CONSISTENT MODEL ARCHITECTURE FOR SCALABLE INFRASTRUCTURE SYSTEMS FOR BIG DATA PROCESSING

Nader Mir, Prita Nigam, Navyatha Marreddy

Department of Electrical Engineering, San Jose State University, San Jose, California 95192.

Introduction

With the growing use of technology and computers that minimise the human work, there is a new challenge that has attracted a lot of attention in the past few years. The computers are continuously generating a lot of data. At each of the organizations and at every step the data has to be stored safely and effectively without the fear of loss of data. This data has been growing many folds and very rapidly. The huge amount of incoming data is thus vital to the organization and therefore has to be processed almost immediately to avoid any loss or congestion of data. Most of the data incoming is not a structured data with a specific type for example tweets, blogs, reports are weakly structured text, the images and videos are made for storage and display. The biggest challenge comes in when the data has to be transformed into a structured format for easy and accurate analysis so that it can be automatically allied with the previously created data in a systematic format.

Big Data is the most well known word heard in the business. Big Data is nothing but only vast information that is being produced from different sources. The information is in such a gigantic sum, to the point that it can't be handled by the current innovation or frameworks. This test of handling gigantic information has prompted different tasks and research in the business. One of the significant issues numerous organizations are confronting today is the enormous measure of information that is being created.

Hadoop

Hadoop is an Apache based project, an open-source software for reliable, distributed computing. It is an open source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware (which indicates the use of already existing computing components for parallel processing to get the greatest amount of useful computation at low cost).

Map Reduce Algorithm is one of the important algorithm, where the basic idea is to map the tasks to the computing nodes (which are called name nodes in case of Hadoop, which are responsible for processing of the tasks and returning the results) and these tasks are kept of note by the Job Tracker in case of Hadoop. Once, the tasks are processed by the computing nodes, the results are returned and reduced to the final result by summing them up in a required manner.

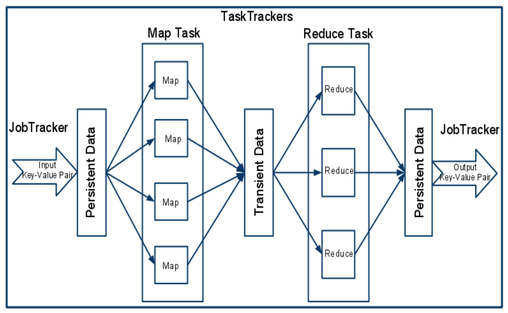
Also, there are many issues to be handled like the contention for CPU, contention for memory and for I/O.

Methodology

So, the motivation behind our project is the major problems faced by the industry: Big Data. Although, we have considered the model of Hadoop as our base, we have our own design. Also, we would be limiting ourselves to solve a couple of sample problems in our project namely:

- **Counting number of words in a dictionary**
- **Portfolio Pricing**

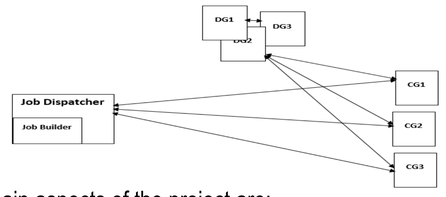
We will be storing the data whatever is required for the task in a **Data Grid**. So, as a part of it, we will be implementing the Distribution Map for Data Grid, which will distribute the data among the Data Grid cluster using UDP. The elements of this Grid will be assigned a multicast group address, which is used to communicate with the Data Grid. As we know, the data access from RAM is faster than accessing over network than accessing through a hard drive. There will be I/O contention, if the read write cycles are slower. So, as we know reading or writing from RAM is faster, we would be storing our data whatever we process in RAM. The Data Grid will be designed in such a way that if one component in data grid exceeds the capacity of data that it can store/handle, then it sends the remaining data to other elements of the Data Grid using UDP.



Another major component is the **Computing Grid**, where the tasks are processed. Each CG (Computing Grid) node, processes the task given to it (it is given the task by the job dispatcher as explained below).

