equals to end time of task then machine would be marked free and thus, the utilization decreases.

Random: In this policy machines are assigned tasks randomly.

MinHR: In this policy, the servers are sorted in ascending order of their temperatures and the VMs are placed in round robin fashion from coolest to hottest servers, with cap on utilization level of 62.5% and 75%. This policy simulates the heat recirculating.

MaxUtilization: In this policy, VMs are placed on servers with maximum average utilization.

MinUtilization: In this policy, VMs are placed on servers with minimum average utilization.

TASA: In this policy the hottest VMs are allocated to the coolest servers, with and without backfilling.

HAWDA: This is the policy proposed in this project and is based on hotspot aware workload scheduling. Using thermal profiles of servers, the outlet temperature is predicted and the VM is placed on the server with the least predicted temperature.

1.2.3 Visualization [1]

Before going to the visualization part, we want to introduce the term **ELK Stack**. This acronym is describing the three open-source projects comprised of **Elasticsearch**, **Logstash** and **Kibana**. We are using this to aggregate and analyze the log files and to generate the visualization of analytics of CPU utilization and amount of heat generated by the servers.

Elasticsearch: It is an open-source distributed search engine that is built on the Apache Lucene.

Logstash: It is also an open-source tool that allows you to gather data from different sources, and send it to your desired location.

Kibana: It is an open-source visualization tool for reviewing and logs and to generate analytics of it in form of graphs and pie-chart.

Now, here comes the visualization part, in parallel with the Simulation, ELK would be running in background. Logstash will load the data into the Elasticsearch, now user could view the analytics by pressing '**Visualize**' button that will open the interface of Kibana where, user would be able to apply different aggregation operations on results and could be able to visualize results in many different formats.

1.3 Motivation

Companies are bearing too much cost for the cooling systems and facing difficulty to maintain it. Our motivation is to provide thermal aware Simulator using which the optimized algorithms can be proposed with the efficient way to divide tasks in the datacenters. So, the servers would use the minimum energy and generate less heat. We have our motivation to build a complete product for the datacenters so, they could also be able to keep the analytical record of their server utilization and to determine which policies best suits for their datacenters.

1.4 Scope

This application can be used by datacenter administrators to simulate their data based on results. They can predict the results with data that they are going to work. Secondly this project will be open source so that anyone who want to implement something new can use it and modify it for achieving their goals. It will also generate the analytical graphs and pie chart and will give the precise behavior of temperature variation on machine, while using different policies.

1.5 Objective

The main objective of this project is to provide a simulator that could simulate the datacenter in efficient way possible. This simulator will keep track of datacenter components which are overloaded and generating heat more than normal values. The simulator will provide policies to data center administrators through which they can forecast their cost and total energy consumption of datacenters. The simulator would be extensive and would be opensource anyone can use it for their research purposes means they can see the implementation of their ideas by just extending the source code of this simulator. The simulator would provide the interactive interface to users.

1.6 Constraints

For our simulator to work efficiently, following are some constraints that are necessary:

- Constraint is applied on file selector so that user can only select files having **.csv** extensions
- If the respective fields doesn't exist in the dataset, then user will not be able to perform simulation. (E.g. If the ID and no. of containers etc. are missing in the machine configuration file then user will not be able to perform further actions).
- We have applied exception handlers on this button for example "file not found exception" so, simulation will not start until the user would not give the correct path of file.
- Users have to load the comma separated file.

1.7 Problem Statement

Due to modern trend of using cloud services load on datacenters in increasing day by day major issues which are being faced by datacenter administrators are Cost on cooling Systems and Servers malfunctioning due to overheating. Our simulator will address these issues efficiently by predicting the total cost and states of datacenter components with the similar data that are going to be placed in that datacenter.

2 Requirements Analysis

2.1 Related Work [2]

2.1.1 CloudSim

CloudSim is extensible cloud simulation tool kit. It is developed by Computer Science department of University of Melbourne, Australia. It enables the users to create datacenter entities and provide relations between them. CloudSim source code contains basic classes for deriving data centers, virtual machines, users and policies for managing diverse parts of the system like scheduling a provisioning. The main features provided by CloudSim are that it can be used for simulation of large-scale Cloud computing data centers and it allow users to define their own policies for allocating hosts to virtual machines. Support the use of Energy aware Computational resources and support the creation of various data center network topologies. It also supports the simulation of a federated cloud environment. Cloudsim is not thermal-aware and it is not user friendly.

2.1.2 Cloud Analyst

Cloud Analyst is a tool developed by R. Buyya et. al at Computer Science Department of University of Melbourne Australia. Cloud Analyst provide Graphical User Interface and it is based on CloudSim. It provides some additional features. It also considers geographical location of datacenters and user bases and perform the simulations accordingly. Its most interesting feature is that it allows users to store their simulation configurations as XML files or even user can export their simulation results as PDF documents. It has the graphical user interface for setting up configurations and viewing the results. It supports high degree of configuration flexibility and allow users to save configurations as xml file, it allows users to export results in pdf files and it provide output in graphical forms which are easily understandable. Cloud Analyst is not thermal-aware.

2.1.3 DCSim (Data Center Simulation)

DCSim is datacenter simulator designed in java. It is extensible. It provides stable and easy interface for performing experiments on datacenters. DCSim is an event driven simulator. DCSim provide the capability of modeling replicated virtual machines. The main features provided by DCSim are that it is based on the multi-tire application model and allows the simulation of dependencies between Virtual Machines. It generates the results based on predefined data center management policies. DCSIM has loss of control and it lacks thermal awareness.

2.1.4 Green Cloud

Green Cloud is an energy aware data center simulator. It also calculates the energy consumed by datacenter components like servers, switches and other equipment's. Green cloud is an open source and it has user friendly interface, it focuses mainly on energy consumption by datacenter components and support researchers to deploy their policies in this simulator and see how the results varies. It focusses mainly on cloud networks. Green Cloud is also not thermal aware.

2.1.5 EMUSIM

This combines emulation and simulation to enable more accurate models and results of simulation. It stands for Integrated Emulation and Simulation. EMUSIM is used to evaluate cloud applications after varying resources and patterns of request on cloud applications. It supports CPU intensive applications that aims to reduce the cost of running datacenter accurately and models the application to supply information regarding performance.

2.2 Stakeholders

Stakeholders of the system are those people who are concerned with the system. They are usually those who invest their time and money for the system, and they try to make the system as better as it can be. These are the following stakeholders of our project:

Table 1: Stakeholders

Actor	Domain Knowledge	Computer Knowledge	Frequency of Use
Users	Should know how the Servers work.	know how to interact with Simulators.	Very frequent
Developers	Should have a complete knowledge of technical aspects of how servers runs and the engineering behind it.	Should have complete grip on programming and know how to use code editors and software that are being used in the development of simulator.	Frequent
Supervisors	Know the theory behind the working of servers.	Should have good grip on computer and different software.	Occasionally

2.2.1 User

It would be open source. So, anyone who want to download it from internet can do it. User gets a complete Software with all the functionalities and user will have options to customize it by implementing their own policies.

2.2.2 Developers

Developers are the most important stakeholders in any project. They invest their time and develop the system and combine all the modules and convert it into the final working product.

2.2.3 Supervisors

Supervisors are the faculty members of university who are willing to help students throughout this project. They are already specialists of the field, so they are there to address all the issues faced by developers during the development phase. It also points out those issues that user might face while using this software so that development team resolve those issues before launching the product.

2.3 Requirement Elicitation

2.3.1 Functional Requirements

2.3.1.1 *FR1: Simulator*

Table 2: Functional requirements for Simulator

Requirement Number	Description
FR1-1	User should have jdk installed on their system.
FR1-2	User need to download simulator.
FR1-3	User must have respective Input files to run on Simulator.

2.3.1.2 *FR2: ELK Stack Installation*

Table 3: Functional requirements for ELK Stack Installation

Requirement Number	Description
FR2-1	User have to install open-source Stack that includes Elasticsearck, Logstash And Kibana.
FR2-2	User have to setup environment for installation of ELK Stack.

2.3.1.3 *FR3: Java GUI*

Table 4: Functional requirements for Java GUI

Requirement Number	Description
FR3-1	User will see the multiple option in the start of interface where ha has to start from INPUT Files section.
FR3-2	User will have to select Input Files and have to name the Output files.
FR3-3	Setting of Default values and Data center settings
FR3-4	Simulate to Visualize

2.3.1.4 *FR4: Selecting Input Files*

Table 5: Functional requirements for selecting Input Files

Requirement Number	Description
FR4-1	User must have to locate Input file of Machine meta from his system
FR4-2	User must have to locate Input file of container meta from his system
FR4-3	User must have to locate Input file of batch task from his system.

2.3.1.5 *FR5: Output Files*

Table 6: Functional requirements for output files

Requirement Number	Description
FR5-1	User must have to name Output file of Machine usage on his system.
FR5-2	User then have to name Output file of Container usage on his system.
FR5-3	User must have to name Output file of Thermal Data on his system.

2.3.1.6 *FR6: Setting default values*

Table 7: Functional requirements for setting default values

Requirement Number	Description
FR6-1	After loading all the files user must to set the default values of Input files.

2.3.1.7 *FR7: Datacenters and Load balancing Settings*

Table 8: Functional requirements for datacenters and load balancing settings

Requirement Number	Description
FR7-1	User have to set the datacenter settings that includes racks and machine per racks
FR7-2	User have to set the load balancing settings that includes balancing policies for containers
FR7-3	User then selects the machine type.

2.3.1.8 *FR8: Simulation*

Table 9: Functional requirements for Simulation

Requirement Number	Description
FR8-1	User will press the simulate button and the ELK will start running in background
FR8-2	User could apply different aggregate functions to see the graphs

2.3.2 Non Functional Requirements

2.3.2.1 *NF1: Performance*

Table 10: Nonfunctional requirements for performance

Requirement Number	Description
NF1-1	Performance of system should not be decreased at any moment
NF1-2	As the size of data is enormous the operations are meant to be fast for which logstash and Elasticsearch are used

2.3.2.2 *NF2: Scalability*

Table 11: Nonfunctional requirements for scalability

Requirement Number	Description
NF2-1	The system should be scalable, and its performance should not be reduced in any scenario for which code is written efficiently and having less complexity in it.

2.3.2.3 *NF3: Capacity*

Table 12: Nonfunctional requirements for capacity

Requirement Number	Description
NF3-1	This simulator should be able to work with enormous data. Because data logs from data centers are very big in size, for handling it logstash will load data into elastic search that will increase its performance.

2.3.2.4 *NF4: Security*

Table 13: Nonfunctional requirements for security

Requirement Number	Description
NF4-1	As the tool is open source and the user could perform all task on his own, without having concern of his Data on risk
NF4-2	The results of simulation should only be displayed to that user and the result would not be stored by any other third party source, user could generate the pdf of his results and then can dump his data from the tool.

2.3.2.5 *NF5: Extendibility*

Table 14: Nonfunctional requirements for extendibility

Requirement Number	Description
NF5-1	The Source code would be extendable. User should add his own load balancing policies for research purposes.

