National Institute of Technology,Raipur

Department of Computer Science & Engineering

Applications of Hadoop in Distributed Computing

A Term Project on Network Programming

GitHub Project Link:
https://github.com/Angelion1/Hadoop_Application_Websites_Interconnections

Submitted By:

Roll no: 14115011	Name: Anshul Verma
Roll no: 14115035	Name: Hari Shrawgi
Roll no: 14115037	Name: Harshdeep Kaur
Roll no: 14115071	Name: Prerana Agrawal
Roll no: 14115083	Name: Shriya Thawait

Abstract

There has been a constant trend of our technology moving towards distributed computing. With the advent of Big-Data and Cloud, it seems inevitable that distributed computing would soon take the centre-stage.

In this era of data proliferation there is a need for intelligent use of our limited resources. We as computer scientists of this new generation must accustom ourselves with the new paradigms of computer science. In an attempt to do the same, in this project we have aimed to learn and harness the power afforded by the Apache Hadoop framework. Being familiar with JAVA through our Network Programming class has been a boon for us as Hadoop works brilliantly in tandem with JAVA.

In this project we have simple demonstrated the use of Hadoop to implement a task which is very time consuming when implemented sequentially. The application developed in this project deals with the domain of web-crawling and website indexing. Finding the various interconnections among websites is often the first step in many web applications like spidering. Thus, it is of prime importance that this step be time efficient.

Thus, we chose and implemented a Hadoop implementation of the task. MapReduce framework of the Apache Hadoop library has been used by us to divide and parallelize the task.

We were able to setup a single node cluster of Hadoop and successfully run our application. As a result of the application we found out the various links from our college website.

Github Repository of the project

https://github.com/Angelion1/Hadoop_Application_Websites_Interconnections

List of Figures

Table of Contents

Introduction

1. Motivation

The main motivation for taking up this project has been to learn a technology whose relevance has significantly increased in the past and is pitted to be the future [1].

We believe that Distributed computing is a very important sub-domain of computer science which will become more and more important as the era Artificial Intelligence ushers in. With an inundation of data, advancements in silicon are failing to keep up [2]. Thus, it is logical that leveraging computing resources all around the world will soon be as common as sharing pen-drives.

The following points outline the need for distributed computing:

- File sizes have continued to increase at a faster rate than computing power
- CPU cores have reached the upper limit of frequency
- Distributed computing when implemented efficiently can be faster
- Distributed computing reduces the cost of computing dramatically

2. Project Objective

This project aims to showcase a use-case of Hadoop. In this project, we aim to implement a Hadoop application which can divide and optimise the computation of a simple web-finder program.

3. Outline

In this project we are developing a Hadoop application which applies MapReduce paradigm to the problem of finding closely linked web-sites. We basically have a list of websites, and we want to find the interconnections between these web-sites via hyperlinking. This task is highly optimisable using MapReduce technique of Hadoop. Using MapReduce, we would divide up the tasks into sub-tasks which will be performed at different nodes and then will be combined together to give the final output. This technique leverages all the resources available at our disposal.

