World Applied Sciences Journal 32 (12): 2436-2441, 2014
ISSN 1818-4952

DOI: 10.5829/idosi.wasj.2014.32.12.1322

Using Supervisor Selection Algorithm for Effective Big Data Processing in Storm

V. Manoj kumar, J.P. Nivash, M. Nirmala, L.D Dhinesh Babu and Ebin Deni Raj

School of Information Technology and Engineering, VIT University, Vellore, India

Abstract: Emergence of new technologies and improvements in networking provide an effective environment where people can communicate among themselves globally. Social networking sites offer a good interface in connecting people. One can share and post their data stored online. The amount of data stored and processed in those social networking sites grow at an extremely fast pace. Big Data processing systems like Hadoop, Splunk, MongoDb etc., are widely used in storing and management of data. This paper focus on Storm, one of the Big Data processing system and comparison is made with Apache MapReduce, YARN and Akka. We have proposed a supervisor selection algorithm which is capable of increasing the efficiency of storm data processing.

Key words: Storm · Big Data processing · Akka · Superviser selection algorithm · MapReduce · YARN

INTRODUCTION

The evolution of World Wide Web incorporates various services which can be acquired through internet [1]. Information which is published on web can be viewed by anyone through a browser. A browser serves as an interface between the user and the web, displaying web pages and their information. Many websites are hence developed to provide useful information to meet the user needs. The client server architecture provides an efficient arena in accessing and transferring information across various users and servers. The social media applications [2] are widely accessed by the users which in turn generate huge amount of data within a very short span of time. The information and data available on web increase day by day as the number of web user increase. Eventually, huge amount of data gets accumulated in web. Content generation and content sharing have increased in multitudes with the increased use of social networking sites [3]. Various social networking companies like Facebook, Twitter, Google plus etc., pull out huge amount of the data generated by their site by means of content generation, sharing and storing of data. The main aim is to efficiently store meaningful information or data which are uploaded online. The data generated is huge and in different formats hence, data processing methods and strategies can't suit in handling those data. Various technologies like Hadoop, MongoDb, Apache Storm solve these issues with its effective data processing techniques, storage, management and analysis of huge amount of data. The semantic web technology allows to process and read data automatically [4].

Storm: Storm is a distributed, real time high computation system which supports stream processing in Hadoop [5]. In batch processing system the data is grouped in to blocks and the blocks are sent one by one for processing or storage [6]. But in stream processing the data is continuously processed or stored. As the computation is done on data which is most frequent, the results obtained after computation are accurate. Storm is very much efficient in data storage processing and management. Fig 1.1 defines the architecture of storm. The processing speed is much more when compared to batch processing systems because in batch processing the data in disk is taken to primary memory and then processed, where as in stream processing all the data are freshly available in primary memory can be executed or processed soon.

Components of Storm: Storm has two nodes master node (Nimbus) and slave node (supervisor) [7]. Nimbus serves as an admonisher, assigns task to machines for execution and handles distribution of code. Supervisor executes the worker process that is pointed by the Nimbus. In other words, worker performs according to the instruction of the master node Nimbus. The skillful and effective interaction

Corresponding Author: Ebin Deni Raj, Research Associate, School of Information Technology, VIT University.