2.3.2.6 *NF6: Availability*

Table 15: Nonfunctional requirements for availability

Requirement Number	Description
NF6-1	All the functions would be available for user and should be easily accessible
NF6-2	Results of simulation would be available to compare with the result of other simulation

2.4 Use Case

2.4.1 Start

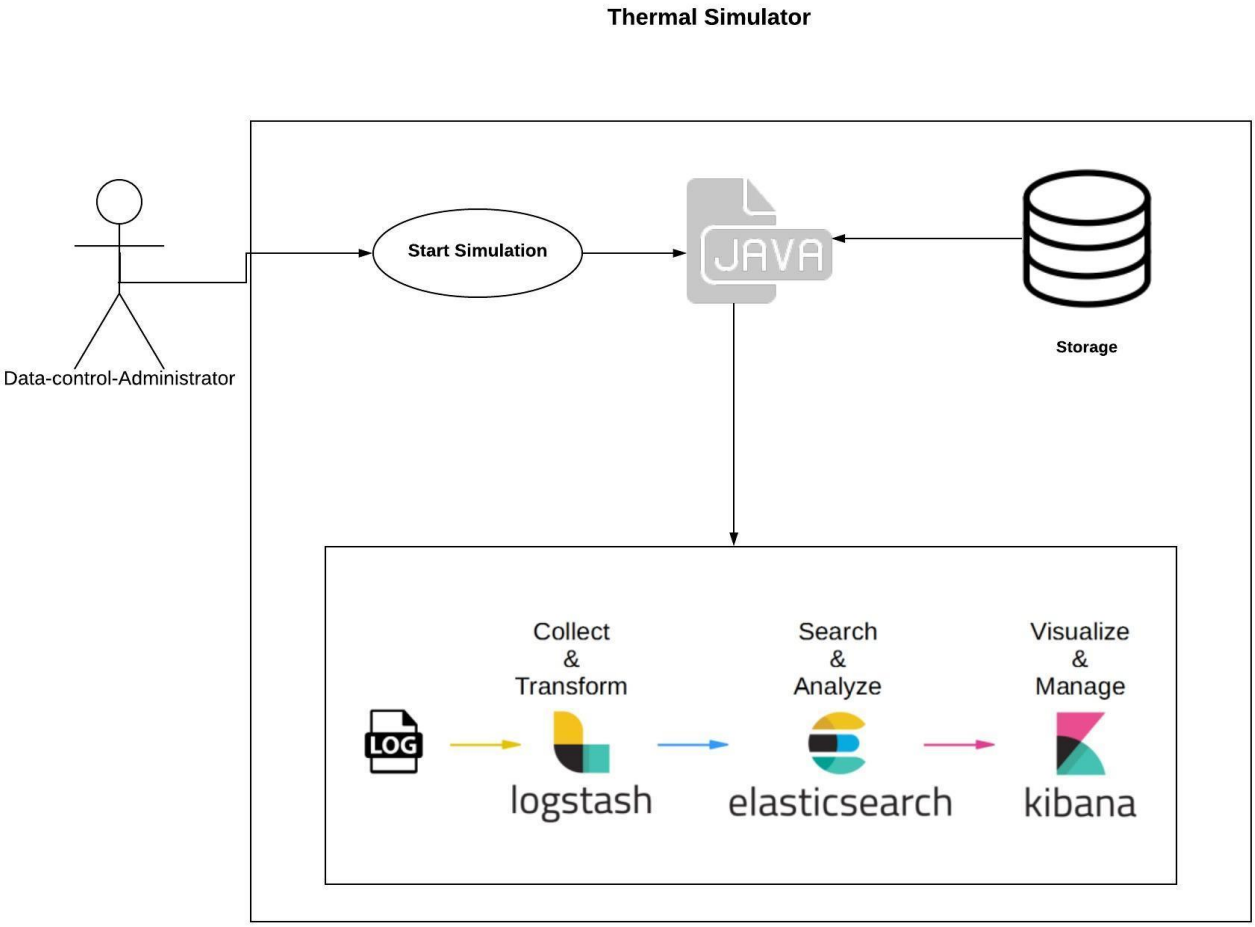


Figure 1: Start Use Case

2.4.2 Upload

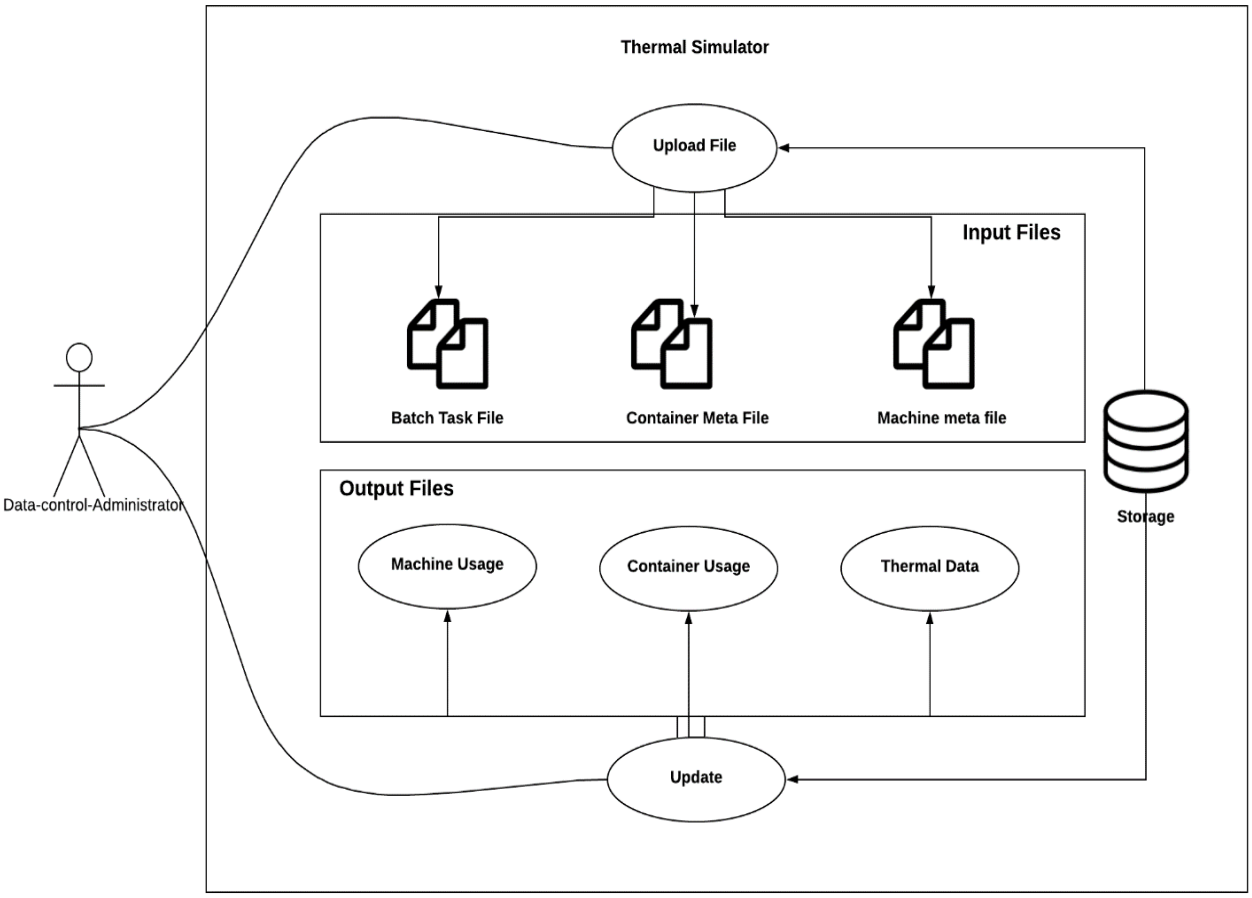


Figure 2: Upload Use Case

2.4.3 Delete

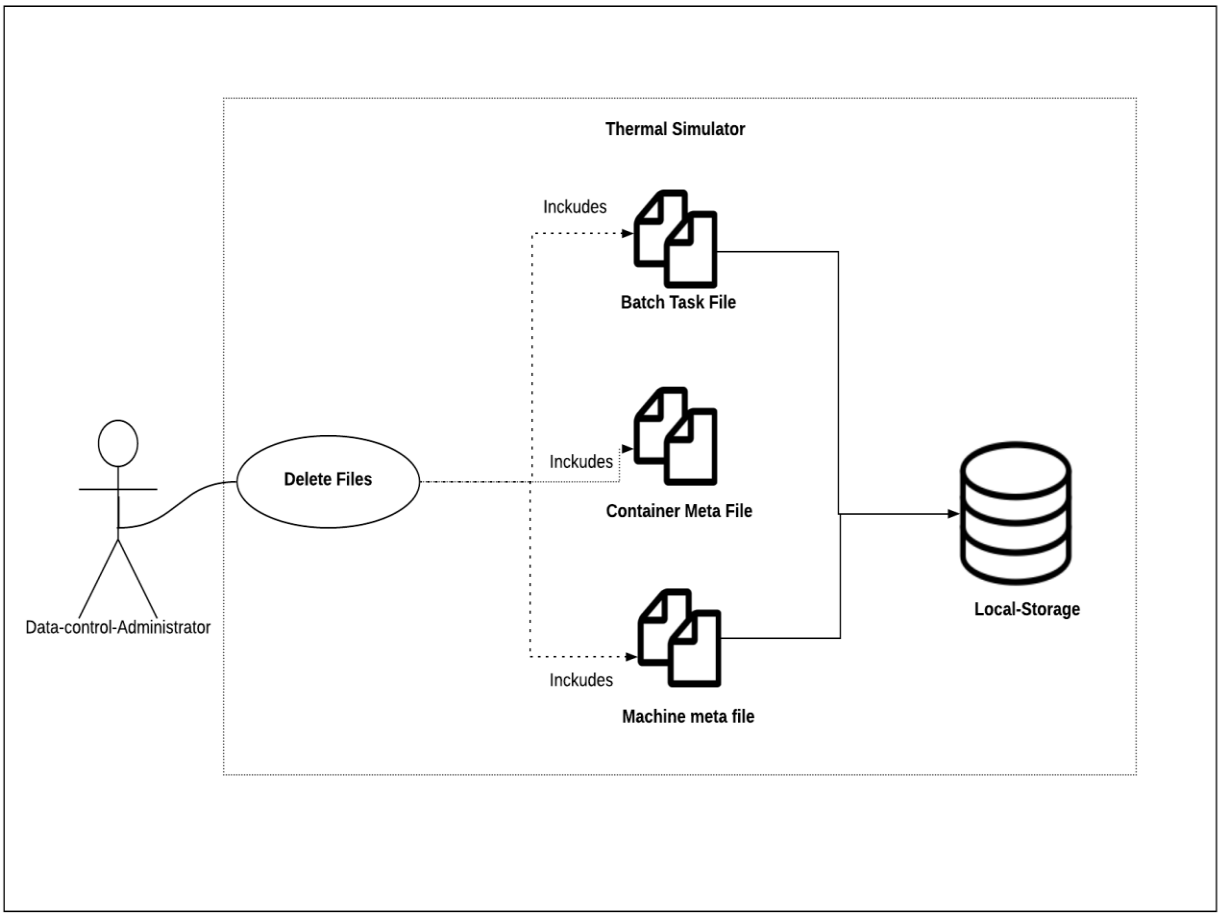


Figure 3: Delete Use Case

2.4.4 Select

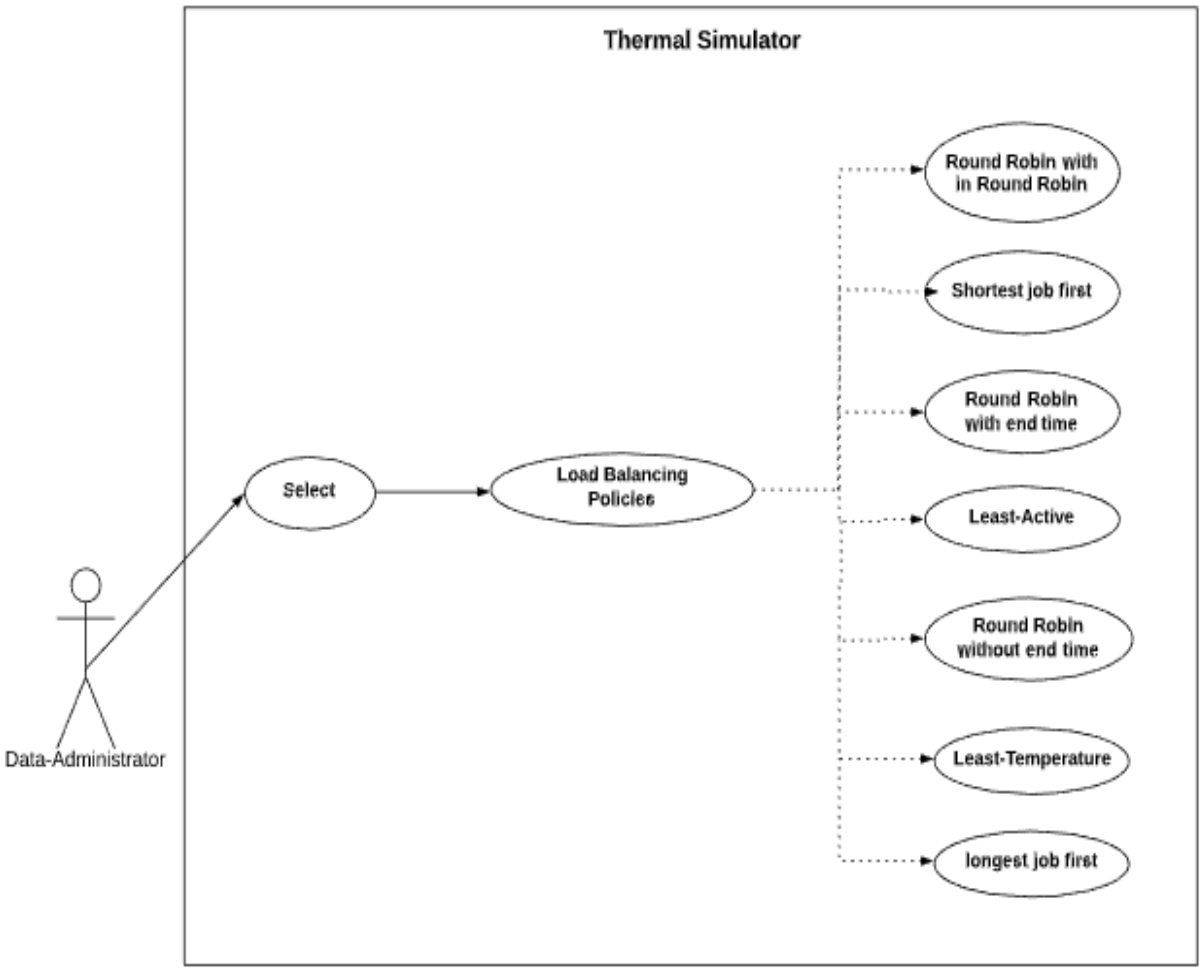


Figure 4: Select Use Case

2.5 Use Cases Description

UC01

Table 16: Input Files- Machine Meta

Use Case ID	UC01
Use Case Name	Machine Meta
Actor (s)	User
Summary	A user will load Machine Meta file.
Pre-conditions	
The .csv file should have these Fields (Machine Id, Time Stamp)	
Post Conditions	
Success	Machine meta file will be successfully loaded.
Failure	File doesn't contain respective fields, file would not be loaded.
Basic Flow	
The use case starts when a user starts the simulator and would select the Input machine meta file to load from his local storage. 1. User will locate machine Meta file. 2. File should contain machine id and time stamp. 3. The simulator verifies and validates the file, and will allow user to upload next Input File. 4. The use case ends.	
Alternate Flows	
1. User selects the 'next' after uploading first file. 2. The simulator prompts the user about re-entering invalid or incomplete requirement. 3. The Basic Flow continues at (2).	
Extends	
None	