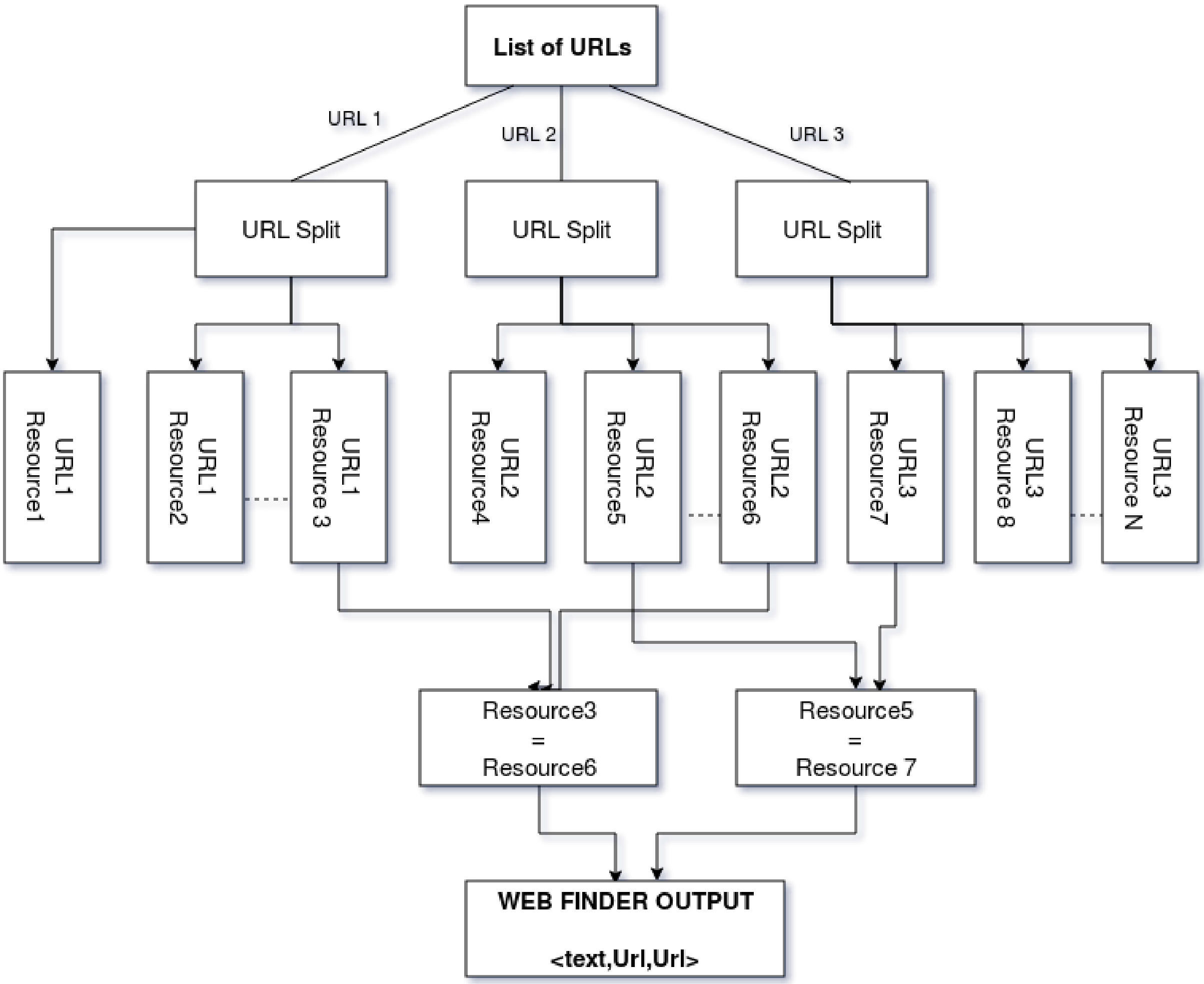

Figure 1: Basic outline of the application

Chapter 1: Apache Hadoop

1. Introduction to Hadoop

Apache Hadoop is a software framework that supports data intensive distributed applications; it enables applications to work with thousands of nodes and petabytes of data.

Hadoop contains:

1. The projects Hadoop Common: the common utilities that support the other Hadoop subprojects.
2. Hadoop distribute file system(HDFS),
3. Hadoop MapReduce: a software framework for distributed processing of large data sets on compute clusters.
4. Avro: A data serialization system.
5. Chukwa: A data collection system for managing large distributed systems.
6. HBase: A scalable, distributed database that supports structured data storage for large tables.
7. Hive: A data warehouse infrastructure that provides data summarization and ad hoc querying.
8. Mahout: A Scalable machine learning and data mining library.

9. Pig: A high-level data-flow language and execution framework for parallel computation.

In our project we mainly focus on the MapReduce framework.

2. MapReduce

MapReduce is a programming model and software framework introduced by Google to support distributed computing on large data sets on clusters of computers. Hadoop MapReduce is an open source project from Apache, it allows us to develop parallel applications without any parallel programming techniques, and the applications could be easy deployed and executed, it works with the HDFS and processes huge datasets (e.g. more than 1TB) on distributed clusters (thousands of CPUs) in parallel.

