ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Considerations for Big Data: Performance of Hadoop in Modern Era

Ashwini A. Somnathe, Abhijit T. Somnathe

Asst. Professor, Dept. of IT, KBP College, University of Mumbai, Vashi, Mumbai, India

Asst. Professor, Dept. of Electronics Engineering, SLRTCE, University of Mumbai, Mira Road, Mumbai, India

ABSTRACT: With the rapid growth of the Internet speed, the amount of data in our in world is exploding. The amount of Internet traffic is continuously increasing and data is being collected and stored at unprecedented rates. The challenge is not only to store and manage the vast volume of data, but also to analyse and extract meaningful value from it. There are several approaches to collecting, storing, processing, and analysing big data. In order to provide stable Internet service, efficient network management based on accurate traffic identification is gaining much importance than ever. MapReduce framework allows users to quickly develop big-data applications and process big-data effectively. Hadoop is a flexible and open source implementation for analysing large datasets using Map Reduce. Hadoop is widely adopted to support data intensive distributed applications. This paper analyses the bottlenecks in the architecture of Hadoop that causes the time delay in the tasks and lead to the prolonged execution time of the jobs. Unfortunately, Hadoop has no high availability support yet, and it is not trivial to enhance Hadoop. We see Hadoop features, extensions, and tools as well as significant opportunities for optimization. We have also found some standard techniques that help in improving the potential and interpretation effectively but also suggested some of their substitutes. Hadoop clusters are an effective means of processing massive volumes of data, and can be improved with the right architectural approach. Altogether, some outstanding ways of opportunity for the simplification of the usage and the optimization of the Hadoop technology are discussed that also helps in making future recommendations for further research work in this area.

KEYWORDS - Hadoop, Metadata, network management, Map Reduce, Nodes, Security.

I. INTRODUCTION

In the daily lives, the data is generated at such unprecedented rates that the data intensive computations are performed in every second and hence so truly called “The digital universe”[4]. Big Data is becoming a hot topic in many areas where datasets are so large that they can no longer be handled effectively or even completely. So any task which is comparatively easy to execute when operating on a small but relevant set of data, but becomes unmanageable when dealing with the same problem with a large dataset can be classified as a Big Data problem [3]. The mindsets of researchers are changed due to the phenomenal raised usage of the internet to store and analyse voluminous data. Typical problems encountered when dealing with Big Data include capture, storage, dissemination, search, analytics and visualization. But there are also new domains encroaching on the Big Data paradigm: data warehousing, Internet and social web search, finance and business informatics. Here datasets can be small compared to the previous domains; however the complexity of the data can still lead to the classification as a Big Data problem. Nowadays , companies need to acquire some reliable solutions that can process multi Petabyte set of data efficiently and hence various groups of engineers in the companies like Nokia and some other telecommunications had already began experimenting with Hadoop. Sectors that employ these techniques include retailers, banking institutions and some insurance companies as well as other health related fields. Hadoop also provides some methods to enhance the availability of applications by making the data replications but it lacks for the high availability for itself which is called SPOF. It means single point of failure or failure due to failure of single node i.e. mainly the critical node.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

II. BACKGROUND AND RELATED WORK

Since the weakness of the centralized namespace storage of Hadoop has surfaced up, there have been attempts in research publications providing strategies for eliminating the single point of failure and address the scalability issue of the architecture. In this section, the background of this research field and related studies are reviewed.
Dhruba Borthapur discussed the issue of single point of failure of Hadoop and suggests improvements in failover of Name node server. The Avtar Node [11] was developed to address the issue of failover and a mechanism to address the single point failure of the Name node. The primary Avtar Node is a Name node and writes its transaction logs into the shared NFS filer. Another instance of Avtar Node is running and called standby node which continually reading the transaction logs from the same shared NFS filer. The standby name node do not participate in the functioning of HDFS. The Avatar Node is effective mechanism to guard against Name node failures and keeps the namespace data protected. However, the Avatar Node does not address the high scalability of the architecture and still has the single point of access to the cluster. As the namespace grows, the two name node servers do not load balance the work. This approach provides failover and not able to accommodate the large namespace.
Feng Wang's discussed the metadata replication based solution to provide high availability and failover technique [2]. The solution has phases: the first is the initialization phase which initializes the execution environment of high availability. The second phase replicates metadata from critical node to corresponding backup node at runtime. The last phase resumes the running of Hadoop. As the file system information and Edit Log transactions are stored as a backup copy on the Name node, the solution emphasizes on the replication of critical metadata. It presents an adaptive method for failure recovery of the Name node by metadata replication with further reduces failover duration, but it does not solve the issue of single point of failure with Hadoop.
Apache group came up with a solution to called federation that means the name nodes are independent and don't require coordination with each other. In order to scale the name service horizontally, federation uses multiple independent name nodes servers. The name nodes are federated [10] that means the name nodes are independent and don't require coordination with each other. This approach is suitable for running many independent namespace in one cluster. All these namespaces are still not contributing to the scalability of single namespace and big cluster deployment still depends on single name node server. Though the namespace data is store on independent data nodes but still namespace cache is not distributed and has scalability limitations. This idea is good to have multiple namespace in a single cluster. A federated cluster is designed to store more data and handle more clients, because it has multiple name nodes. However, each individual name node is subject to the same limits and shortcomings, such as lack of High Availability (HA), as a non-federated one. The federated approach provides a static partitioning of the federated namespace. If one volume grows faster than the other and the corresponding Name node reaches the limit, its resources cannot be dynamically repartitioned among other name nodes except by manually copying files between file systems.

