# Analyzing Game Statistics: a **way to** analyze **user's gaming behavior**

**Rian Dwi Putra**
***School of Computing***

***National College of Ireland***

***Dublin, Ireland***

**x22108637**@student.**ncirl.ie**

ABSTRACT

Big Data refers to a sizable volume of expanding data that is kept at numerous independent sources. It is a collection of both structured and unstructured data that cannot be managed by conventional database management tools or conventional data processing application models because it is too big, quick, and distinct.

The biggest difficulties facing big data applications are the storage of massive amounts of data from many sources and the processing of that data to derive knowledge or information for subsequent actions.

Using a programming approach, Apache Hadoop is a platform that offers dependable shared storage and distributed processing of massive data sets across clusters of commodity computers. Map Reduce handles data processing, while Hadoop Distributed File System (HDFS) handles data storage.

The main objective of the project is to implement the fundamental Hadoop components by constructing a multimode cluster, creating a shared HDFS base platform for the storage of enormous amounts of data from many sources, and performing Map Reduce processing on the data stored at these various nodes.

## INTRODUCTION

The eventual rise in popularity of personal computer (PC) games has captured the attention of a significant number of gamers around the world. The gaming industry is growing bigger than anyone can imagine.

One of the most well-known brands in the gaming industry is Steam, which **is a digital distribution platform for PC games**.[1]. Steam gives game developers and publishers more hope that their games will sell. Thanks to Steam, numerous video games, including fighting games, have been made available for PC.

Gamers can buy original games through the Steam Sale program without worrying about the price, but they may also be willing to spend more money on expensive new games. It appears that Steam has been beneficial, particularly for gamers who have grown accustomed to purchasing original products.

Everything begins at the absolute basics, such as playing on analog devices and a black-and-white game monitor, just like any other kind of media that existed before the video game business took off. So, as technology has advanced, gaming has become increasingly exciting. Without understanding the influence that gaming has on the players, it is very difficult to forecast the effect. These effects will depend on a few factors from the perspective of gamers. This viewpoint will be later compiled as player information or data.

There are several connections between the gaming industry and data analysis. It would be quite interesting to talk about the enormous amount of data that is used in this sector. The usage of achievement statistics—data that in this study was referred to as data—in digital games has become a predictor of how this trend will develop.

*The findings of this study will be interpreted as follows:*
*1. What is the top 10 platform used in all games?*
*2. What is top 5 genres by the games played all over the time?*
*3. How many games have been released during past 15 years?*

## DATA

The experiments presented in this study are based on every game, over 16,000 games distributed by Stream, and is focused on the dataset used in this study was:

1. https://www.kaggle.com/datasets/gregorut/videogamesales

Table 1. Games

| Column Name | Description |
|---|---|
| No | Number |
| Name | Game's Name |
| Platform | Platform of the Published Games |
| Year | Year of the Released Games |
| Genre | Game's Genre |
| Publisher | Game's Publisher |
| | |

Table 2. Sales of Games

| Column Name | Description |
|---|---|
| No | Number |
| Name | Game's Name |

| | |
|---|---|
| NA_Sales | Total Sales of The Games in North America (in Million) |
| EU_Sales | Total Sales of The Games in Europe (in Million) |
| JP_Sales | Total Sales of The Games in Japan (in Million) |
| Other_Sales | Total Sales of The Games in Rest of The World (in Million) |
| Global_Sales | Total Sales of The Games in Worldwide(in Million) |

## METHODOLOGY

This study's distribution strategy will be centered on Hadoop. Hadoop is an open-source database management system for storing massive datasets. This system is effective at both storing and processing data, from gigabytes to petabytes. Despite processing a lot of data, the process moves quickly since more computers are involved. The processing time can be sped up since data is distributed and processed simultaneously. [4]

The following modules make up the Hadoop framework:

1. Among the libraries that support Hadoop Modules is Hadoop Common.
2. HDFS, a distributed data storage platform that enables the storing of very massive data
3. The Hadoop cluster's resource management is aided by the Hadoop YARN architecture.
4. MapReduce in Hadoop

Before moving to Implementation and Architecture Phase, this study will not remove any (blank) variable from Table Sales of Games. Because (blank) variable means that games have no Sales in specified region at all.