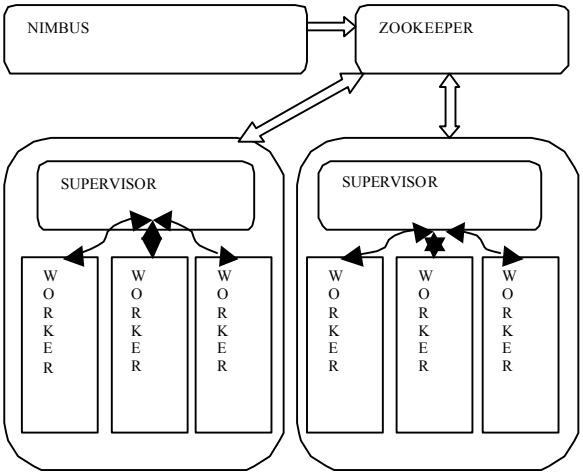

Fig. 1.1: Storm Architecture

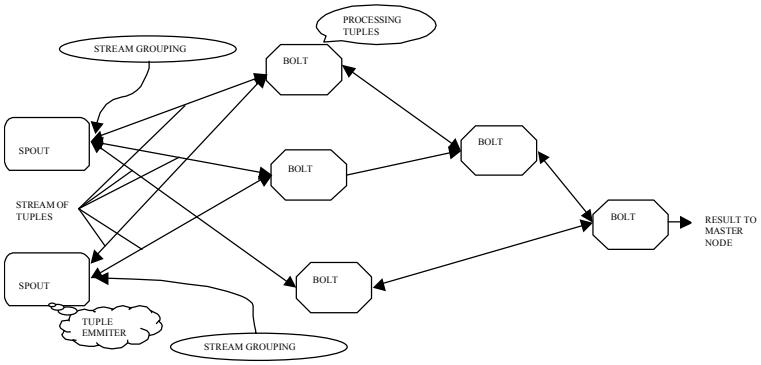

Fig. 1.2: Working of Storm

of movements between Nimbus and Supervisor are interfaced by Zookeeper. Zookeeper serves as a repository maintaining, configuration, information and providing group services.

Operational Modes Of Storm: Storm clusters can be deployed in Local and Remote modes. Basically Storm clusters are developed in local mode because it allows the possibility of deploying all software life cycle process including development and testing which can be hosted by using a single machine.

Elements of Storm: Hence, it is possible to test the working of the topology with changing storm configurations, working of bolts etc. In Remote modes the storm topology which is developed is deployed in distributed clusters where processes are executed in different machines. It is difficult to acquire configuration information during debugging since clusters are distributed. Thus it is always better to test all possibilities of errors before deployment.

The main components of storm are Tuple, Stream, Spout, Bolt and Topology. *Tuple* is the basic data in different formats. Sequence of tuple form streams and Spout generates streams.

Bolt is the area where data processing is done. *Topology* is the design structure of spouts and bolts which decides the work flow of the system.

Working of Storm: The data or the working set on which analysis is performed will be taken from various sources which may be distributed globally will be fed in to spout as shown in Fig 1.2

Spout- Message Emitter: Spout, which is the main data source emit tuples, which in turn are made into stream, that is, the tuples are continuously forwarded to storm's topology as streams. The spout takes responsibility of emitting streams. To increase the reliability of the tuple during the emission period every tuple is assigned with the tuple id and message ids are also given with encryptions. Many or single stream generated by spout

reaches the bolt for storage or processing. But before this step stream grouping takes place, which decides the selection of bolt by streams.

Stream Grouping Techniques: Stream grouping are of different types shuffle, field, all, custom, direct and global grouping.

Shuffle Grouping: Allows the tuples to choose bolts randomly such that the bolts always receive same type of tuples. This type of grouping is mostly used for atomic operations.

Field Grouping: The grouping is done based on the field values of the tuples. Field grouping allows you sending tuples to a particular bolt, based on the field value present in the tuples. Thus it is made sure that tuples of same field value always reach a particular bolt.

All Grouping: In this type of grouping the all bolts are allowed to accept a single copy of tuple. This kind of grouping is done when all bolts acquire a position to accept common tuple.

Refreshing bolts forms the best example where all the bolts acquire refreshing cache signal.

Custom Grouping: As the name indicates one can customize which tuple should be accepted by which bolt. Hence, it provides flexibility to users in handling tuples.

Direct Grouping: The grouping decision is made by the source. Grouping is based on the first letter of the tuple where various bolts are assigned with different letters and based on it the tuples reach that particular bolt.

Global Grouping: All tuples are grouped such that tuples from various sources reach a single bolt. This integrates and allows bolt to accept all tuples.

Processing Data: Tuples reach bolts for processing, bolts process each and every tuple based on the design structure of bolts. It allows guarantee of message delivery by acknowledging each tuple, this means that every single tuple is processed without any loss of tuples. Even if there is any loss, negative acknowledgement makes the bolt to process that particular tuple again. Fig 1.2 clearly defines the working of storm based on the design of the bolts there may be one or many tuples produced out from bolts. Anchoring helps in manipulation of tuples by providing stream ids and memory which serves as a buffer during processing. Hence, all the bolts gets executed and the desired output is provided to the master node.