UC02

Table 17: Input Files- Container Meta

Use Case ID	UC02
Use Case Name	Container Meta
Actor (s)	User
Summary	A user will load Container Meta file.
Pre-conditions	
The .csv file should have these Fields (Container Id, Time Stamp)	
Post Conditions	
Success	Container meta file will be successfully loaded.
Failure	File doesn't contain respective fields, file would not be loaded.
Basic Flow	
The use case starts when user selects the Input Container meta file to load from his local storage. 1. User will locate Container Meta file. 2. File should contain container id and time stamp. 3. The simulator verifies and validates the file, and will allow user to upload next Input File. 4. The use case ends.	
Alternate Flows	
1. User selects the 'next' after uploading second file. 2. The simulator prompts the user about re-entering invalid or incomplete requirement. 3. The Basic Flow continues at (2).	
Extends	
None	

UC03

Table 18: Input Files- Batch Task

Use Case ID	UC03
Use Case Name	Batch Task
Actor (s)	User
Summary	A user will load Batch Task file.
Pre-conditions	
The .csv file should have these Fields (Task Id, Time Stamp)	
Post Conditions	
Success	Batch task file will be successfully loaded.
Failure	File doesn't contain respective fields, file would not be loaded.
Basic Flow	
The use case starts when a user selects the Input machine meta file to load from his local storage. 1. User will locate batch task file. 2. File should contain task id and time stamp. 3. The simulator verifies and validates the file, and will allow user to upload next Input File. 4. The use case ends.	
Alternate Flows	
1. User selects the 'next' after uploading third input file. 2. The simulator prompts the user about re-entering invalid or incomplete requirement. 3. The Basic Flow continues at (2).	
Extends	
None	

UC04

Table 19: Output Files- Machine Usage

Use Case ID	UC04
Use Case Name	Machine Usage
Actor (s)	User
Summary	A user will create Machine Usage file.
Pre-conditions	
The location to create .csv file should be selected.	
Post Conditions	
Success	Empty Machine usage file will be successfully created.
Failure	File is not created due to improper name conventions.
Basic Flow	
The use case starts when a user will create an empty machine usage file on his local storage. 1. User enters the name of machine usage file. 2. User will enter a name of index to visualize result in kibana. 3. The simulator verifies and validates the file name and type, and will allow user to upload next Output file. 4. The use case ends.	
Alternate Flows	
1. User selects the ‘next’ after uploading first output file. 2. The Basic Flow continues at (2).	
Extends	
None	

UC05

Table 20: Output Files- Container Usage

Use Case ID	UC05
Use Case Name	Container Usage
Actor (s)	User
Summary	A user will create Container Usage file.
Pre-conditions	
The location to create .csv file should be selected.	
Post Conditions	
Success	Container usage file is successfully created.
Failure	File isn't created due to wrong convention of file type or name.
Basic Flow	
The use case starts when a user will create an empty Container usage file on his local storage. 1. User will enter the name of container usage file. 2. User will enter a name of index to visualize result in kibana. 3. The simulator verifies and validates the file name and type, and will allow user to upload next Output file. 4. The use case ends.	
Alternate Flows	
1. User selects the 'next' after creating second output file. 2. The Basic Flow continues at (2).	
Extends	
None	

UC06

Table 21: Output Files- Thermal data

Use Case ID	UC06
Use Case Name	Thermal Data
Actor (s)	User
Summary	A user will create Thermal Data file.
Pre-conditions	
The location to create ThermalData.csv file should be selected.	
Post Conditions	
Success	Empty Thermal Data file will be successfully created.
Failure	File is not created due to improper name conventions or type.
Basic Flow	
The use case starts when a user will create an empty Thermal Data file on his local storage. 1. User enters the name of Thermal Data file. 2. User will enter a name of index to visualize result in kibana. 3. The simulator verifies and validates the file name and type, and will allow user to upload next Output file. 4. The use case ends.	
Alternate Flows	
1. User selects the ‘next’ after uploading third output file. 2. The Basic Flow continues at (2).	
Extends	
None	

UC07

Table 22: Default settings- Machine Usage

Use Case ID	UC07
Use Case Name	Default values of Machine Usage
Actor (s)	User
Summary	A user will enter default values of Machine Usage file.
Pre-conditions	
The machine usage file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User set the default values of machine usage or it already exist.
Failure	User sets all the values to null, or leave any field empty.
Basic Flow	
The use case starts when a user will set the default values of machine usage file. 1. User selects the machine type either A or B. 2. User enter the total CPU cores. 3. User enter total memory size. 4. User enter inlet temperature. 5. User selects machine status.	
Alternate Flows	
1. Default values are already exist. 2. User selects the ‘next’ after setting default values of machine usage.	
Extends	
None	

UC08

Table 23: Default settings- container Meta

Use Case ID	UC08
Use Case Name	Default values of Container meta
Actor (s)	User
Summary	A user will enter default values of Container meta file.
Pre-conditions	
The container meta file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User set the default values of container meta or it already exist.
Failure	User sets all the values to null, or leave any field empty.
Basic Flow	
The use case starts when a user will set the default values of container meta file. 1. User enters the total CPU required. 2. User enters the total memory required.	
Alternate Flows	
1. Default values are already exist. 2. User selects the ‘next’ after setting default values of container Meta.	
Extends	
None	

UC09

Table 24: default settings- Batch task

Use Case ID	UC09
Use Case Name	Default values of Batch Task
Actor (s)	User
Summary	A user will enter default values of Batch task file.
Pre-conditions	
The batch task file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User set the default values of batch task or it already exist.
Failure	User sets all the values to null, or leave any field empty.
Basic Flow	
The use case starts when a user will set the default values of machine usage file. 1. User enters the total CPU required. 2. User enter the total memory required. 3. User enter total instances.	
Alternate Flows	
1. Default values are already exist. 2. User selects the ‘next’ after setting default values of Batch task.	
Extends	
None	

UC10

Table 25: Datacenter settings

Use Case ID	UC10
Use Case Name	Datacenter Settings
Actor (s)	User
Summary	A user will set the datacenter settings.
Pre-conditions	
The machine usage file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User enters the datacenter settings.
Failure	User sets all the values to null, or leave any field empty.
Basic Flow	
The use case starts when a user will enter the settings of datacenter. 1. User enters the no. of Racks in datacenter. 2. User enter the Machines per rack. 3. User enter the total type of machines.	
Alternate Flows	
1. Default values are already exist. 2. User selects the ‘next’ after setting default values of machine usage.	
Extends	
None	

UC11

Table 26: load balancer settings

Use Case ID	UC11
Use Case Name	Load Balancer Settings
Actor (s)	User
Summary	A user will enter Load Balancer settings.
Pre-conditions	
The machine usage file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User enters all the load balancer settings.
Failure	User sets all the values to null, or leave any field empty.
Basic Flow	
The use case starts when a user will set the load balancer settings. 1. User selects the balancing policy for containers. 2. User selects the balancing policy for tasks.	
Alternate Flows	
1. Default values are already exist. 2. User selects the ‘next’ after setting default values of machine usage.	
Extends	
None	

UC12

Table 27: thermal profile settings

Use Case ID	UC12
Use Case Name	Thermal Profile
Actor (s)	User
Summary	A user selects the thermal profile.
Pre-conditions	
The machine usage file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User set the thermal profile either as machine type ‘A’ or ’B’.
Failure	User leaves the fields of increase in temp of CPU utilization as empty.
Basic Flow	
The use case starts when a user selects the thermal profile. 1. User selects the thermal profile either A or B. 2. User enter the increase in temperature at 0% CPU utilization. 3. User enter the increase in temperature at 12% CPU utilization. 4. User enter the increase in temperature at 25% CPU utilization. 5. User enter the increase in temperature at 37% CPU utilization. 6. User enter the increase in temperature at 50% CPU utilization 7. User enter the increase in temperature at 62% CPU utilization 8. User enter the increase in temperature at 75% CPU utilization 9. User enter the increase in temperature at 87% CPU utilization 10. User enter the increase in temperature at 100% CPU utilization	
Alternate Flows	
None	
Extends	
None	

UC13

Table 28: Simulation

Use Case ID	UC13
Use Case Name	Simulation
Actor (s)	User
Summary	A user Simulate.
Pre-conditions	
The machine usage file already exist and could contain the values of respective fields or not.	
Post Conditions	
Success	User successfully simulates the whole process.
Failure	Simulation doesn't start.
Basic Flow	
The use case starts when a user press the simulate button to start simulation. 1. User checked the option to start elastic search in background to fetch data. 2. User checked the option to start kibana in background to visualize data.	
Alternate Flows	
None	
Extends	
None	

2.6 System Development Life Cycle

2.6.1 Iterative system development life cycle

Iterative system development life cycle is a type of system development life cycle in which one module of a system is designed and tested after testing when flaws are removed for the next version and the next version is better than the previous one and this cycle continues until the error free final product is obtained. This is considered as one of the finest models for developments in which goals are not very clear. In the start of development as it goes on, the flaws are removed from the system and the actual product having real requirements is extracted out of it. The main feature of this model is that it provides flexibility to developer unlike in other models (where goals and tasks are initially defined) here the developer can make modifications in their work while designing the product. Iterative system development life cycle consists of 6 steps:

- Requirement Gathering and System Analysis
- System Design.
- Coding
- Integration and Testing
- Deployment of System
- Maintenance

2.6.1.1 Requirement Gathering and System Analysis

This is the first step in iterative system development life cycle. In this step all the requirements for a system are developed and general model of a product would be defined. The basic requirements for the system are also defined during this step.

2.6.1.2 System Design

In this phase the complete design of a system is made, all the requirements are described, user requirements are being done in this step and the most important task which is done in this step is the study of similar products. The team developed all the goals of the product based on the collected data and research and define tasks to achieve those goals. This is one of the most important steps in the iterative system development because, it is the actual base of the whole work. If this step would be done well then all it provides is the ease to all other steps and similarly problem in this step can cause problems in all other steps.

2.6.1.3 Coding

In this step the code of the product is written. This step is broken down into sub steps, the complete product is break down into modules. The separate code for each module is written in the end all the modules and are combined to get the final product. The well-defined code of each module is written, and all the well coding practices must be fulfilled in this phase to get the better and understandable product.

2.6.1.4 Integration and Testing

In this step the code of all modules is integrated into one product. After that basic testing of product is done and observed that if there is any serious issue arise during integration. If any problem is faced, then the above steps are done again in a better way to get rid of problem.

2.6.1.5 Deployment of a system

When the product achieves all its goals defined during requirement gathering phase and passed all its testing phases it is deployed. And if any issues arise during deployment then they are resolved during deployment.

2.6.1.6 Maintenance

After deployment the product is available for its target audience to be used for whatever purpose it is designed. If the user faces any problem in the product they must be listed down and must be addressed in the next version of the product, this will keep the product maintained and up to date.

