

TWITTER TREND ANALYSIS USING MAPREDUCE

Renuka. K

**MTech, CSE, KMIT
renukakurma@gmail.com**

ABSTRACT

Successful businesses know the significance of following current trends. Monitoring the latest trends plays a critical role in almost every aspect of marketing. It helps a business stay relevant and establish a relationship with consumers. But, rapidly changing interests can make keeping up with trends pretty difficult. A decade ago, businesses needed access to expensive resources to track consumer trends.

Today, we have access to the same type of resources that rival their predecessors in not only cost, but in content and relevancy as well. Today's resources give access to present-day trends on a daily or even hourly basis. What's more, they're a simple click away. Twitter's trending topics is a prime example resource.

Through this free resource, you can instantaneously identify what's buzzing around both the nation and the world through the hash-tag or the "Trends" box.

hence in this project we will try to generate global trends on some real time data logs using mapreduce techniques.

1. INTRODUCTION

Successful businesses know the significance of following current trends. Monitoring the latest trends plays a critical role in almost every aspect of marketing. It helps a business stay relevant and establish a relationship with consumers. But, rapidly changing interests can make keeping up with trends pretty difficult. A decade ago, businesses needed access to expensive resources to track consumer trends.

Today, we have access to the same type of resources that rival their predecessors in not only cost, but in content and relevancy as well. Today's resources give access to present-day trends on a daily or even hourly basis. What's more, they're a simple click away. Twitter's trending topics is a prime example resource.

Through this free resource, you can instantaneously identify what's buzzing around both the nation and the world through the hash-tag or the "Trends" box.

hence in this project we will try to generate global trends on some real time data logs using mapreduce techniques.

2. Introduction to Hadoop

Hadoop was chosen for this thesis for several reasons. First, it is popular and widely used by a number of leading organizations including, Amazon, Facebook, Google, Yahoo! and many others. Second, it is designed for commodity hardware, significantly lowering the cost of building the cluster. Third, Hadoop is an open source technology developed in Java, making it easier to obtain, distribute and modify whenever necessary. Its effective use, lower cost and easy access makes it a potentially stable base as a large scale storage system for future technologies. Thus, research into the architecture of Hadoop file system should help data intensive applications with their rapidly growing storage needs.

3. Introduction to HDFS

Implementation of Hadoop is carried out in two main service components of the master / slave architecture. The file system metadata is decoupled from its actual data located on an individual NameNode machine. Decoupling provides flexibility to the architecture to accommodate more DataNodes in the cluster. Hadoop was chosen for this thesis for several reasons. First, it is popular and widely used by a number of leading organizations including, Amazon, Facebook, Google, Yahoo! and many others. Second, it is designed for commodity hardware, significantly lowering the cost of building the cluster. Third, Hadoop is an open source technology developed in Java, making it easier to obtain, distribute and modify whenever necessary. Its effective use, lower cost and easy access makes it a potentially stable base as a large scale storage system for future technologies. Thus, research into the architecture of Hadoop file system should help data intensive applications with their rapidly growing storage needs.

3.1 Data Node

Hadoop Distributed File System (HDFS) serves as the large scale data storage system.

Similar to other common file systems, the HDFS supports hierarchical file organization. The Name Node splits large files into fixed sized data blocks which are scattered across the cluster. Typically the data block size for the HDFS is configured as 128MB. Since HDFS is built on commodity hardware, the machine failure rate is high. In order to make the system fault tolerant, data blocks are replicated across multiple Data Nodes. HDFS provides replication, fault detection and automatic data block recovery to maintain seamless storage access. By default replication takes place on three nodes across the cluster. When a client tries to access the failed Data Node, the Name Node maps the block replica and returns it to the client. For achieving the high throughput, the file system nodes are connected by high bandwidth network.