Storm vs Apache Mapreduce: MapReduce provides an efficient framework in the grouping and processing of data in distributed clusters. MapReduce's parallel processing system makes client to acquire their results quickly and accurately [8]. Mapping and reducing are the two important works which is undergone. Both mapping and reducing functions take input and output as key value pairs. Various security related issues arise when data is transferred and manipulated among distributed clusters. In storm each tuple (data) is forwarded to bolts and the mapping or reducing operations are processed in bolts according to the topology design.

Based on Components: The Hadoop's MapReduce structure basically comprises of the name node and the data node. Name node holds the metadata information and serves as the master of the file system where the actual process of data takes place by keeping the index of all data nodes. Data node holds the data and processing of data takes place at the worker node. But in storm spout takes the complete responsibility of the data. The JobTracker in MapReduce works with name node manages the allocation and distribution of work across the clusters. TaskTracker works with the data node. Mapping or reduce function is processed in the worker node and the result is updated to the jobtracker conveying the status of the job[9]. The message ids which are provided to each data in storm help in tracking. In storm, every tuple holds the tuple id which is very much helpful in processing particular tuple and tracking it.

Though MapReduce seems working good with big data processing on exceeding with number of node more than 4,000 cascading and multi-tenancy forms the biggest issue to be resolved.

Storm performs exceptionally well when number of nodes is more than 40000 and tasks can be processed concurrently. The advanced version of MapReduce which is called as MapReduce 2.0 is much more reliable than the previous MapReduce structure and supports better scalability.

Storm vs Apache Yarn: YARN (Yet Another Resource Negotiator) architecture works on a hierarchical approach which provides better scalability and reliability among the Hadoop clusters [10]. The YARN architecture was proposed by altering Mapping & Reducing function with a new set of daemons.

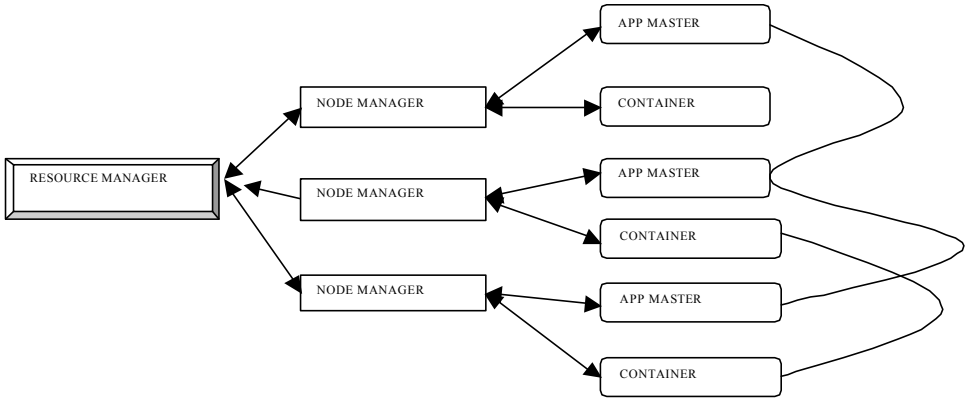

Fig. 2: Working of YARN

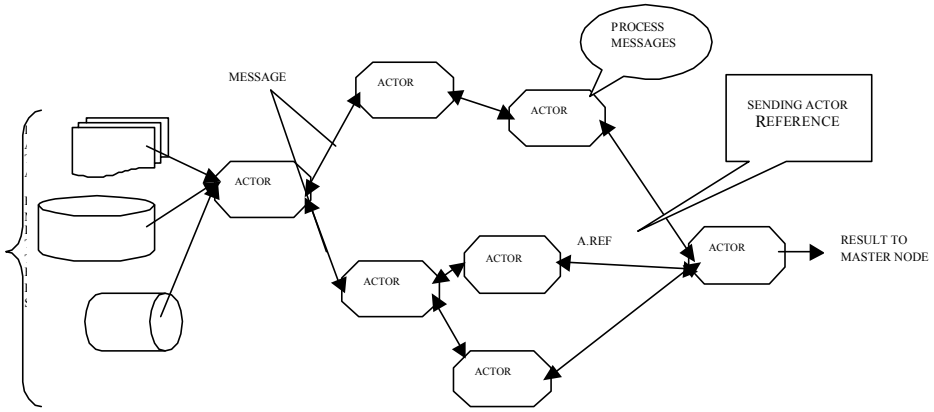

Fig. 3: Working of Akka

Based on Components: Resource Manager manages the tasks to computer resources which are distributed across the clusters and plays a huge role in allocation of resources like memory, compute, bandwidth etc., to underlying nodes in the cluster.