3 System Design

3.1 Flow Chart

3.1.1 Flowchart-Diagram-1

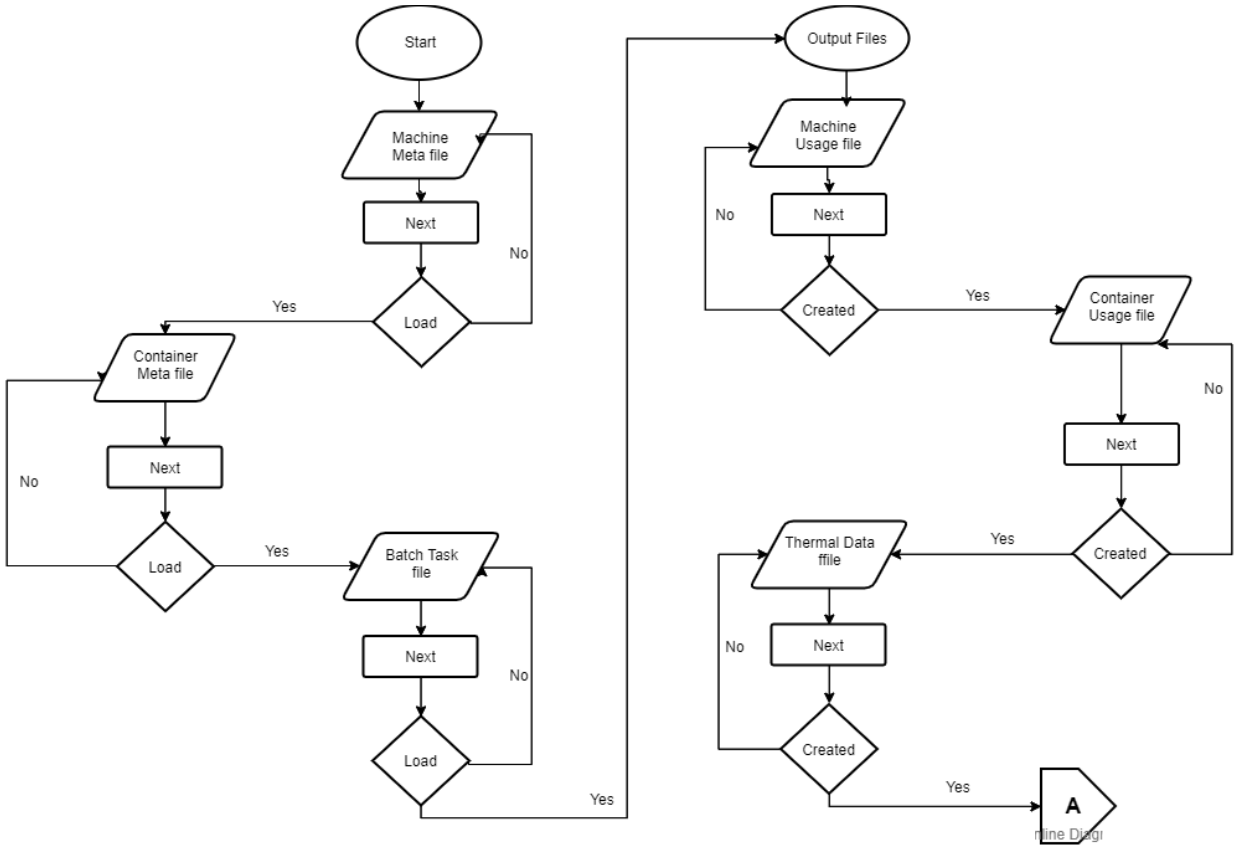


Figure 5: Flowchart-Diagram-1

3.1.2 Flowchart-Diagram-2

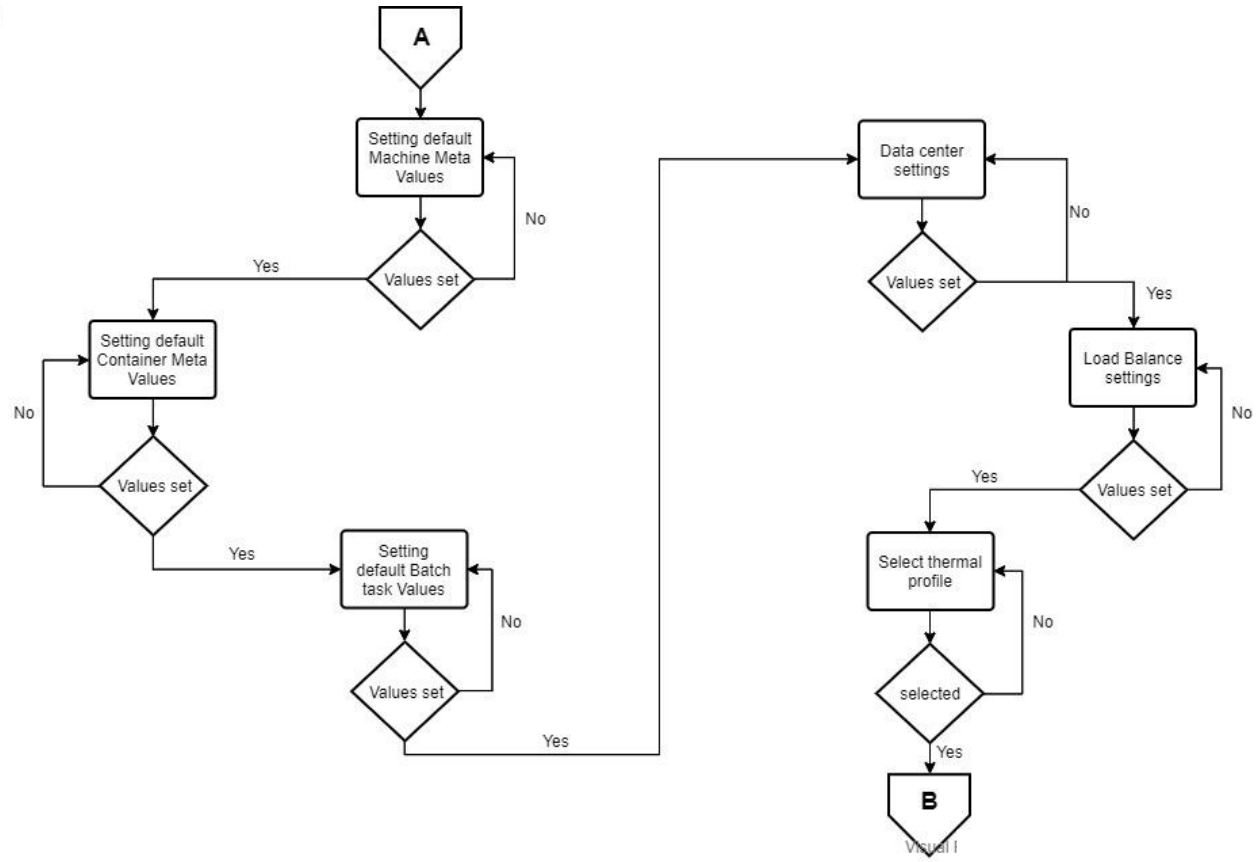


Figure 6: Flowchart-Diagram-2

3.1.3 Flowchart-Diagram-3

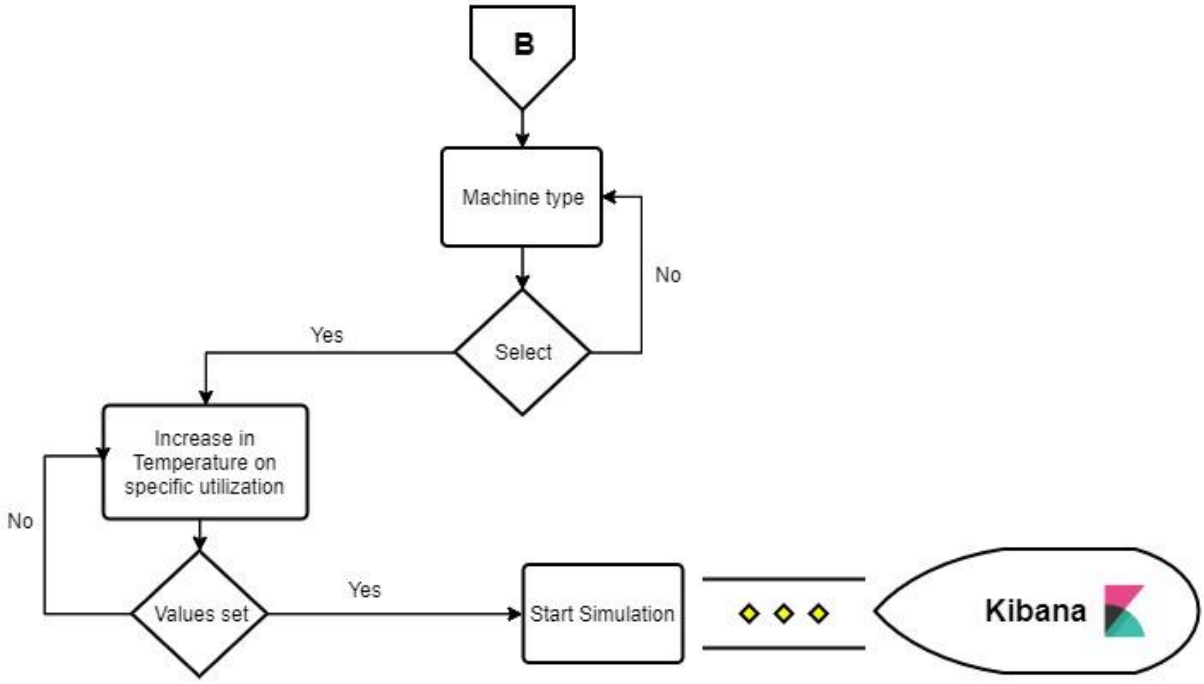


Figure 7: Flowchart-Diagram-3

3.2 System Architecture

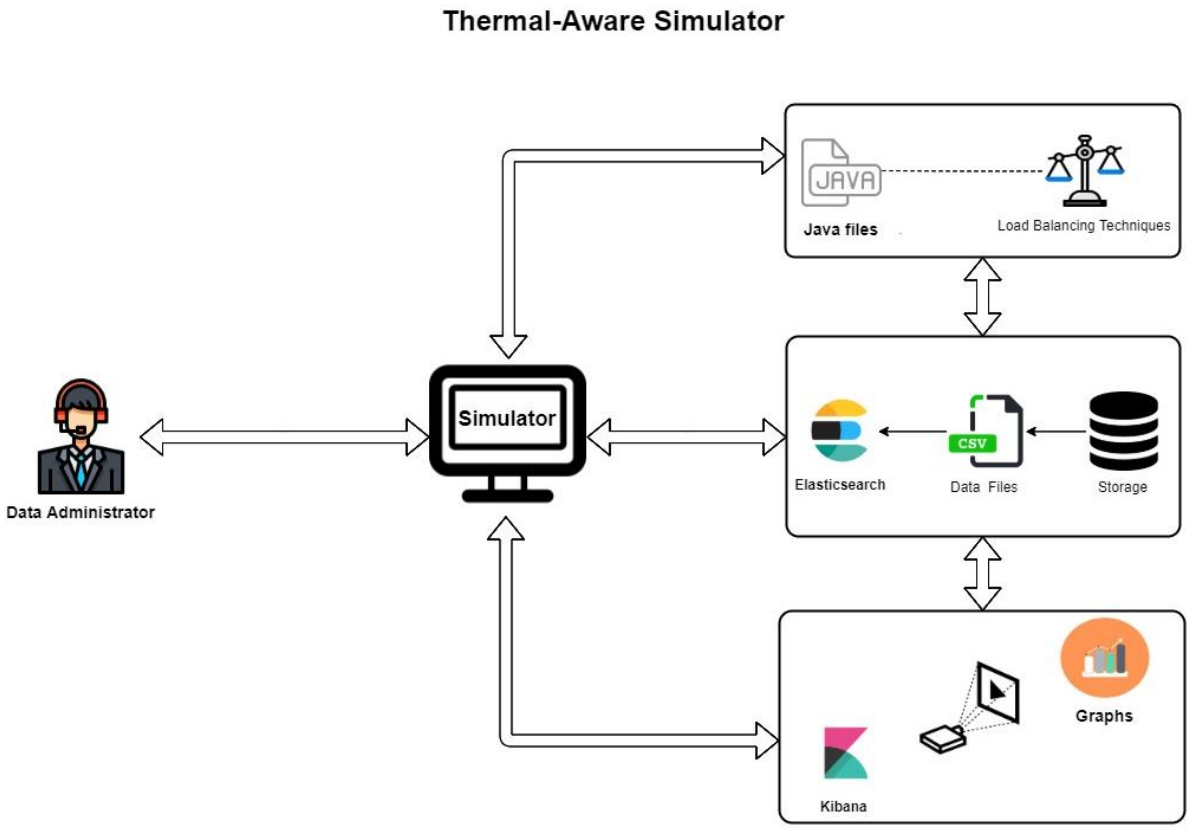