Meanwhile this study will replace 'N/A' (Not Available) from Year because it has no value and can be replace with the Mode of dataset population. Mode is the most frequently occurring value in the data set. Mode is also the majority or most frequent value. This mode can be used to determine a sample from a population.

```
public static int mode(int a[]) {
    int maxValue, maxCount;

    for (int i = 0; i < a.length; ++i) {
        int count = 0;
        for (int j = 0; j < a.length; ++j) {
            if (a[j] == a[i]) ++count;
        }
        if (count > maxCount) {
            maxCount = count;
            maxValue = a[i];
        }
    }

    return maxValue;
}
```

Figure 1. Replacing N/A value with Mode

## IMPLEMENTATION AND ARCHITECTURE

This study's performance evaluation will pay particular attention to MapReducer's output. "MapReduce is a programming paradigm designed to process enormous amounts of data in distributed and parallel fashion across large computer clusters. MapReduce can be roughly separated into two processes, namely the Map process and the Reduce process, while processing data. Each computer in a cluster—a collection of interconnected computers—receives both sorts of activities, which run concurrently and independently of one another.

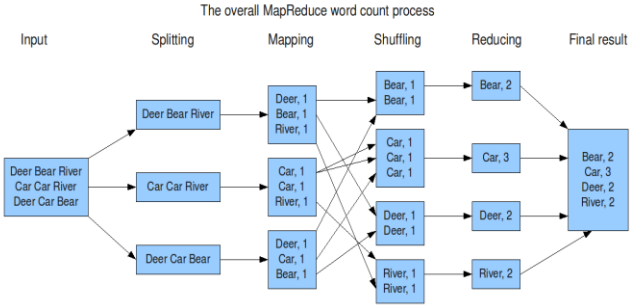



Figure 2. Mapreduce Process

By breaking up the processing into multiple independent tasks, MapReduce is supposed to be able to handle very huge amounts of data. [2] [3] There are several stages to MapReduce:

1. Dividing the data in the input. This is accomplished by slicing up the input data supplied by MapReduce customers into smaller chunks.
2. Mapping. One of MapReduce's most crucial phases is mapping. The partitioned data blocks are processed during the mapping phase to produce intermediate key-value pairs.
3. Shuffling or randomization. One or more computers can be used for the mapping phase. The shuffle phase, prior to the shrink phase, collects one or more distinct keys on a specific machine to facilitate aggregation. This allows the key-value pairs **generated by** the mapper to be distributed **across multiple machines**.
4. Reducing. The task of the reduction **phase is** to aggregate all intermediate key-value pairs under **the same key.**

In simple terms, a mapper is meant to filter and convert input into something that the reducer can collect.

Installation of Hadoop

Before executing file in Hadoop, this study presume that we have already installed Hadoop in our environment

Formatting namenode


Figure 3. Formatting Namenode

Go to C:\Hadoop3.3.0\sbin


Figure 4. Move to sbin directory

By using this command, we start all the services ;