3.2 Name Node

The NameNode maintains the file system metadata as the HDFS directory tree and operates as a centralized service in the cluster. It controls the mapping between file name, data block locations and the DataNodes on which data blocks are stored. It also writes the transaction logs to record modifications in the file system. Clients communicate with the NameNode for common file system operations such as open, close, rename and delete. The namespace is a live record of the HDFS located on the centralized NameNode server. It is a directory tree structure of the file system which documents various aspects of the HDFS such as block locations, replication factor, load balancing, client access rights and file information. The namespace serves as a mapping for data location and helps HDFS clients to perform file system operations.

The metadata is stored as a file system image (fsimage) file which is a persistent checkpoint of the file system. The edit log records the write operations

submitted by the file system clients. When the edit log size exceeds a predefined threshold, the NameNode moves the transactions into live memory and apply each operation to the fsimage.

A backup of the namespace is periodically stored on the local disk of the NameNode and synchronized with a secondary master node as a provision against NameNode failure.

3.3 Job Tracker

The Job Tracker is the service within Hadoop that farms out Map Reducetasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack.

1. Client applications submit jobs to the Job tracker.
2. The Job Tracker talks to the Name Node to determine the location of the data. The Job Tracker submits the work to the chosen TaskTracker nodes.
3. The TaskTracker nodes are monitored. If they do not submit heartbeat signals often enough, they are deemed to have failed and the work is scheduled on a different Task Tracker.
4. A Task Tracker will notify the Job Tracker when a task fails. The Job Tracker decides what to do then: it may resubmit the job elsewhere, it may mark that specific record as something to avoid, and it may even blacklist the Task Tracker as unreliable.
5. When the work is completed, the Job Tracker updates its status.

Client applications can poll the Job Tracker for information.

The Job Tracker is a point of failure for the Hadoop Map Reducetask. If it goes down, all running jobs are halted.

3.4 Task Tracker

A TaskTracker is a node in the cluster that accepts tasks - Map, Reduce and Shuffle operations - from a JobTracker.

Every Task Tracker is configured with a set of slots; these indicate the number of tasks that it can accept. When the JobTracker tries to find somewhere to schedule a task within the Map Reduce operations, it first looks for an empty slot on the same server that hosts the Data Node containing the data, and if not, it looks for an empty slot on a machine in the same rack. The TaskTracker spawns a separate JVM processes to do the actual work; this is to ensure that process failure does not take down the task tracker. The TaskTracker monitors these spawned processes, capturing the output and exit codes. When the process finishes, successfully or not, the tracker notifies the JobTracker. The TaskTrackers also send out heartbeat messages to the JobTracker, usually every few minutes, to reassure the Job Tracker that it is still alive. These messages also inform the Job Tracker. The JobTracker will first determine the number of splits (each split is configurable, ~1664MB) from the input path, and select some TaskTracker based on their network proximity to the data sources, then the JobTracker send the task requests to those selected TaskTrackers.

Each TaskTracker will start the map phase processing by extracting the input data from the splits. For each record parsed by the "Input Format", it invoke the user provided "map" function, which emits a number of key/value pair in the memory buffer. A periodic wakeup process will sort the memory buffer into different reducer node by invoke the "combine" function. The key/value pairs are sorted into one of the R local files (suppose there are R reducer nodes). When the map task completes (all splits are done), the Task Tracker will notify the Job Tracker. When all the Task Trackers are done, the JobTracker will notify the selected Task Trackers for the reduce phase.

Each TaskTracker will read the region files remotely. It sorts the key/value pairs and for each key, it invokes the "reduce" function, which collects the key/aggregatedValue into the output file (one per reducer node).