Figure 8: System Architecture

3.3 Sequence Diagram

3.3.1 Sequence Digram-1(Input Files)

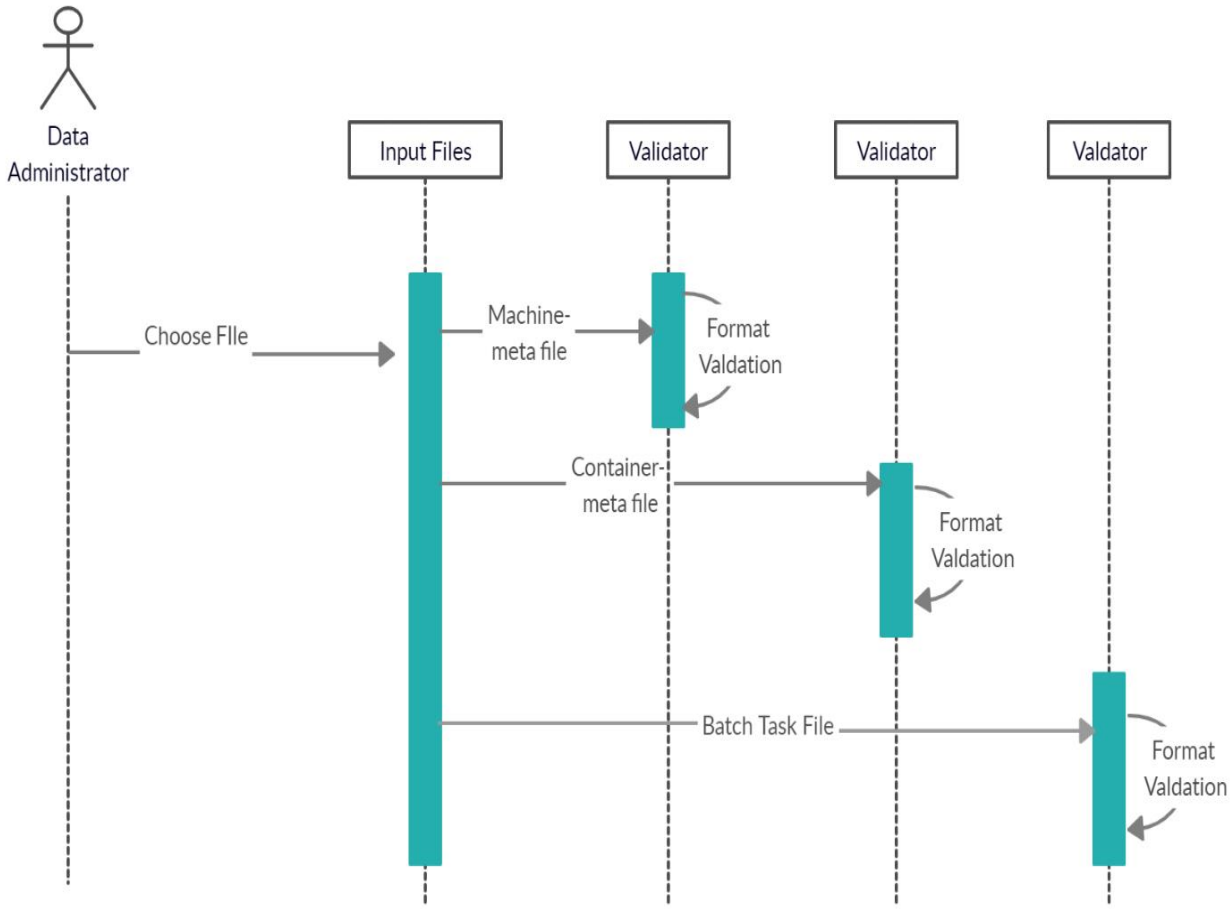


Figure 9: Sequence Digram-1(Input Files)

3.3.2 Sequence Digram-2(Output Files)

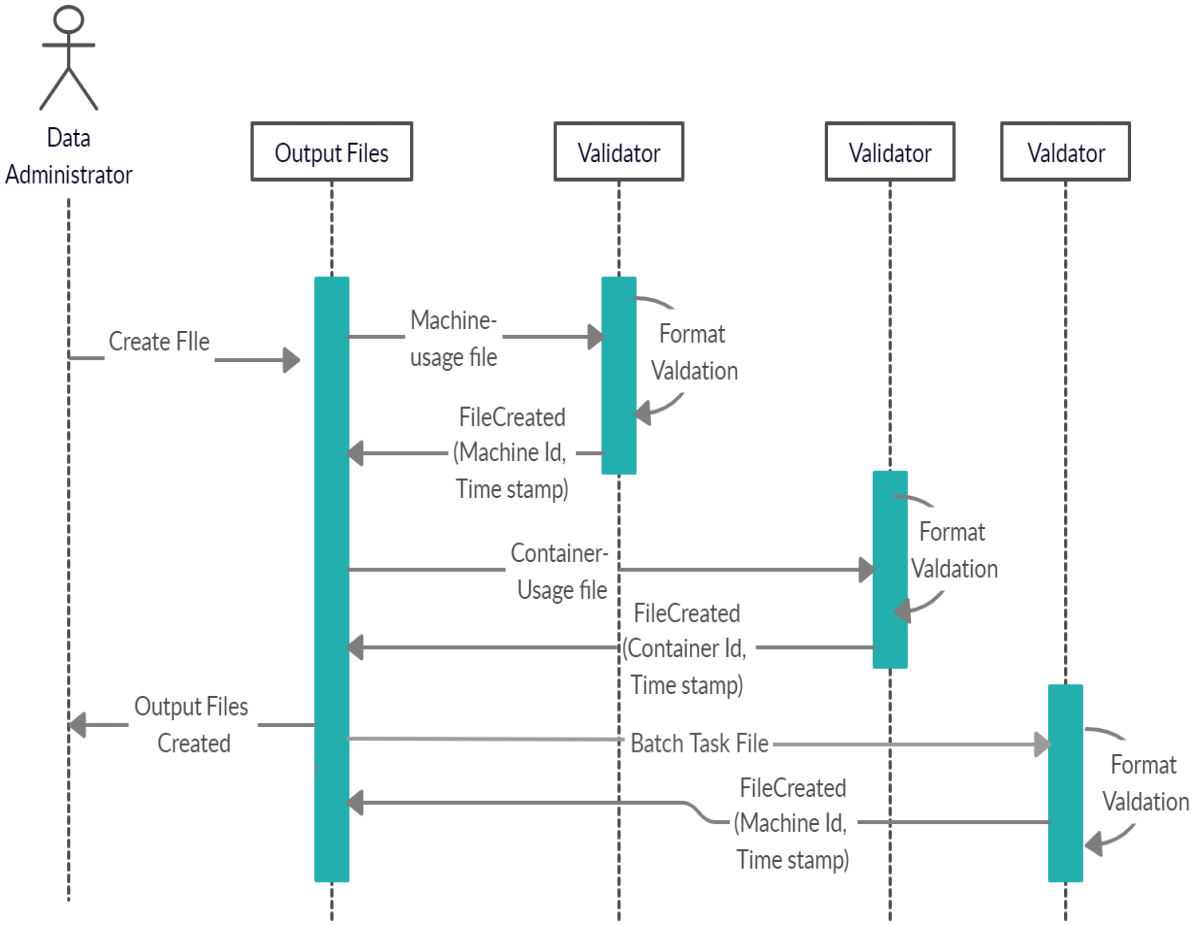


Figure 10: Sequence Digram-2(Output Files)

3.3.3 Sequence Digram-3(Default Values)

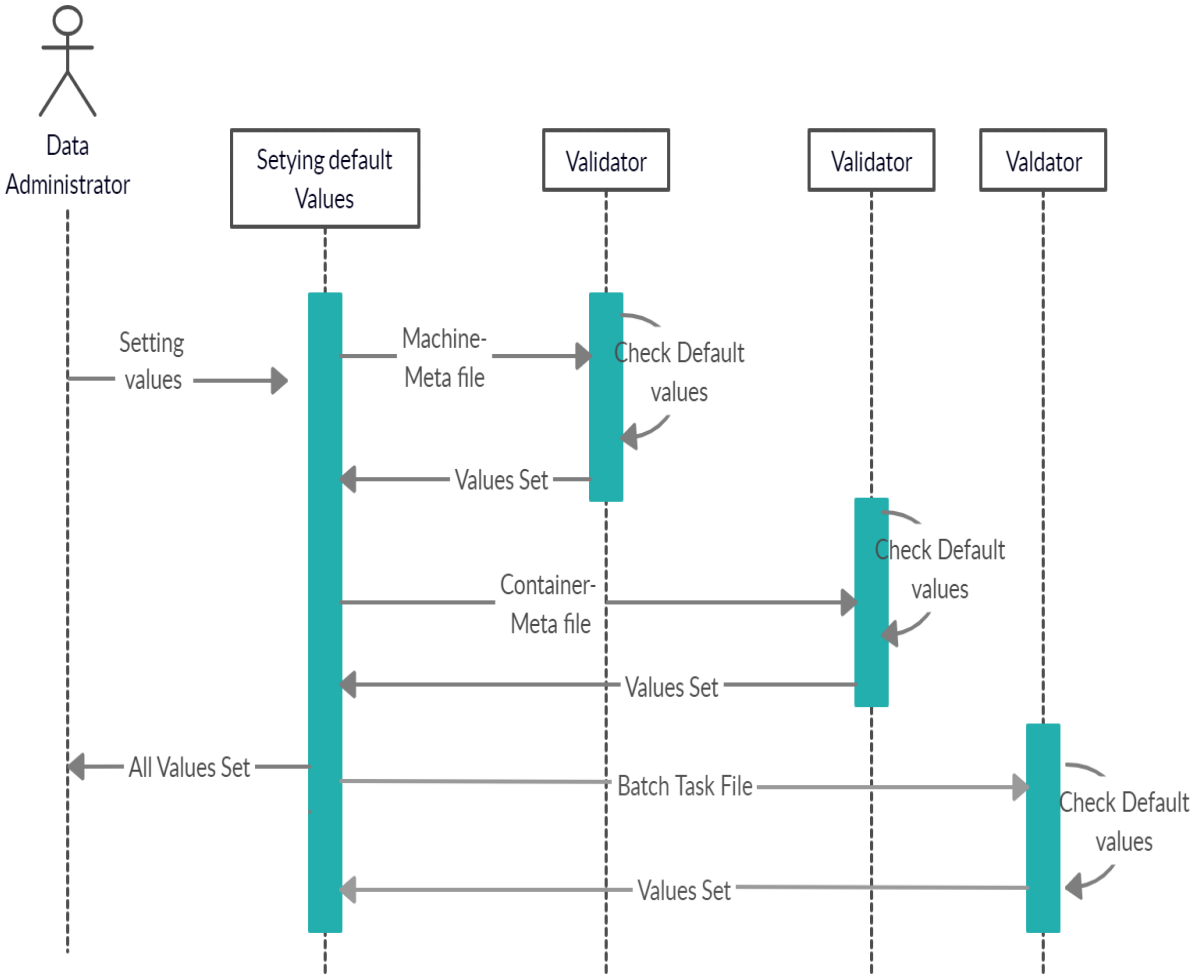


Figure 11: Sequence Digram-3(Default Values)