We will be implementing a **Job Dispatcher**, which is the main component of our project. The job dispatcher is given the input task and it divides the task in an optimal manner (which is the Map phase) and distributes the tasks to the Computing Grid nodes (Note that a TCP connection is opened with the Computing Grid node in order to communicate with it). So, this is where we implement the Map Reduce algorithm in order to be able to divide the tasks and sum them up once the processing is done.

Each CG node processes the task given to it. As a part of processing, it fetches the data whatever is needed from the Data Grid over UDP. Once the Computing Grid nodes finish processing the task given to them, they send their individual results back to the Job Dispatcher and close the TCP connection. The Job dispatcher then reduces all the result and gives the final result to the application.



The main aspects of the project are:

1) We should be able split the big CPU centric job into smaller jobs.
2) We should be able to utilize computing grids (where the job execution happens) effectively in terms of CPU
3) We should be able to build the jobs according to number of Computing grids available to use them effectively
4) Reduce the job input data as much as possible (to avoid wire transfer across network)
5) utilize RAM as much as possible for faster response, for getting data and putting results back
6) Minimize I/O operations as much as possible

Solution and Results

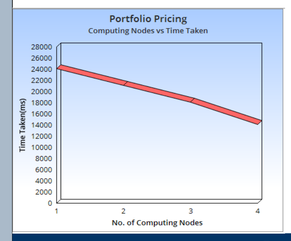
1) Hydrate the Task relevant input data into data grid (JVM which is responsible to serving data to Computing grids)
2) Allow data distribution, as data may not be able to fit onto single RAM, to scale the data grids.
3) give the flexibility the Task builder to decide the Job building algorithm.
4) While building jobs we need to ensure that each task that we create carries just the meta data, not the real data (to avoid I/O condense on network.
5) We should be able to run possible aggregations on computing grids itself to use CPU effectively
6) We should be able to scale the computing grids without compromising on any other factors (such as input data size, and the complexity of the task etc.)

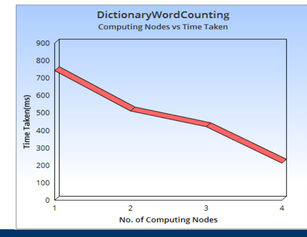
Results for counting words in a dictionary:

No of Computing Grid Nodes	No of Data Grid Nodes	Time Taken For Processing Task
1	1	780ms
2	2	620ms
3	2	300ms
4	2	210ms

Results for Portfolio Pricing:

No of Computing Grid Nodes	No of Data Grid Nodes	Time Taken For Processing Task
1	1	28050ms
2	2	19100ms
3	2	17202ms
4	2	14020ms





Summary

we were successfully able to build a framework, that effectively divides the input tasks and distributes them to the computing nodes. The computing nodes do a micro-aggregation that saves time during the final aggregation at the job-dispatcher. We can scale our model to any number of computing grid nodes and data grid nodes. Also, the model shows effective results for the time taken for processing our problem statements (counting the number of words in dictionary and portfolio pricing). As stated in one of our aspects that we considered for our project (the I/O contention), we tried to reduce the I/O operations by using the logic of keeping the data in memory, instead of writing it to the disk, which involves a lot of overhead (while serializing and de-serializing the data). Our model can be used to process big data tasks that are CPU centric and can be scaled to any number of nodes and the data results will be consistent.

Key References

[1] L. Neumeyer, B. Robbins, A. Nair and A. Kesari "S4: Distributed stream computing platform", *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, pp.170 -177

[2] M. H. A. Wahab, M. N. H. Mohd, H. F. Hanafi and M. F. M. Mohsin "Data pre-processing on web server logs for generalized association rules mining algorithm", *World Acad. Sci., Eng. Technol.*, vol. 48, pp.970 2008

[3] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. USENIX OSDI, 2004

Acknowledgements

We would like to thank our Graduate advisor Dr. Nader F. Mir for his guidance and support in completing our graduate project and to the Department of Electrical Engineering for providing the Laboratory facilities and all assistance which made this project a successful one.