Map/Reduce framework is resilient to crash of any components. The JobTracker keep tracks of the progress of each phases and periodically ping the TaskTracker for their health status. When any of the map phase TaskTracker crashes, the JobTracker will reassign the map task to a different TaskTracker node, which will rerun all the assigned splits. If the

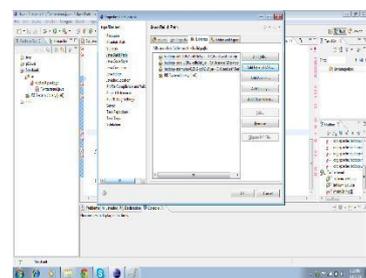
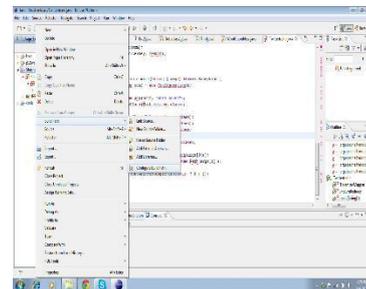
reduce phase TaskTracker crashes, the JobTracker will rerun the reduce at a different TaskTracker.

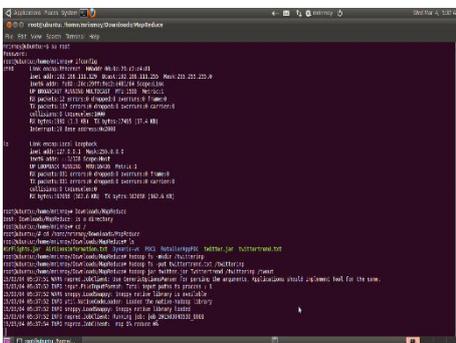
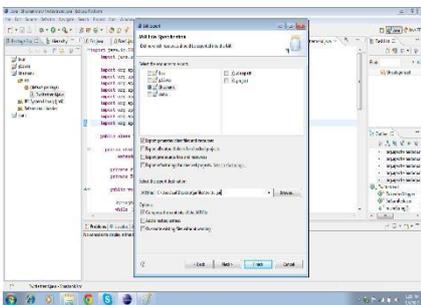
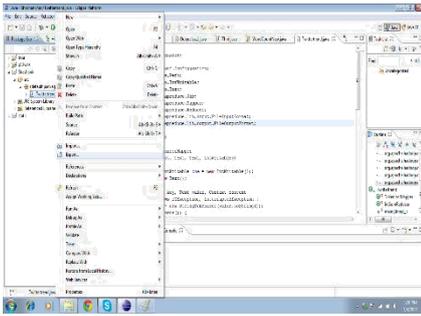
After both phases completes, the JobTracker will unblock the client program.

4. Proposed Model

Today, we have access to the same type of resources that rival their predecessors in not only cost, but in content and relevancy as well. Today's resources give access to present-day trends on a daily or even hourly basis. What's more, they're a simple click away. In addition, basic factors for designing word occurrences such as storage technique, index size, look up speed, maintenance of large data; fault tolerance etc. will also take into account. Building a local index for files within those systems will improve to be a great way to solve problems that could arise in the search system.

5. Screenshots





6. REFERENCES

- [1] Bowman, M., Debray, S. K., and Peterson, L. L. 1993. Reasoning about naming systems. .
- [2] Ding, W. and Marchionini, G. 1997 A Study on Video Browsing Strategies. Technical Report. University of Maryland at College Park.
- [3] Fröhlich, B. and Plate, J. 2000. The cubic mouse: a new device for three-dimensional input. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems
- [4] Tavel, P. 2007 Modeling and Simulation Design. AK Peters Ltd.Sannella, M. J. 1994 CSatisfaction and Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
- [5] Forman, G. 2003. An extensive empirical study of feature selection metrics for text classification. J. Mach. Learn. Res. 3 (Mar. 2003), 1289-1305.
- [6] Brown, L. D., Hua, H., and Gao, C. 2003. A widget framework for augmented interaction in SCAPE.
- [7] Y.T. Yu, M.F. Lau, "A comparison of MC/DC, MUMCUT and several other coverage criteria for logical decisions", Journal of Systems and Software, 2005, in press.
- [8] Spector, A. Z. 1989. Achieving application requirements. In Distributed Systems, S. Mullender