3.3.4 Sequence Digram-4(Data center Settings)

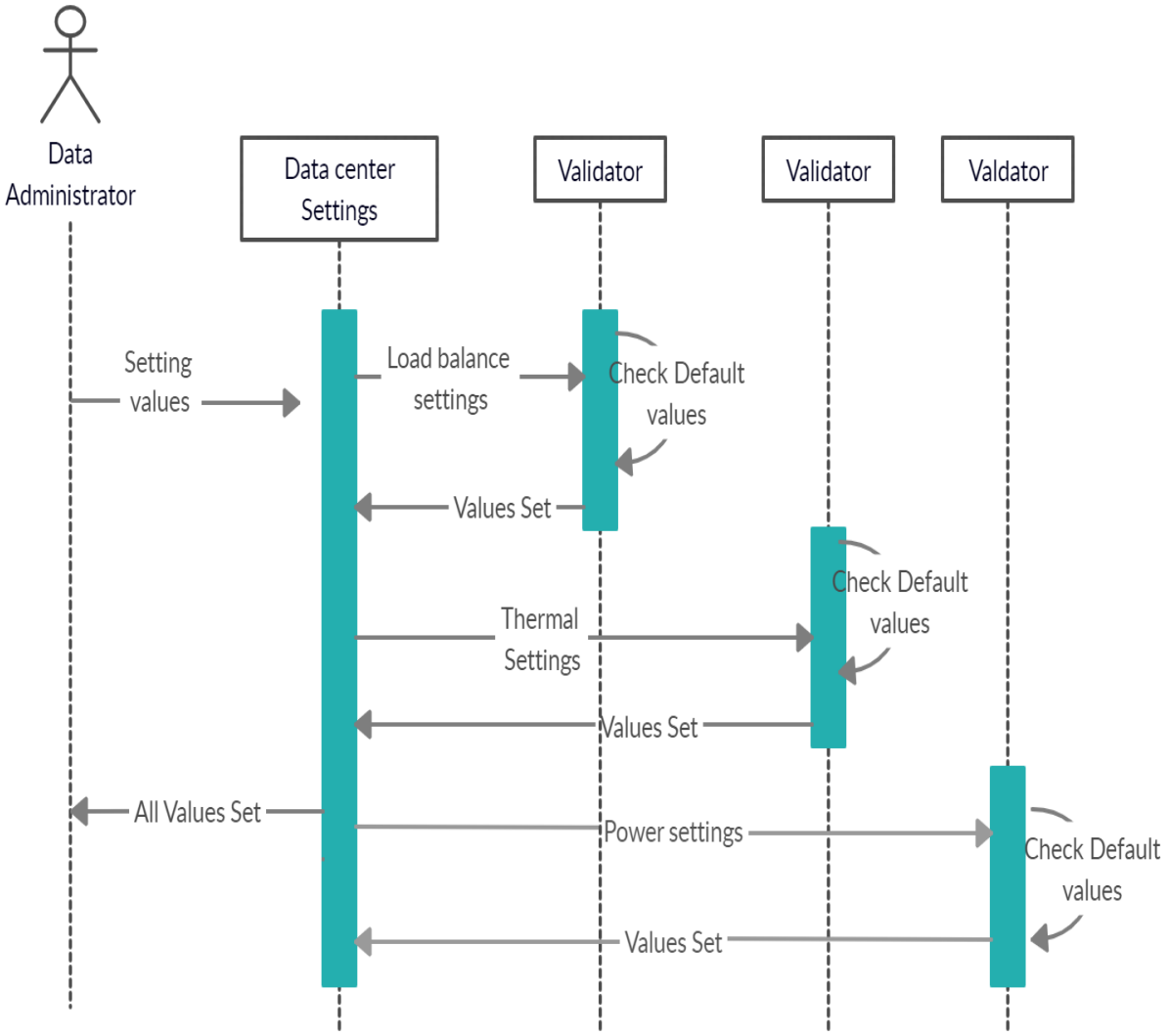


Figure 12: Sequence Digram-4(Data center Settings)

3.3.5 Sequence Digram-5(Simulation)

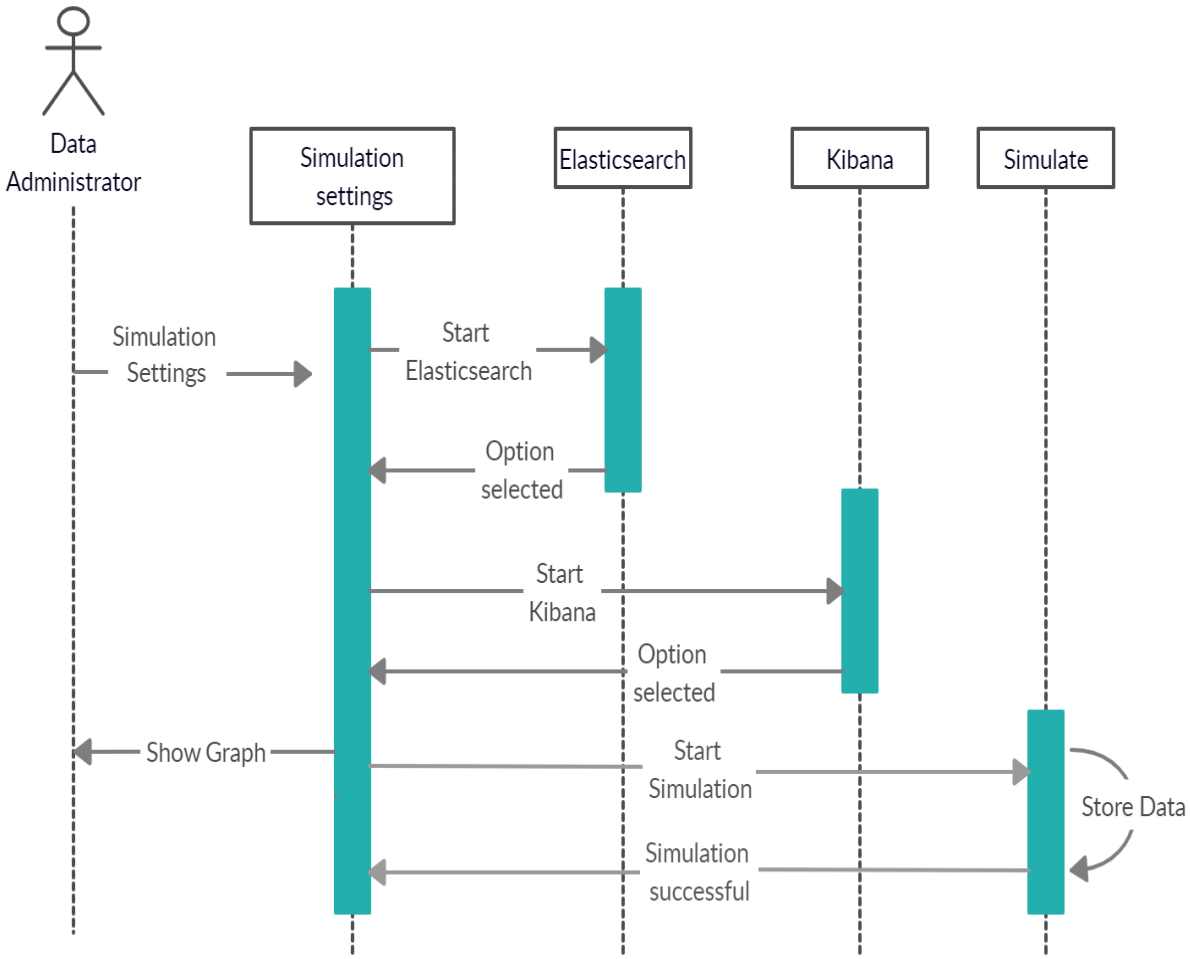


Figure 13: Sequence Digram-5(Simulation)

3.3.6 Sequence Digram-6(General)

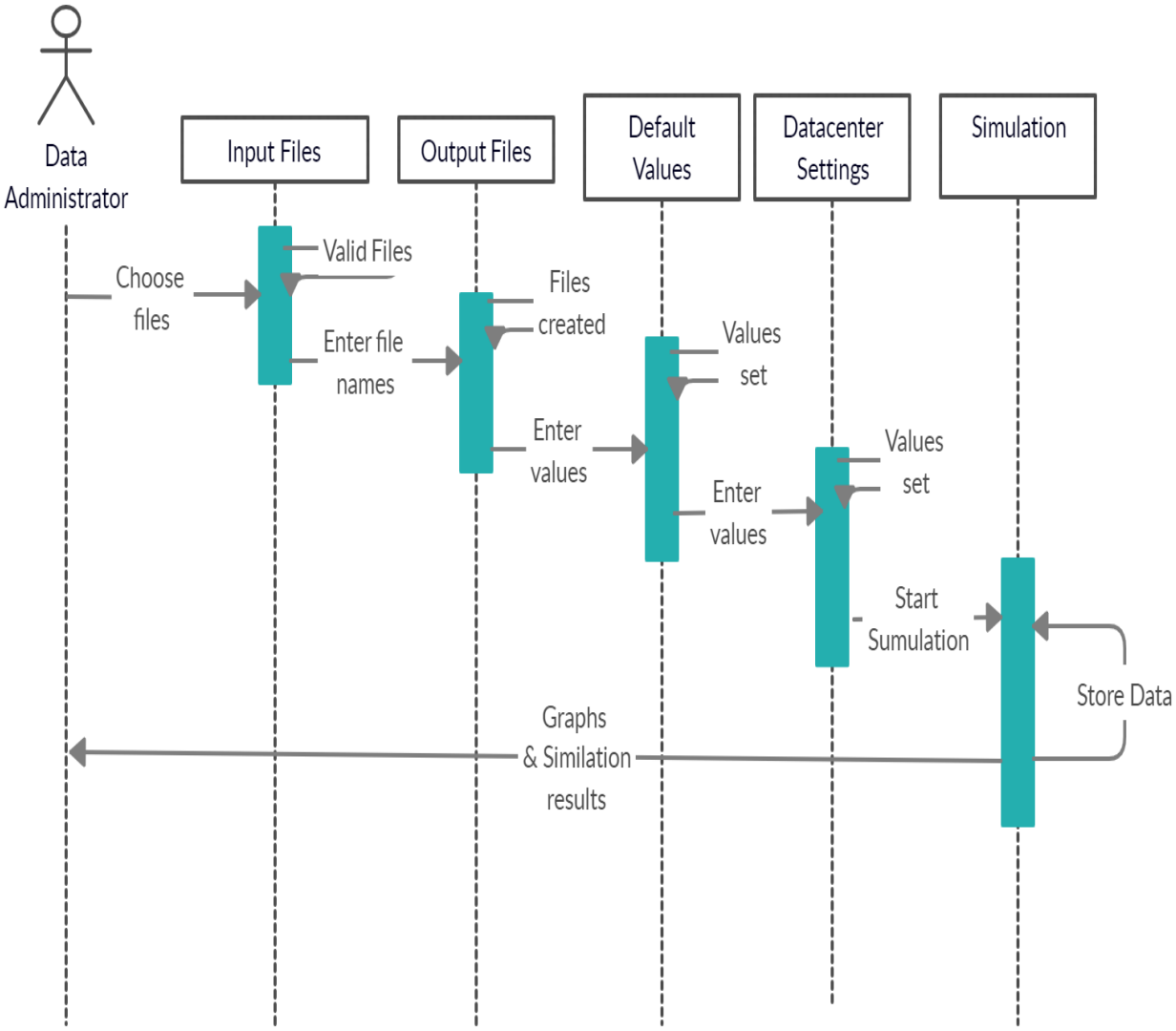


Figure 14: Sequence Digram-6(General)