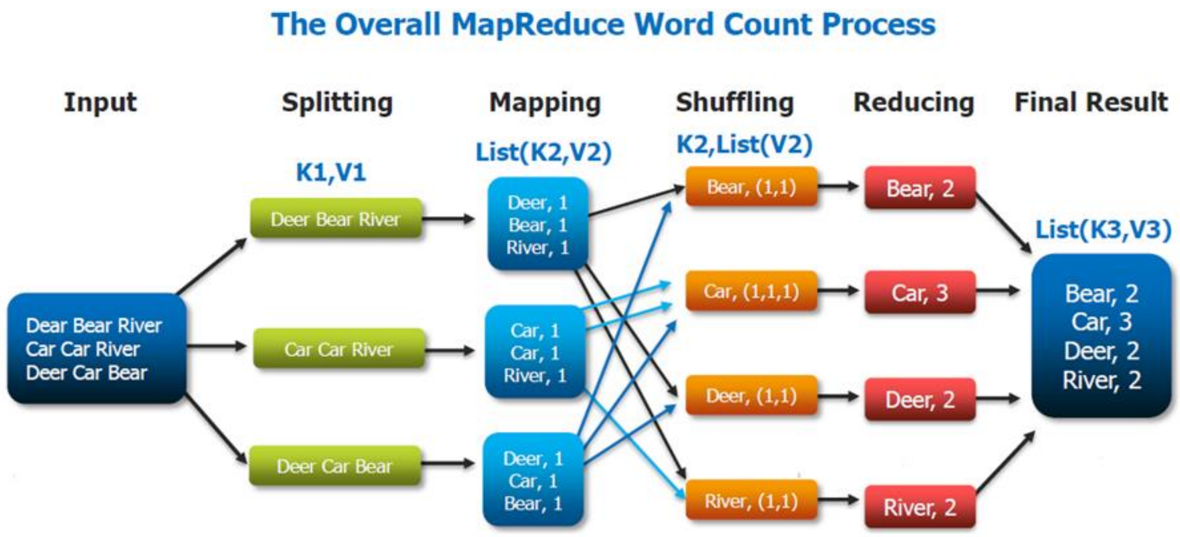

Figure 2: Basic working of MapReduce through simple Word Count example

We can easily understand the working of MapReduce by using word count example. Figure 1. represents the MapReduce solution to the word count problem.

3. Features of MapReduce

MapReduce framework has several advantageous features, some of which are as follows:

- Automatic Parallelization & Distribution
- Fault Tolerance
- Locality Optimization
- Backup tasks

4. Use of MapReduce in our task

Our task was to optimise the task of finding links between websites. In a sequential programming technique, we would need to iterate over each website and find all of the incoming as well as outgoing links. Then we would need to find all the common links and create a set of combinations that have the same source or destination. This is easily a very time taking process if done sequentially.

Further complexity arises when we define depth of the web. That is how far the web crawler will search for an interconnection. When the depth of search is increased to 10, it gets computationally very expensive.

Thus, using MapReduce can help in sharing the load and reducing the time complexity of the overall process. The next chapter is dedicated to our implementation of MapReduce applied to the task.

Chapter 2: Implementation

1. Introduction

We have used Java as the programming language to implement our project. Map-Reduce framework from the Apache Hadoop library has been used to parallelize the task of finding the interconnections between different web sites.

2. Input

The application accepts a list of URLs as input, i.e., one or multiple text file(s) where each line represents a single URL. The program has three parameters, the input folder, the output folder, and the maximum depth limit for spidering the websites. This limit is an optional parameter and 1 by default, meaning that for each base URL, one level of links is followed. If one increases the parameter the case as described in the previous chapter arises, but Hadoop is able to handle it well.

3. Mapper

The mapper receives tuples of the type:

<*Integer (line number), Text (the line contents, i.e., the URL)*>

Each URL is loaded recursively, similar to how a bot from a search engine would do it. When reading a web page (using Java's URLConnection object), we look for links/usages of other resources, such as images, CSS, JavaScript, links, frames, and iframes. The latter three are traced recursively up to a given maximum depth (the third, optional, command line parameter of the program). For each linked resource, the mapper will output a tuple as follows:

<*Text (with the URL to the resource), Text (with base URL from the input*>

4. Reducer

The reducer receives elements of the form:

<Text (with the URL to the resource), Iterable<Text> (all base URLs linking to the resource>

The reduction step is very simple: All input pairs which just contain a single element in the values, i.e., which stand for resources only linked from a single on of the originally specified URLs, are discarded. This leaves only resources linked from multiple input URLs. These resources are thus shared amongst different sites. The reducer returns tuples as follows:

<Text (with the URL to the shared resource), List<Text> (with all originally specified sites linking to the shared resource)>

This will help us to understand how different websites are connected and which resources are vital, i.e., which dependencies are needed by several sites to work properly.