Fig 2 explains clearly about the components and working of YARN. In storm, the topology design decides the amount of resource needed. Node Manager, manages and supervises the container which runs over the distributed cluster nodes, the nodes which are within the YARN cluster are also managed by node manager. Application master manages various applications which are running on the distributed cluster and has direct contact with the resource manager for negotiating resources for clusters.

Storm vs Akka: Akka is the open source apache toolkit which is used to build distributed and concurrent applications [11]. This technology is very much similar to storm. Akka provides an effective abstract level design in handling fault tolerant, concurrent and distributive applications. Huge number of scalable and real time transaction processing system can be deployed easily with minimal effort by using akka since it provides an effective strategy in all abstraction levels. The concurrency, fault tolerant, Location-transparency and persistence properties of akka deserves efficient deployment of distributive cluster environments. The basic unit of data in akka is message, which are transferred and processed among the distributed clusters. Actor serves as the main units which are used to process the acquired messages. Tuples are the main unit of data in storm and bolts serves as the main unit for processing tuples.

Based on Components: Actors are the real objects which capsulate state and behavior. It is much related to object oriented concept where task can be allotted to persons for execution and persons themselves have a group of persons to execute subtasks allotted. Hence, they communicate among them through messages and process to get the desired output. Actors are similar to bolts in storm. Every actor has a mailbox. All the messages from other actors are enqueue in mailbox and processed by the actors. The communication between the sender and he receiver is made through the mailbox.

The mailbox fallows First Come First Serve (FCFS) scheduling policy in default for scheduling messages to actors for processing. The grouping schemes allow tuples to select bolts where a complete address of the bolt is mentioned. The basic unit of data in akka is messages and tuple in storm. Messages can be any object but the main thing is that all messages must be serialized in order to distribute it across remote actors. Actors has an ability to create more children's for execution of tasks that is, Actors split their tasks in to subtasks and assign those tasks to their children's for execution. Actors supervise those children's through many supervisor strategies and terminate them after completion of tasks allotted to them. Bolt may refer another bolt in case of huge execution of tasks.

Based on Working: The working of akka is much similar to storm in processing but there is a difference in handling the data.

Millions of actors can be created and processed which is the biggest advantage with akka. Various kinds of messages from various sources is fed in to the actor. The mailbox which is present in the actor receives all the messages from the source/message emitter and forwards each and every message to actor in first come first serve basis. The actor process those messages, if the task assigned to that particular actor is huge or that single actor cannot manage those task then actor creates children and divide those tasks to them for execution, This is very similar to storm which receives tuples from spout and executes it in bolts. The actors supervise and monitor in an asynchronous way so that the child cannot block their actors/supervisor. After processing the messages are forwarded to another actor for processing. The actors are identified in that distributed cluster through actors reference, actor path and actor address. The stream grouping allows tuples to select bolts in storm. Then the processed message from an actor reaches the other actor.

The same process is undergone again in that actor which has mailbox. The most advantage in actor is that messages can be sent back. Actors can send not only messages but can also send the actor references to another actor to choose a particular actor in that big distributed cluster. Hence the communication between the actors and the message delivery is guaranteed.

Thus the messages travel across various actors depending on the topological model and gets executed. All these results are fed to the master node. Storm rollbacks and process again if there is any loss of data or loss of acknowledgement after processing hence reliability and delivery is guaranteed. Data processing in cloud has become the cutting edge technology in the last few years [12-14].