3.4 Work Breakdown Structure

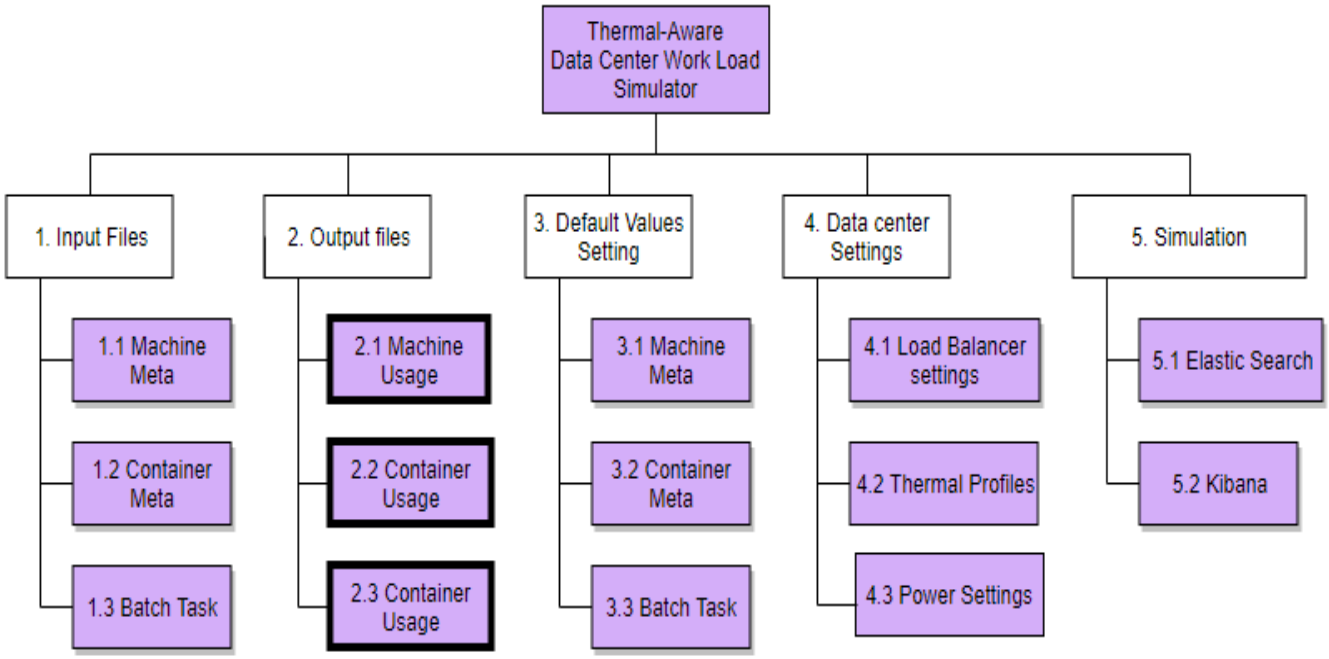


Figure 15: Work Breakdown Structure

3.5 Activity Diagram

3.5.1 Activity Diagram-1(Input Files)

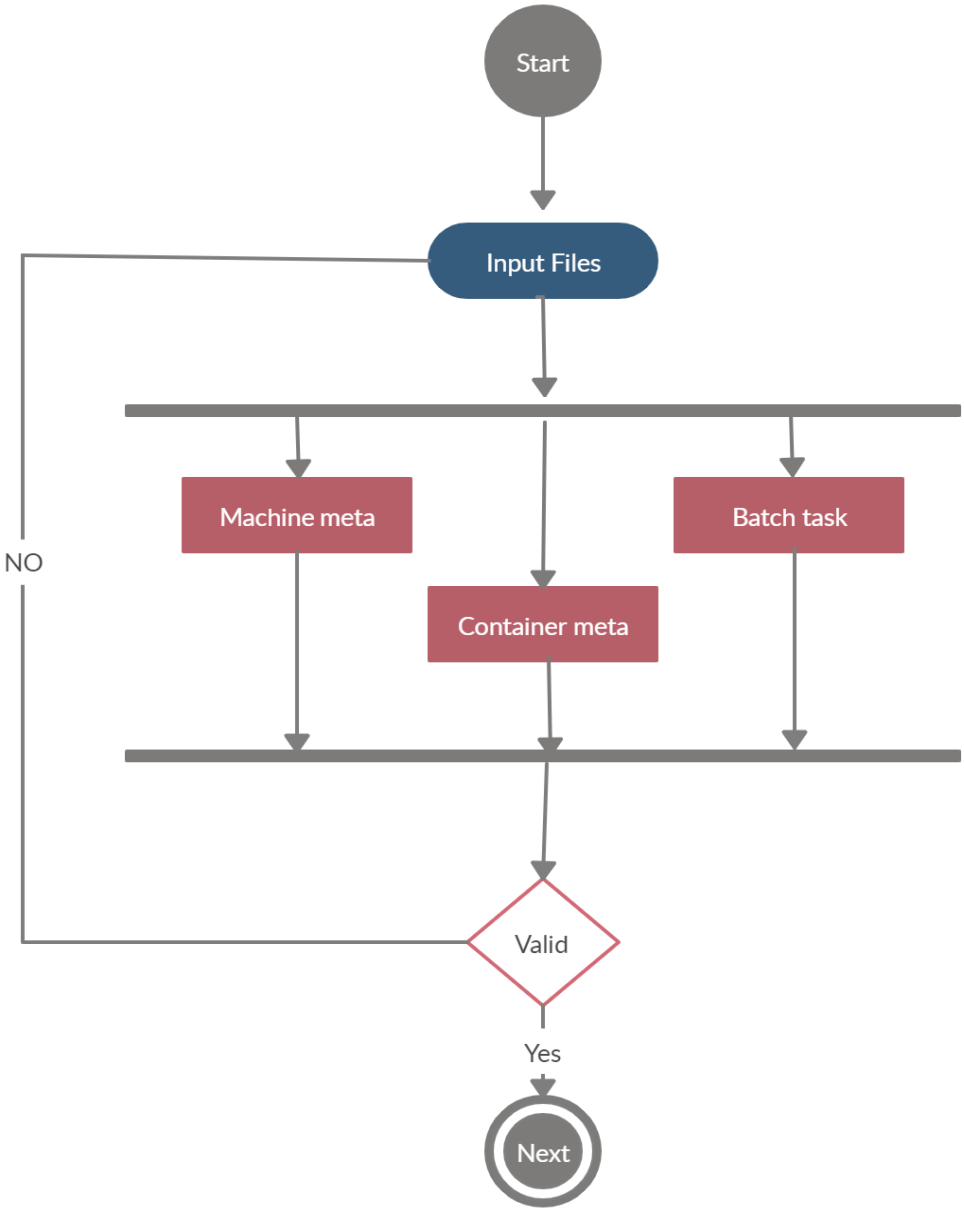


Figure 16: Activity Diagram-1(Input Files)

3.5.2 Activity Diagram-2(Output files)

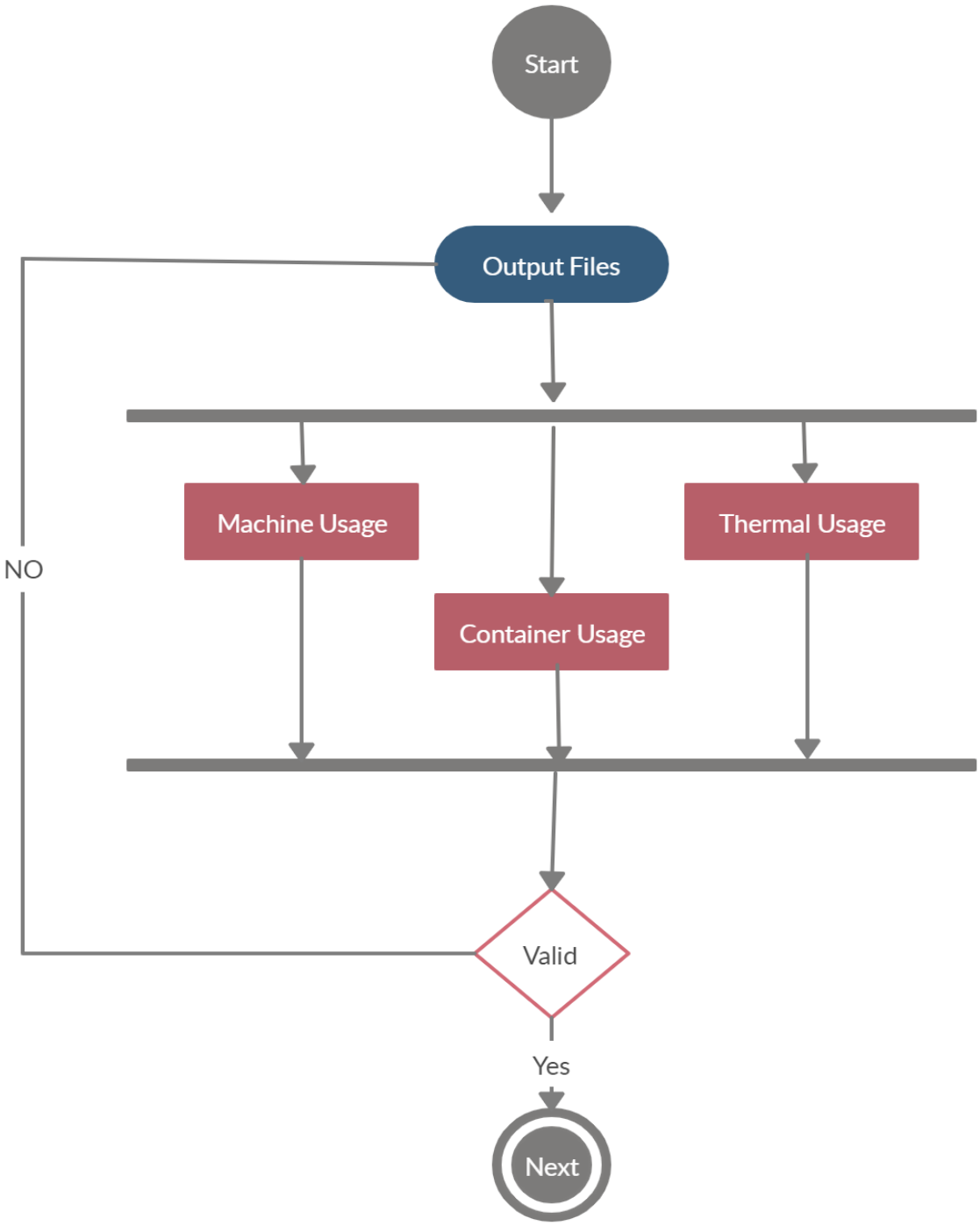


Figure 17: Activity Diagram-2(Output files)

3.5.3 Activity Diagram-3(Default settings)

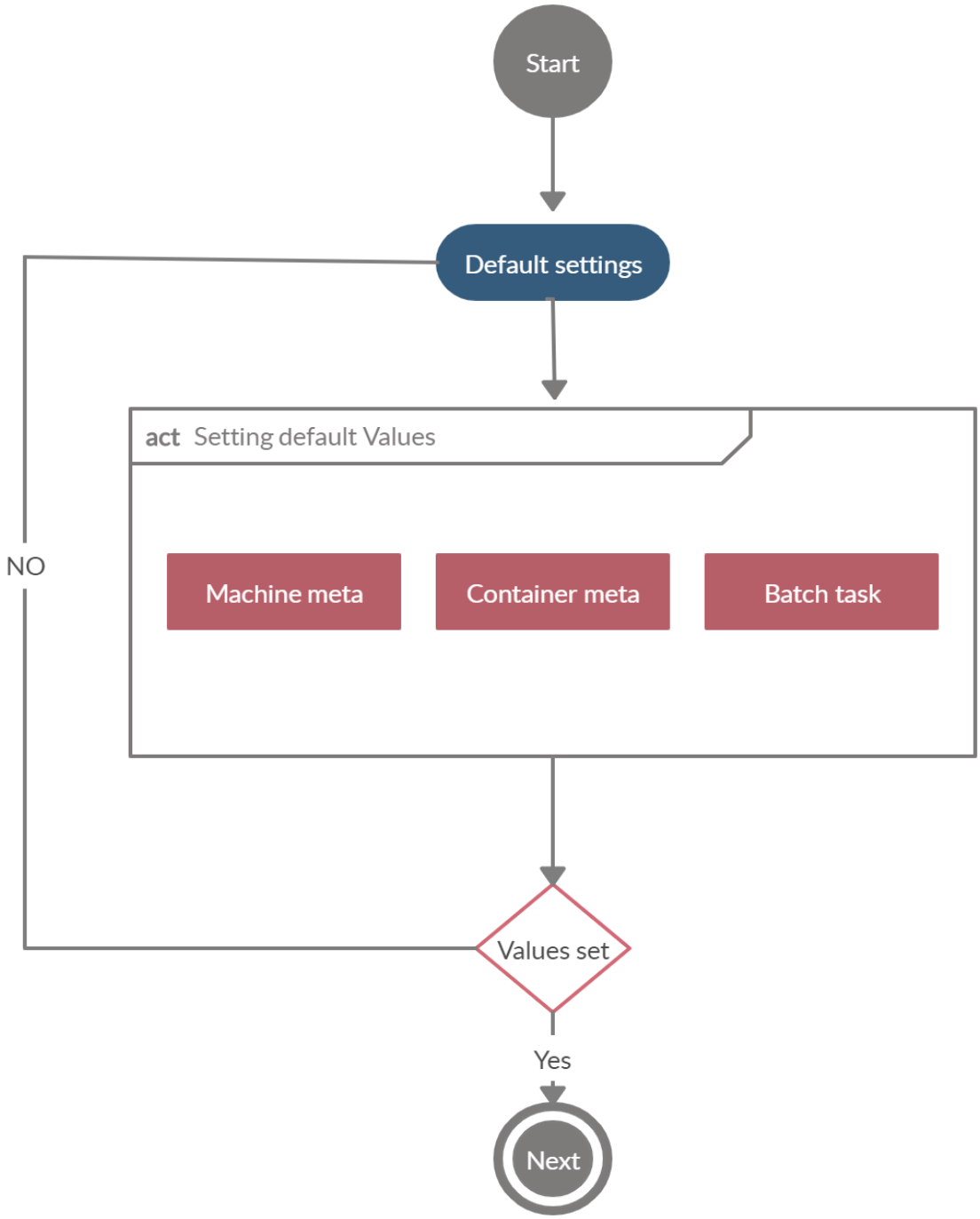


Figure 18: Activity Diagram-3(Default settings)