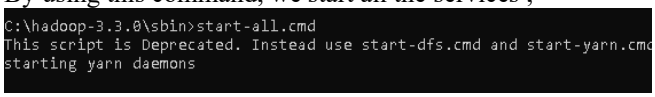



Figure 5. Running all the services

Make sure these apps are running

– Hadoop Namenode

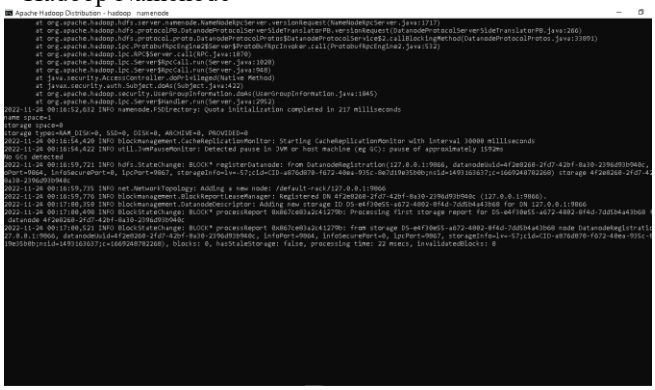


Figure 6. Running Hadoop Namenode

– Hadoop datanode

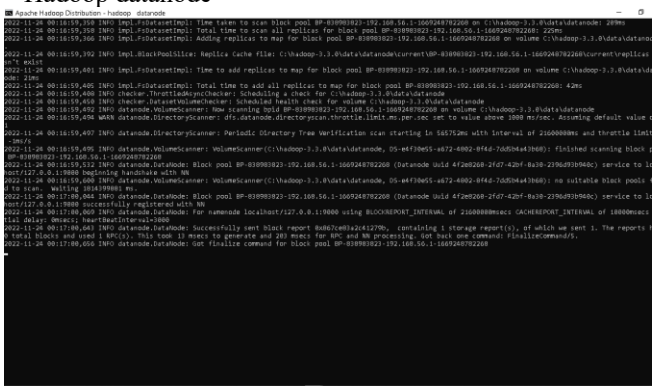


Figure 7. Running Hadoop Datanode

– YARN Resource Manager

Figure 8. Running YARN

– YARN Node Manager

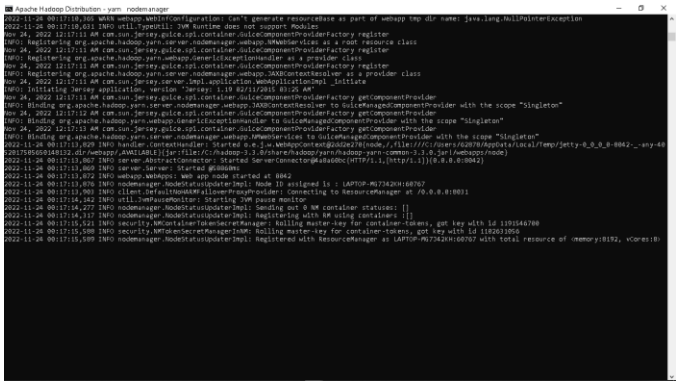


Figure 9. Running Node Manager

Confirmation that all services running smoothly;

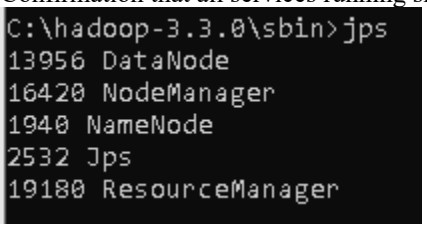
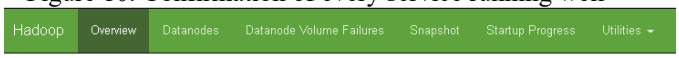



Figure 10. Confirmation of every service running well




Overview 'localhost:9000' (active)

| | |
|---|---|
| **Started:** | Thu Nov 24 00:16:42 +0000 2022 |
| **Version:** | 3.3.0, raa96f1871bfd858f9bac59cf2a81ec470da649af |
| **Compiled:** | Mon Jul 06 19:44:00 +0100 2020 by brahma from branch-3.3.0 |
| **Cluster ID:** | CID-a876d870-f672-40ea-935c-8e7d19e35b0b |
| **Block Pool ID:** | BP-838983823-192.168.56.1-1669248782268 |

Figure 11. overview of hadoop running in the system

Mapper Class for Game by Platform

```
package GameApp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GamePlatformMapper extends MapReduceBase implements Mapper <LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SinglePlatformData = valueString.split(",");
        output.collect(new Text(SinglePlatformData[2]), one);
    }
}
```

Figure 12. Mapper Class for Game by Platform

Reducer Class for Game by Platform

```
package GameApp;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GamePlatformReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        Text key = t_key;
        int frequencyForPlatform = 0;
        while (values.hasNext()) {
            IntWritable value = (IntWritable) values.next();
            frequencyForPlatform += value.get();

        }
        output.collect(key, new IntWritable(frequencyForPlatform));
    }
```

Figure 13. Reducer Class for Game by Platform

Driver Class for Game byPlatform

```
package GameApp;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class GamePlatformDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        JobConf job_conf = new JobConf(GamePlatformDriver.class);
        job_conf.setJobName("Game per Platform");
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);
        job_conf.setMapperClass(GameApp.GameMapper.class);
        job_conf.setReducerClass(GameApp.GamePlatformReducer.class);
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figure 14. Driver Class for Game byPlatform

Mapper Class for Games by Genre

```
package GameApp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GameGenreMapper extends MapReduceBase implements Mapper <LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleGenreData = valueString.split(",");
        output.collect(new Text(SingleGenreData[4]), one);
    }
}
```

Figure 15. Mapper Class for Games by Genre

Reducer Class for Games by Genre

