Interim Report

Vinay Kakkar

December 5, 2022

Contents

Abstract

1 aims, objectives and literature survey

1.1 aims

The aim provided in the Project description: Provide a framework where large numbers of protein structures can be analysed using a user-provided executable in the MapReduce formalism.

This aim is very benifitial for the biology sector as this will help them perform large scale data analysis on to protien structures. This is very helpful as it allows biologists to be able to learn from the anlysis about protein strucutres functionality and behaviours. Finaly this will allow biologists to be able to create a new drugs which can help with future or current desises called new drug discovery.

Being able to conduct large scale data processing means that given lots of protien structures the user would be able to to pass each protien structre into a executable that the user can choose that will return a result from the executable for each protien structre that is passed in. More specifically using Hadoop/Spark with python to store these protien data and processes the data in a parallel so that multple protiens can be processed at the same time giving better speeds and performance.

The aim of this program is to build it as a framework that has multiple user executables which can be picked by the user. So this program should be able to handle the protien data provided by the protein data bank and most the file types provided by it. The user should also be able to choose what protien data it wants to send into the framework.

I will aim to complete this by using batch processing and mapreduce formalism. Where i can have multiple computers procssing this data in an automated process.

Be able to effeciently perform executables on PS data ..who? ..what? ..where? ..why? ..how? is objectives

1.2 objectives Completed

Tutorials: Explain the prrof of concepts

In order to complete the projects aim i need to understand some technologies. I went through a few tutorials which are named in my readme file within gitlab.

The first two tutorials showed me how to set up pyspark and how to use spark and python together.

The next tutorial tought me how to conduct mapredcue within pyspark on spark dataframes.

Lastly the last tutorial showed me how to distribute these dataframes on a cluster by turning the dataframes into a RDD which is then be put on to a distributed cluster.

In order to complete this project i need to show some proof of concepts. The main proof of concepts are split into two parts.

First is to prove that we can distribute files provided from a pdb bank into a distrubuted cluster. In order to this we need to be able to read a .pdb file and convert it into a dataframe and then into a RDD

Second proof of concept program is that we can run these clusters that contain the data from pdb into a user executable. This is completed by collecting all the data from a rdd back into dataframes and then pack into pdb filrs which we are able to then run user executables on.

Reports

In order to understnad the data i am working with i needed to complete reports on protein structures. Which gave me in depth knowladge on protiens and large scale expressions. This gave me the extra knowladge to be able to

The other main subject of reasearch is mostly to do with protein data bank which provides the data files and the file formats that come with it. This is important as it will be the files types i will be working with and that i will need to put these files types into a cluser and be used with pyspark. The Protein Data Bank and the file formats used in accessing it.

Proof of Concepts

Explain how the program works and how that means we achived this proof. distributing protein structure files amongst MapReduce cluster. Running MapReduce using a single type of executable for analysing protein structures.

1.3 Futher Objectives

Provide the basis for a UI that a user can use to be able to conduct a user execuatble onto a protien structure.

Be able to perform/use user executables that have been created to be perfomred on pdb files that will result in an output.

Reasearch on how to distribute clusters to multiple computers allowing them to comunicate and perform actions and achive goals needed by working together.

Reasearch on how to conduct benchmarking comparing the speeds difference when trying to accomplish the same task but not using batch processing.

Final delievarbles

The final framework will allow users to input a map and reduce step (namely an executable to run on an individual protein structure in the former and a parser for the generated log file from the reduce step) to analyse a set of protein structures.

The developer will provide a clear set of guidelines for the map/reduce executables.

The final data should be returnable in variety of different formats.

Extensions

The PDB is updated on a regular basis. The data set can mirror the official database and is distributed amongst the MapReduce cluster.

A set of libraries so that arbitrary executables can be easily put into the MapReduce formalism. The PDB is comprised of many different types of structure (DNA and RNA macro-molecules are included as well as proteins). An interface to select fractions of the PDB for analysis.

Implementation on a Public Mapreduce cluster e.g. Amazon.

2 literature survey

I completed reports as a summery of my research which was needed during the first part of my project.

2.1 Protein Structures

Add info from Protein Structures report.

2.2 The Protein Data Bank and the file formats used in accessing it

Add info from the Protein Data Bank and the file formats used in accessing it report

2.3 Hadoop spark and pyspark

What is Hadoop

Hadoop is a open source framework for writing and running distriubted applications that process large amounts of data. Hadoop provides key aspects making it valuable such as: 1.Accessable 2.Robust 3.Scalable 4.simple [Lam10].

Apache Hadoop[8] consists of a filesystem(HDFS) and a MapReduce engine. Hadoop cluster consists of a single master node and many worker nodes. The master node provides instructions to the slave nodes and computations are performed on the slave nodes. Copies of the same data exist in different slave nodes ensuring a fault tolerant working[12]. The default factor is 3 [HRJ17].

Mapper

Input key/value pairs are mapped to a set of intermediate key/value pairs. Mapper then sorts the key value pairs by the keys. Partitioners are mainly responsible for providing intermediate key/values to the reducers[11] [HRJ17].

Reducer

Firstly, shuffler combines data having the same key from different map functions. The intermediate values having the same key are reduced to smaller set of values and output is produced [HRJ17].

What is Spark

Apache Spark is a popular open-source platform for large-scale data processing that is well-suited for iterative machine learning tasks [MBY+16].