Results

Following are the screenshots at various timepoints during the execution of the application on Ubuntu.

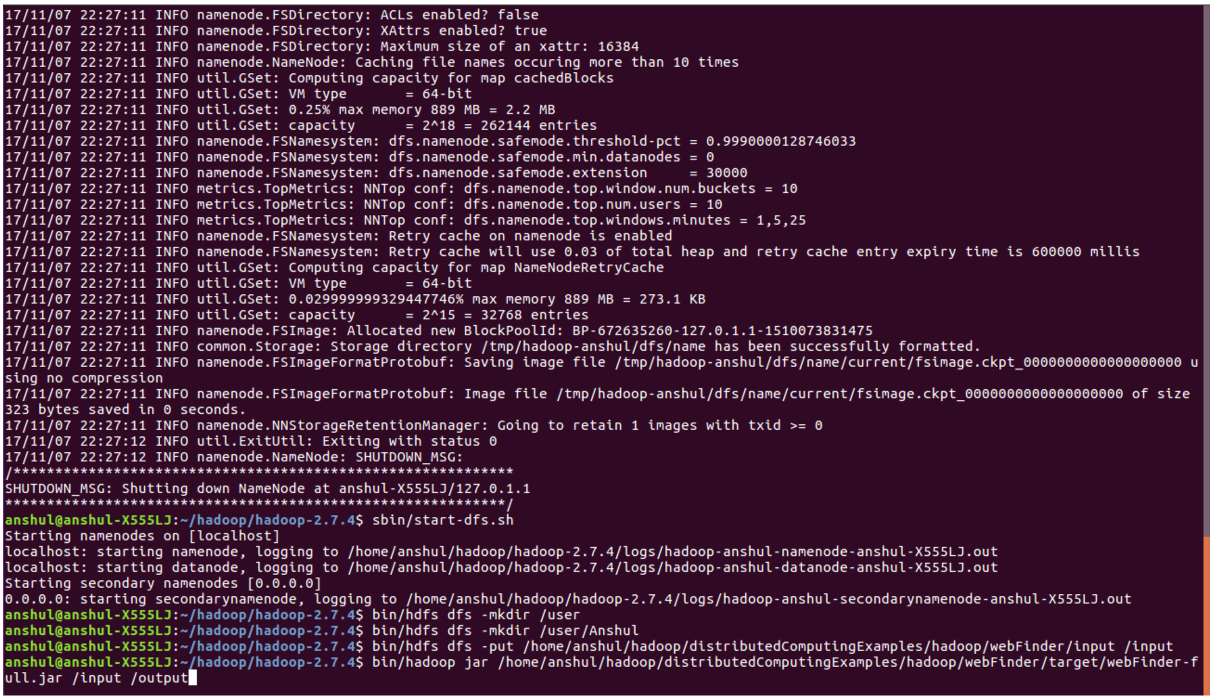

Figure 3: Starting Hadoop Services

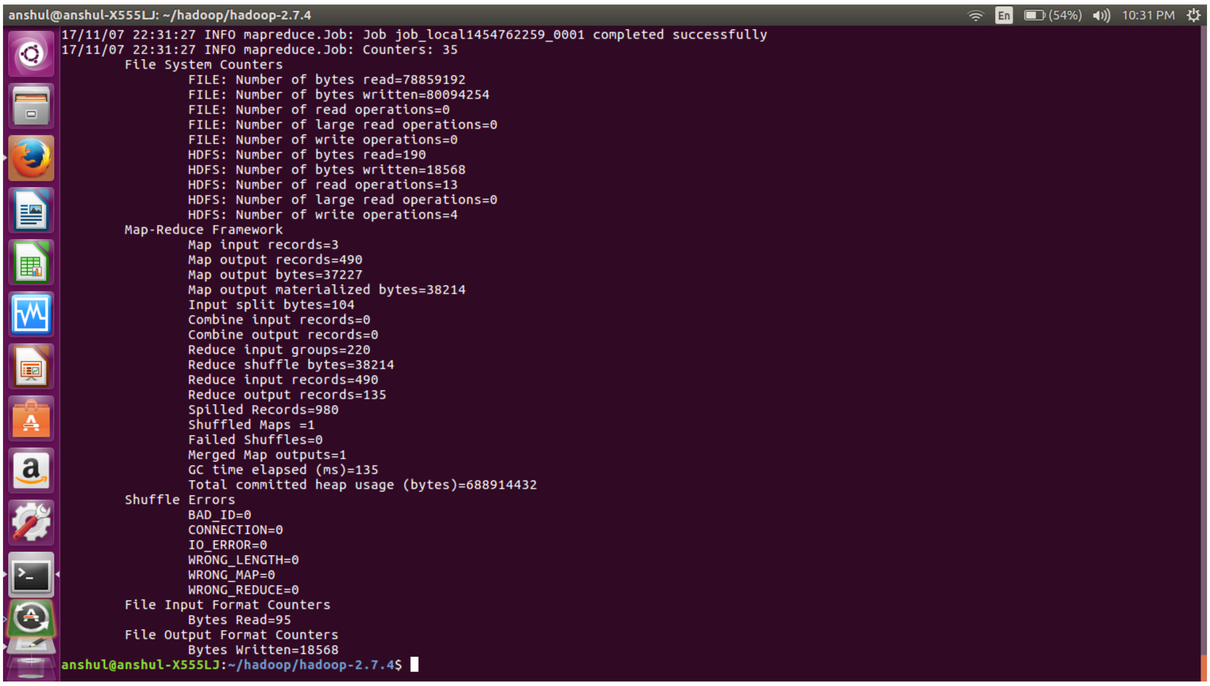

Figure 4: Information of our single node cluster

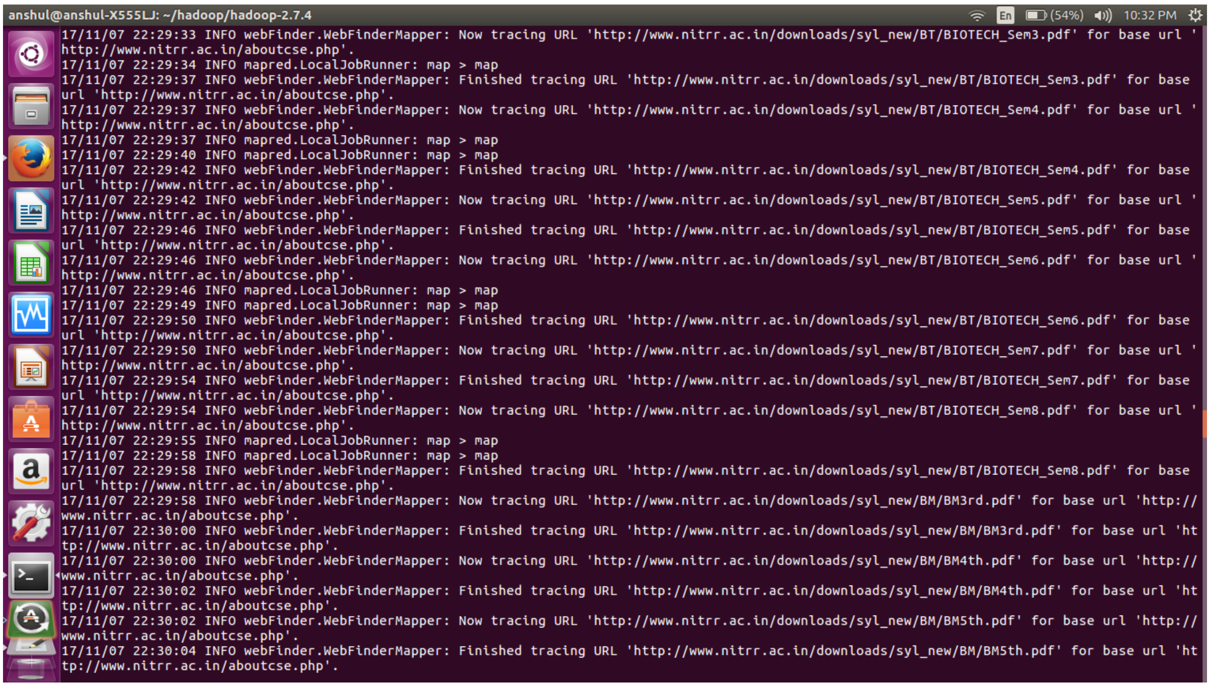

Figure 5: Screenshot during the execution of the code

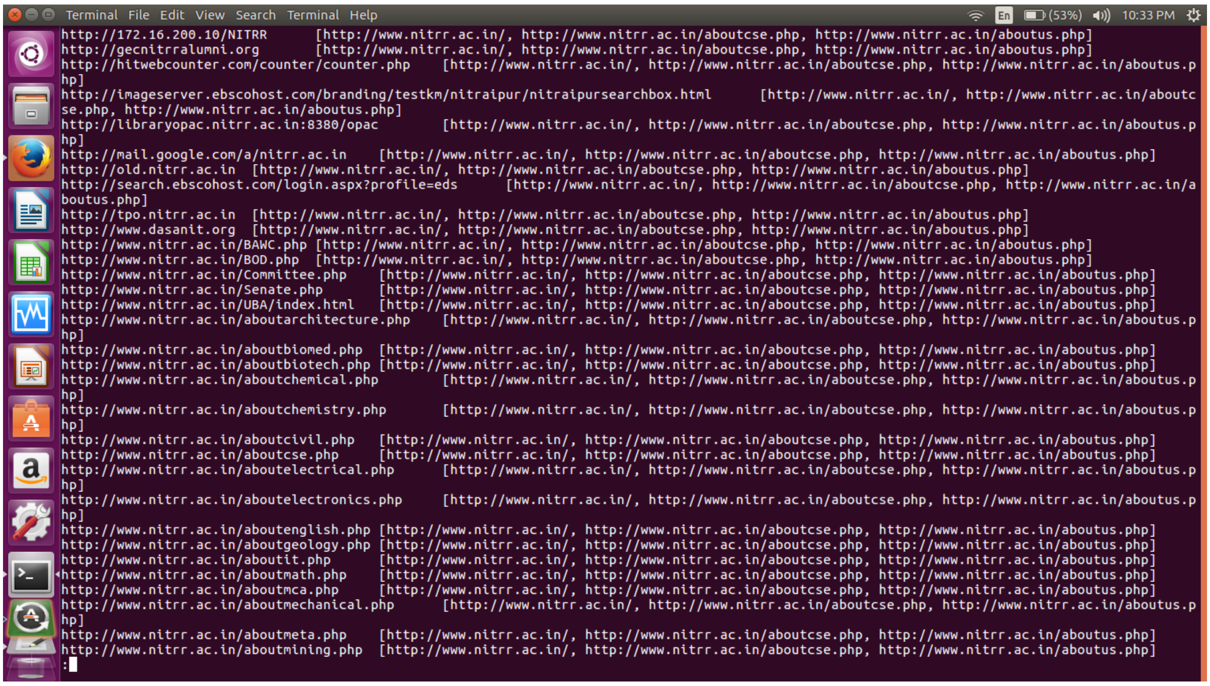

Figure 6: Final Output which shows the interlinking of websites

Conclusion

We have implemented the defined project and have learnt the underlying concepts of Hadoop and its application to distributed computing. We hope to continuously add to this knowledge in the future and tackle tasks that are more difficult to divide and parallelize.

References

[1] Eugenia Sawa. Retrieved from https://blogs.microsoft.com/jobs/story-library/5-reasons-why-distributed-systems-are-the-future/

[2] benjismith, Shane. 2009. Retrieved from https://stackoverflow.com/questions/522438/why-does-moores-law-necessitate-parallel-computing

[3] Chu Yan Shing, Wu Bing Chuan. 2011. *Cloud Computing technologies and applications.* Term Project Report, Department of Computer Science and Engineering. The Chinese University of Hong Kong, Hong Kong, China.

[4] StackOverflow. 2017. Retrieved from https://stackoverflow.com/

[5] Github. 2017. Retrieved from https://github.com/ [for various project parts]