```
package GameApp;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GameGenreReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        Text key = t_key;
        int frequencyForGenre = 0;
        while (values.hasNext()) {
            IntWritable value = (IntWritable) values.next();
            frequencyForGenre += value.get();

        }
        output.collect(key, new IntWritable(frequencyForGenre));
    }
```

Figure 16. Reducer Class for Games by Genre

Driver Class for Games by Genre

```
package GameApp;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class GameGenreDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        JobConf job_conf = new JobConf(GameGenreDriver.class);
        job_conf.setJobName("Game per Genre");
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);
        job_conf.setMapperClass(GameApp.GameMapper.class);
        job_conf.setReducerClass(GameApp.GameGenreReducer.class);
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figure 17. Driver Class for Games by Genre

Mapper Class for Game by Year

```
package GameApp;

import java.io.IOException;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GameYearMapper extends MapReduceBase implements Mapper <LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);

    public void map(LongWritable key, Text value, OutputCollector <Text, IntWritable> output, Reporter reporter) throws IOException {

        String valueString = value.toString();
        String[] SingleYearData = valueString.split(",");
        output.collect(new Text(SingleYearData[3]), one);
    }
}
```

Figure 18. Mapper Class for Game by Year

Reducer Class for Game by Year

```
package GameApp;

import java.io.IOException;
import java.util.*;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.*;

public class GameYearReducer extends MapReduceBase implements Reducer<Text, IntWritable, Text, IntWritable> {

    public void reduce(Text t_key, Iterator<IntWritable> values, OutputCollector<Text,IntWritable> output, Reporter reporter) throws IOException {
        Text key = t_key;
        int frequencyForYear = 0;
        while (values.hasNext()) {
            IntWritable value = (IntWritable) values.next();
            frequencyForYear += value.get();

        }
        output.collect(key, new IntWritable(frequencyForYear));
    }
```