3.5.4 Activity Diagram-4(Datacenter Settings)

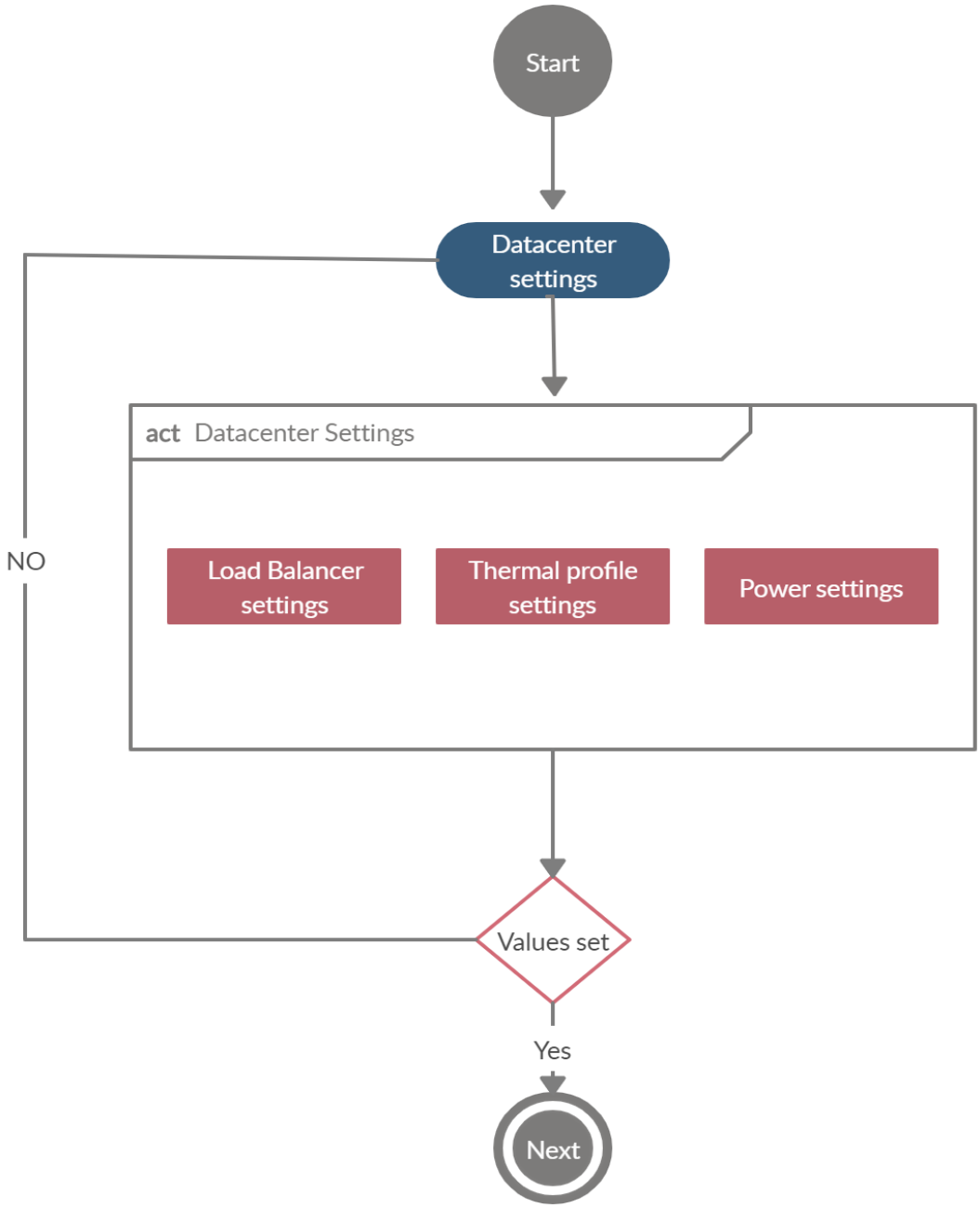


Figure 19: Activity Diagram-4(Datacenter Settings)

3.5.5 Activity Diagram-5(Simulation)

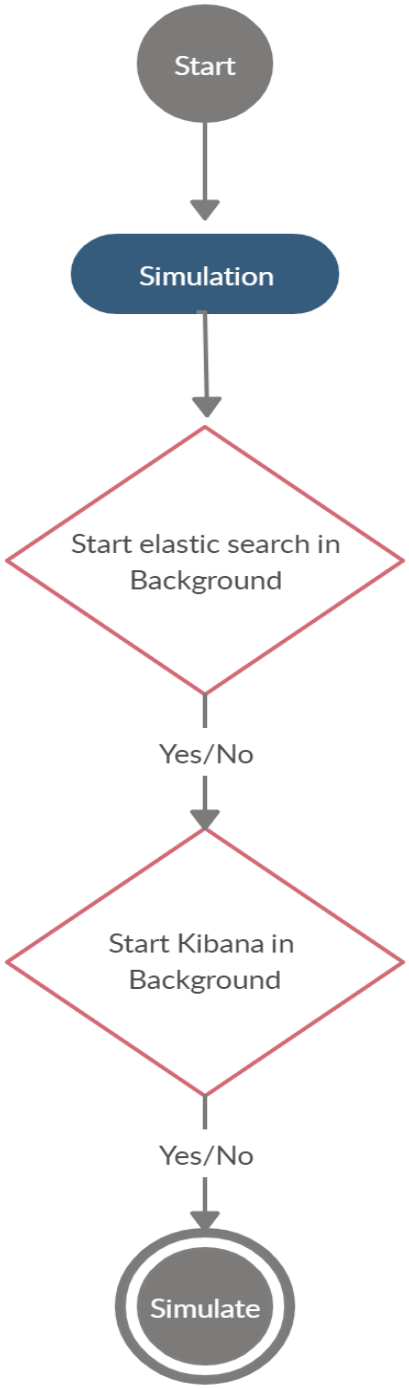


Figure 20: Activity Diagram-5(Simulation)

3.6 Network Diagram

Sr.	Task	Assigned to	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul
1	Planning & Designing	Usama & Bilal										
2	Research	Usama & Bilal										
3	Implementation Of Load Balancing Techniques	Usama										
4	Integration of ELK Stack	Usama										
5	Java GUI	Bilal										
6	Testing	Bilal										
7	Documentation	Usama & Bilal										

Figure 21: Network Diagram

3.7 Tools & technologies

Simulator: The core of the project is our simulator. Its logic is written in JAVA programming language using intelliJ IDEA.

Interface: The Interface of the simulator is made using "**java FX**".

Visualization: We are using "ELK Stack" for the visualization part of our project. It is comprised of three open-source projects stands for **Elasticsearch**, **Logstash and Kibana**.

4 Testing

4.1 Test Cases

Test Case ID	TC01		
Test Case Name	Input Files		
Designed Date	10/05/20		
Description	Testing the input file validation		
#	**Actions**	**Expected Result**	**Actual Result**
1	User selects the first input machine meta file	Files contain respective (Machine Id, time stamp) fields and have .csv type.	Files contain respective (Machine Id, time stamp) fields and have .csv type.
2	User selects the second input container meta file	Files contain respective (Container Id, time stamp) fields and have .csv type.	Files contain respective (Machine Id, time stamp) fields and have .csv type.
3	User selects the third input batch task file	Files contain respective (Machine Id, time stamp) fields and have .csv type.	Error: please select .csv file

Table 29: Test Case-01(Input Files)

Test Case ID	TC02		
Test Case Name	Output Files		
Designed Date	10/05/20		
Description	Testing the Output file creation and indexing		
#	**Actions**	**Expected Result**	**Actual Result**
1	User creates the first output machine usage file	File created with respective (Machine Id, time stamp) fields.	Files contain respective (Machine Id, time stamp) fields.
2	User creates the second output container usage file	File created with respective (Container Id, time stamp) fields.	Error: enter name of index is not entered
3	User creates the third output thermal usage file	Files created with respective (Machine Id, time stamp) fields.	Files contain respective (Machine Id, time stamp) fields.

Table 30: Test Case-02(Output Files)

Test Case ID	TC03		
Test Case Name	Default setting values		
Designed Date	10/05/20		
Description	Testing the default values of files		
#	**Actions**	**Expected Result**	**Actual Result**
1	User sets the machine meta default values.	Files contain default values of (Machine type, total CPU cores, total memory size, inlet temperature, machine status)	Files contain default values of (Machine type, total CPU cores, total memory size, inlet temperature, machine status)
2	User sets the container meta default values.	Files contain default values of (total CPU required, total memory required)	Files doesn't contain the values of (total CPU required, total memory required)
3	User sets the batch task default values.	Files contain default values of (total CPU required, total memory required, total instances)	Files doesn't contain the values of (total instances)

Table 31: Test Case-03(default settings)

Test Case ID	TC04		
Test Case Name	Data center Settings		
Designed Date	10/05/20		
Description	Testing the datacenter settings		
#	**Actions**	**Expected Result**	**Actual Result**
1	User sets the datacenter settings	Files contain respective (no. of racks in data center, machines per rack, total types of machine) fields.	User himself entered respective (no. of racks in data center, machines per rack, total types of machine) fields.
2	User sets the load balancer settings	Files contain respective (Balancing policy for containers, balancing policy for task) fields.	User himself entered respective (Balancing policy for containers, balancing policy for task) fields.
3	User sets the thermal profile settings	Files contain respective (increase in temperature at % CPU utilization) fields.	User himself entered respective (increase in temperature at % CPU utilization) fields.
4	User sets the power settings		

Table 32: Test Case-01(Datacenter settings)

Test Case ID	TC05		
Test Case Name	Simulation		
Designed Date	10/05/20		
Description	Testing the Simulation		
#	**Actions**	**Expected Result**	**Actual Result**
1	User checked the “start elastic search in background” option	Data loaded into elastic search through .csv files	Data loaded into elastic search through .csv files
2	User checked the “start kibana in background” option	Kibana runs on local host	Kibana runs on local host
3	User press Simulate button	Processed data stored in output files and result shows on kibana in the form of graph.	Processed data stored in output files and result shows on kibana in the form of graph.

Table 33: Test Case-05(Simulation)

5 Conclusion

5.1 Problems Faced and lessons learned

We were processing data in such a bulk that in excel sheet, it is not possible to apply power query on such a large amount of entries. To solve this problem of searching on data, we use elasticsearch which is an open-source tool for performing searching in data which also provides different aggregate functions that could be applied to fetch the desired data.

As the focus of our project was to made java based thermal aware simulator by implementing different load balancing policies, we need to visualize our results in the form of graph, pie charts etc. for the better understanding of results. For this problem of visualization of results, we use Kibana that is also an open-source tool in which there are several options of visualizing results in the form of graphs.

5.2 Project Summary

This report includes the working of simulator which would simulate the data center and would be efficient enough to process large amount of data. This would be capable of visualizing the results of data processing according to the user defined load balancing policy. And this simulator would be extendable whose features will be very beneficial for research community. Because, they can implement their own policies in this simulator and can see how the results varies from normal scenarios. We introduce thermal aware load balancing technique by doing research through which the heat generated by the servers get minimum and thus it will help to control the temperature of datacenters and will also be helpful in reducing the budget spent for cooling systems in datacenters that is solution to our problem statement.

6 References

[1] "Amazon," [Online]. Available: https://aws.amazon.com/elasticsearch-service/the-elk-stack/.

[2] D. A. Nayyar, "Best Open Source Cloud Computing Simulators," [Online]. Available: https://opensourceforu.com/2016/11/best-opensource-cloud-computing-simulators/.

Thermal Simulator

ORIGINALITY REPORT

18%	3%	1%	18%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Higher Education Commission Pakistan Student Paper	11%
2	Submitted to Leiden University Student Paper	3%
3	Submitted to University of Nicosia Student Paper	<1%
4	Submitted to Informatics Education Limited Student Paper	<1%
5	Submitted to University of Bradford Student Paper	<1%
6	Submitted to Staffordshire University Student Paper	<1%
7	Submitted to Middlesex University Student Paper	<1%
8	Submitted to CTI Education Group Student Paper	<1%
9	Submitted to Central Queensland University	

	Student Paper	<1%
10	Submitted to University of Greenwich Student Paper	<1%
11	cit3.ldl.swin.edu.au Internet Source	<1%
12	Submitted to London School of Science & Technology Student Paper	<1%
13	www.slideshare.net Internet Source	<1%
14	Submitted to Harrisburg University of Science and Technology Student Paper	<1%
15	hdl.handle.net Internet Source	<1%
16	Submitted to West University Of Timisoara Student Paper	<1%
17	Submitted to University of Bahrain Student Paper	<1%
18	Submitted to Hanoi National University Student Paper	<1%
19	Submitted to University of Technology, Sydney Student Paper	<1%

20	Submitted to University of Wales Institute, Cardiff Student Paper	<1%
21	Submitted to Macquarie University Student Paper	<1%
22	Derek K. Jones, Mara Cercignani. "Twenty-five pitfalls in the analysis of diffusion MRI data", NMR in Biomedicine, 2010 Publication	<1%
23	Submitted to The Hong Kong Polytechnic University Student Paper	<1%
24	Submitted to University of Mauritius Student Paper	<1%
25	www.coursehero.com Internet Source	<1%
26	othes.univie.ac.at Internet Source	<1%
27	Submitted to University of Oklahoma Student Paper	<1%
28	Submitted to University of Hertfordshire Student Paper	<1%