Spark is a fault-tolerant and general-purpose cluster computing system providing APIs in Java, Scala, Python (pySpark), and R, along with an optimized engine that supports general execution graphs. Moreover, Spark is efficient at iterative computations and is thus well-suited for the development of large-scale machine learning applications [MBY+16].

Spark is a lightning fast and general engine used for analyzing large scale data stored across a cluster of computers providing the programmers with an application program interface. Spark uses in-memory cluster computing which is its most important feature for increasing the processing speed of an application. Spark[2] gives high performance, is easy to use, executes everywhere. It also combines SQL streaming[4] and complex analytics [HRJ17].

Spark vs Hadoop

Hadoop Map Reduce	Spark
For Applications that repeatedly reuse the same set of data, map reduce is very inefficient.	Spark uses in-memory processing, reusing it for faster computation.
MapReduce is quite faster in batch processing.	As memory size is limited, it would be quite slower in batch processing of huge data set.
Data is stored in disk for processing.	Data is stored in main memory. As it is an inmemory computation engine entire data is copied.
Difficulty in processing and modifying data in real time due to its high latency.	Used to process and modify data in real time due to its low latency.
Predominantly used to process from bygone datasets.	Predominantly used for streaming, batch processing and machine learning
For fault tolerance, MapReduce uses replication.	For fault tolerance, Spark uses RDDs.
It merges and partitions shuffle files.	It does not merges and partition shuffle files.
Primarily disk based computation.	Primarily RAM based computation.

Table 1: Showing the differences between haddop and spark [HRJ17].

Summerysing the results it show Spark to be quicker in both experiments. Spark also provides an api for python which will be very helpful in this project seeing its easy nature to be able to read files and work with text based files. Therfore i have decided to work with Pyspark for this project.

Number of words	Hadoop (Sec)	Spark(Sec)
100	79	28.841
1000	91	31.185
10000	96	35.181
100000	103	36.969
1000000	116	39.569

Table 2: Comparision of Execution time for wordcount program [HRJ17].

Number of words	Hadoop (Sec)	Spark(Sec)
5	2.541	0.9030
10	3.370	1.459
50	6.420	2.840
100	9.383	3.452
200	10.100	5.749

Table 3: Comparison of Execution time for logistic regression program [HRJ17].

Software Architectural Bottlenecks

HDFS has scheduling delays in the architecture this resuls in cluster nodes waiting for new tasks. This is due to the fact the he access pattern is periodic. On top of this, when tasks are available for computation, the HDFS client code, serializes computation and I/O instead of decoupling and pipelining those operations. Data prefetching does not improve performance, even though MapReduce streaming access pattern is highly predictable [SRC10].

definition HDFS The Hadoop Distributed File System (HDFS) is a distributed file system designed to run on commodity hardware

Portability Limitations

Some performance-enhancing features in the native filesystem are not available in Java in a platform-independent manner. This includes options such as bypassing the filesystem page cache and transferring data directly from disk into user buffers. As such, the HDFS implementation runs less efficiently and has higher processor usage than would otherwise be necessary [SRC10].

Portability Assumptions

The classic notion of software portability is simple: does the application run on multiple platforms? But, a broader notion of portability is: does the application perform well on multiple platforms? While HDFS is strictly portable, its performance is highly dependent on the behavior of underlying software layers, specifically the OS I/O scheduler and native filesystem allocation algorithm [SRC10].

3 Software Development

Show case how we used TDD to develop the program.

4 planning and time-scale

4.1 planning

4.2 time-scale

5 summary of completed work

6 diary

WeekOf:

10th october 2022

Made notes on large scale gene expression Very confused on large scale gene and Protein expressions maybe better to start looking into protien structers chapter first Investigated into hadoop and spark Questions asked in meeting with supervisor: 1. Do i need to look into experimental techniques used in labs to analyse protien structure 2. what are Protein folds 3 Is it important to understand the chemical physical properties of amino acids 4. Do i need to read into peptide bonds 5. When is the fine line to stop looking into the biology side of things

17th october 2022

24th october 2022

31th october 2022

7th october 2022

14th october 2022

21th october 2022

Bibliography

[HRJ17] Akaash Vishal Hazarika, G Jagadeesh Sai Raghu Ram, and Eeti Jain. Performance comparision of Hadoop and spark engine. In *2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, pages 671–674, February 2017.

[Lam10] Chuck Lam. *Hadoop in Action*. Simon and Schuster, November 2010. Google-Books-ID: 8DozEAAAQBAJ.

[MBY$^+$16] Xiangrui Meng, Joseph Bradley, Burak Yavuz, Evan Sparks, Shivaram Venkataraman, Davies Liu, Jeremy Freeman, D. B. Tsai, Manish Amde, Sean Owen, Doris Xin, Reynold Xin, Michael J. Franklin, Reza Zadeh, Matei Zaharia, and Ameet Talwalkar. MLlib: Machine Learning in Apache Spark. *Journal of Machine Learning Research*, 17(34):1–7, 2016.

[SRC10] Jeffrey Shafer, Scott Rixner, and Alan L. Cox. The Hadoop distributed filesystem: Balancing portability and performance. In *2010 IEEE International Symposium on Performance Analysis of Systems & Software (ISPASS)*, pages 122–133, March 2010.