Figure 19. Reducer Class for Game by Year

Driver Class for Game by Year

```
package GameApp;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.*;
import org.apache.hadoop.mapred.*;

public class GameYearDriver {
    public static void main(String[] args) {
        JobClient my_client = new JobClient();
        JobConf job_conf = new JobConf(GameYearDriver.class);
        job_conf.setJobName("Game per Year");
        job_conf.setOutputKeyClass(Text.class);
        job_conf.setOutputValueClass(IntWritable.class);
        job_conf.setMapperClass(GameApp.GameMapper.class);
        job_conf.setReducerClass(GameApp.GameYearReducer.class);
        job_conf.setInputFormat(TextInputFormat.class);
        job_conf.setOutputFormat(TextOutputFormat.class);
        FileInputFormat.setInputPaths(job_conf, new Path(args[0]));
        FileOutputFormat.setOutputPath(job_conf, new Path(args[1]));

        my_client.setConf(job_conf);
        try {
            JobClient.runJob(job_conf);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Figure 20. Driver Class for Game by Year

RESULTS

Based on generated data, this study will analyze and visualizing data using Microsoft Power BI – another business intelligence tools.

- Top 10 Platform used in All Games

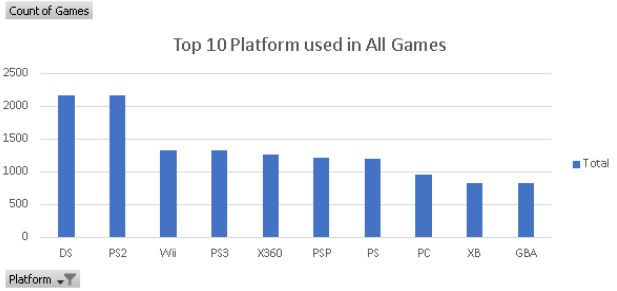



Figure 21. Top 10 Platform used in All Games

This figure above tells that most of the games are generally played in DS and PS2 Platform, as shown in table below :

Table 3. Top 10 Platform used in All Games

| | |
|---|---|
| DS | 2163 |
| PS2 | 2160 |
| Wii | 1325 |
| PS3 | 1323 |
| X360 | 1259 |
| PSP | 1209 |
| PS | 1196 |
| PC | 960 |
| XB | 824 |
| GBA | 822 |

- Top 5 Genre based on Games Played

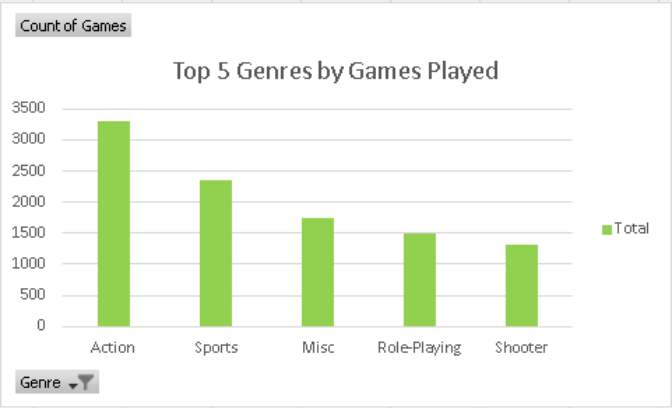



Figure 22. Top 5 Genres by Games Played

The previous figure tells that the most played games by genre are Action and Sport, as shown in table below

Table 4. Top 5 Genre by Games Played

| | |
|---|---|
| Action | 3313 |
| Sports | 2346 |
| Misc | 1739 |
| Role-Playing | 1488 |
| Shooter | 1310 |

- Total number of Games Published for the Last 15 Years

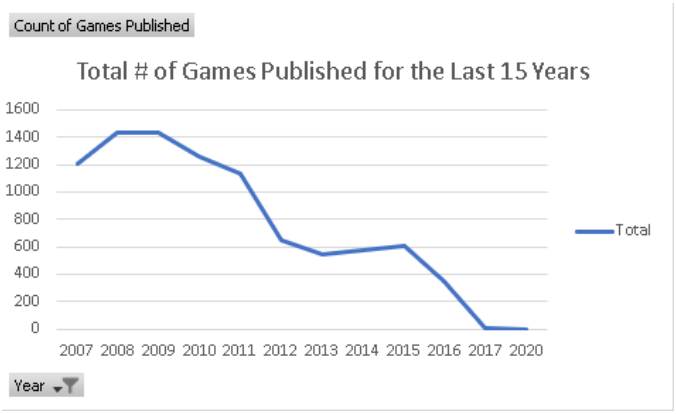



Figure 23. Total # of Games Published for the Last 15 Years

From the picture above, we can see the trendline for published are decreasing all over the period. This does not include year 2018-2019 and 2021-2022 as there are no published games during that time. As shown in table below;

Table 5. Total # of Games Published over the Last 15 Years

| | |
|---|---|
| 2007 | 1201 |
| 2008 | 1428 |
| 2009 | 1429 |
| 2010 | 1257 |
| 2011 | 1136 |
| 2012 | 652 |
| 2013 | 543 |
| 2014 | 578 |
| 2015 | 608 |
| 2016 | 344 |
| 2017 | 3 |
| 2020 | 1 |

CONCLUSION AND FUTURE WORKS

Based on the result generated, this study concludes and interprets every drawn diagram in the Result section. In Future works, this study is supposed to do analysis in Table Sales of Games. Some conclusions can be drawn from this table are such as;

- What games generate the most sales from every region?
- How many percentages of Sales from every region which contribute to the Global Sales?
- Which region has the most impact to the Global Sales?
- Which region has the least impact to the Global Sales?

## REFERENCES


[1] D. Lin, C. P. Bezemer, Y. Zou, and A. E. Hassan, “An empirical study of game reviews on the Steam platform,” Empirical Software Engineering, vol. 24, no. 1, pp. 170–207, 2019.

[2] C. Ji et al., “Big data processing: Big challenges and opportunities,” Journal of Interconnection Networks, vol. 13, no. 03n04, 2012.

[3] J. P. Verma, B. Patel, and A. Patel, “Big data analysis: recommendation system with Hadoop framework,” in 2015 IEEE International Conference on Computational Intelligence & Communication Technology, IEEE, 2015, pp. 92–97.

[4] J. Nandimath, E. Banerjee, A. Patil, P. Kakade, S. Vaidya, and D. Chaturvedi, “Big data analysis using Apache Hadoop,” in 2013 IEEE 14th International Conference on Information Reuse & Integration (IRI), IEEE, 2013, pp. 700–703.