Superviser Selection Algorithm: Storm works with stream processing, where there is continuous flow of tuples in to the topology, so an effective selection of supervisor and worker node is necessary to process the incoming streams. Zookeeper takes the full responsibility in selection of supervisors. Concentrating on selection of supervisor we propose a supervisor selection algorithm for Storm, which selects the supervisor based upon the resource it holds. The algorithm could increase the efficiency of the data processing in storm. The design of the algorithm is as follows:

Ssa Algorithm: Selection of Supervisor

- Initialize J_{max}
- *For i* = 1 *Ton Do*
- *Compute* J_i
- *If* (J_i< J_{max})
- *Set* $J_{max} = J_i$
- End If
- End For
- Choose a node which is having maximum free space as a supervisor S
- For M = 1 To K-1 Do
- Compute free workspace in each worker node i
- Choose the node with the smallest workspace and assign i to S
- End For

The upper bound on the optimal value of the number of tasks that storm can execute is $J_{opt.}$. The 'i' denotes the Node and 'N' denotes the total number of nodes present in the cluster. 'K' is the number of worker nodes present in the storm topology. J_i is computed by each and every node present in the cluster which provides the number of tasks which is executing in the cluster. This is then initialized as J_{max}. The node which is having the maximum resources for task execution is selected as the supervisor S. Then the computation of workers takes place in order to find the free space. This allows us to compute the amount of free space available in the worker node such worker nodes are selected and these nodes are assigned to the supervisor. The implementation of the SSA algorithm will be taken up as a separate project in the future.

CONCLUSION

This paper aims in understanding various components and working of Storm which is a real time data processing system. Storm is compared to other data processing systems which include MapReduce, YARN and an application data processing toolkit called Akka. An algorithm called SSA was proposed for selecting supervisor in the storm. Further developments and improvements in algorithm design may help in selecting efficient supervisor based on task, which can be implemented as a future work.

REFERENCES

1. Albert, Réka, Hawoong Jeong and Albert-László Barabási, 1999. Internet: Diameter of the world-wide web. Nature, 401(6749): 130-131.
2. Kaplan andreas, M. and Michael Haenlein, 2010. Users of the world, unite! The challenges and opportunities of Social Media. Business horizons, 53(1): 59-68.
3. Berkman Lisa, F. and S. Leonard Syme, 1979. Social networks, host resistance and mortality: a nine-year follow-up study of Alameda County residents. American Journal of Epidemiology, 109(2): 186-204.
4. Berners-Lee, Tim, James Hendler and Ora Lassila, 2001. The semantic web. Scientific american, 284(5): 28-37.
5. Cherniack Mitch, Hari Balakrishnan, Magdalena Balazinska, Donald Carney, Ugur Cetintemel, Ying Xing and Stanley B. Zdonik, 2003. Scalable Distributed Stream Processing. In CIDR, 3: 257-268.
6. Ikura Yoshiro and Mark Gimple, 1986. Efficient scheduling algorithms for a single batch processing machine. Operations Research Letters 5(2): 61-65.
7. http://storm.incubator.apache.org/documentation.html
8. White Tom, Hadoop, 2009. The Definitive Guide: The Definitive Guide. O'Reilly Media.
9. Dean Jeffrey and Sanjay Ghemawat, 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1): 107-113.
10. Vavilapalli Vinod Kumar, Arun C. Murthy, Chris Douglas, Sharad Agarwal, Mahadev Konar, Robert Evans, Thomas Graves *et al*., 2013. Apache hadoop yarn: Yet another resource negotiator. In Proceedings of the 4th annual Symposium on Cloud Computing, pp: 5. ACM, 2013.
11. http://akka.io/
12. Babu, L.D. and P. Venkata Krishna, 2013. Honey bee behavior inspired load balancing of tasks in cloud computing environments. Applied Soft Computing.
13. Babu, L.D. Dhinesh, *et al*., 2011. An analysis of security related issues in cloud computing. Contemporary Computing. Springer Berlin Heidelberg, pp: 180-190.
14. Ebin Deni Raj, L.D. Dhinesh Babu, Ezendu Ariwa, M. Nirmala and P.V. Krishna, 2014. Forecasting the Trends in Cloud Computing and its Impact on Future IT Business. IGI Global Publishers, 2014 In Green Technology Applications for Enterprise and Academic Innovation on pages, pp: 14-32.