III. HADOOP SYSTEM

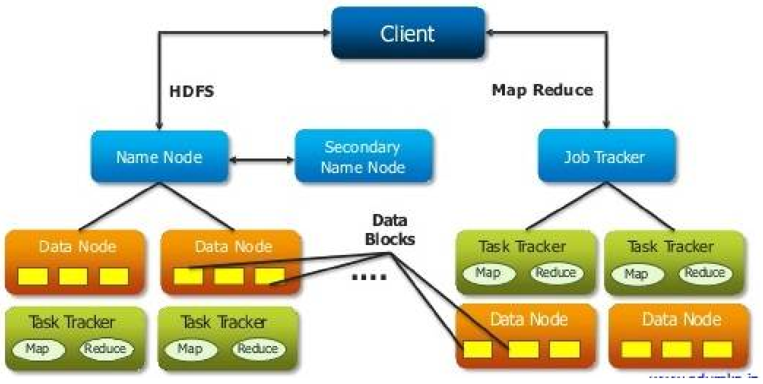

Fig 1: Hadoop Architecture

A. NAME NODE

The Name Node is the centrepiece of an HDFS file system. It keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. It does not store the data of these files itself. Client applications talk to the Name Node whenever they wish to locate a file, or when they want to add/copy/move/delete a file. The Name Node responds the successful requests by returning a list of relevant Data Node servers where the data lives. The Name Node is a Single Point of Failure for the HDFS Cluster. HDFS is not currently a High Availability system. When the Name Node goes down, the file system goes offline. There is an optional Secondary Name Node that can be hosted on a separate machine. It only creates checkpoints of the namespace by merging the edits file and does not provide any real redundancy.

B. DATA NODE

A Data Node stores data in the Hadoop File System. A functional file system has more than one Data Node, with data replicated across them. On start-up, a Data Node connects to the Name Node; spinning until that service comes up. It then responds to requests from the Name Node for file system operations. Client applications can talk directly to a Data Node, once the Name Node has provided the location of the data. Similarly, Map Reduce operations farmed out to Task Tracker instances near a Data Node, talk directly to the Data Node to access the files. Task Tracker instances can indeed should be deployed on the same servers that host Data Node instances, so that Map_Reduce operations are performed close to the data. Data Node instances can talk to each other, which is what they do when they are replicating data.

IV. PROPOSED METHODOLOGY

MapReduce is a framework for processing parallelizable problems across huge datasets using a large number of computers (nodes), collectively referred to as a cluster (if all nodes are on the same local network and use similar hardware) or a grid (if the nodes are shared across geographically and administratively distributed systems, and use more heterogeneous hardware). Processing can occur on data stored either in a file system (unstructured) or in a database (structured). MapReduce can take advantage of the locality of data, processing it near the place it is stored in order to reduce the distance over which it must be transmitted. MapReduce is proposed by Google. It is a programming model and implemented for large scale data processing in distributed cluster HDFS.

- **"Map" step:** Each worker node applies the "map ()" function to the local data, and writes the output to a temporary storage. A master node ensures that only one copy of redundant input data is processed.
- **"Shuffle" step:** Worker nodes redistribute data based on the output keys (produced by the "map ()" function), such that all data belonging to one key is located on the same worker node.
- **"Reduce" step:** Worker nodes now process each group of output data, per key, in parallel.

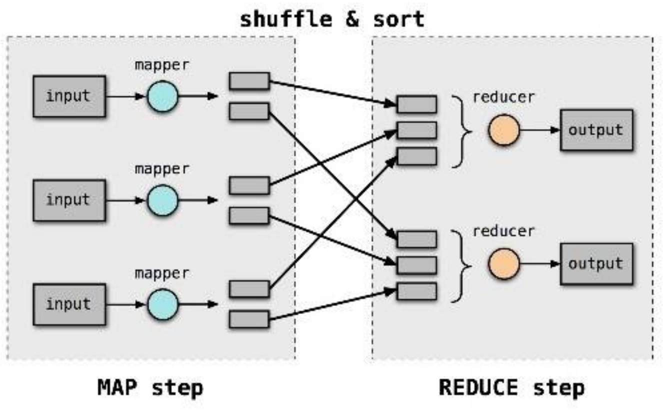

Fig 2: Map Reduce Framework

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

MapReduce allows for distributed processing of the map and reduction operations. Hadoop MapReduce framework is used for executing applications holding enormously immense amounts of data (terabytes of data) in parallel on largely built clusters with numerous no of nodes in a reliable and fault-tolerant way. Similarly, a set of 'reducers' can perform the reduction phase, provided that all outputs of the map operation that share the same key are presented to the same reducer at the same time, or that the reduction function is associative. Though it can run on a single machine, its real power reside in its potential to scale up to several thousands of systems each with several processors. Hadoop is constructed in a way that it distributes the data effectively and efficiently across multiple nodes in the cluster. MapReduce can be applied to significantly larger datasets than "commodity" servers can handle – a large server farm can use MapReduce to sort a petabyte of data in only a few hours. It involves the distributed file system that is responsible for the distribution of the massive amount of data sets efficiently across the nodes in the cluster. MapReduce framework splits the task into various numbers of chunks and the Map tasks process all them in parallel. The parallelism also offers some possibility of recovering from partial failure of servers or storage during the operation: if one mapper or reducer fails, the work can be rescheduled – assuming the input data is still available. The yield from the map tasks are sorted by the framework and provided as input to Reduce tasks. Both the input and output of the following tasks are registered in a file system. The scheduling of the tasks while monitoring the failed ones and re-executing them is done by the framework itself.

V. ROLE OF TRACKERS

a) Job Tracker

The Job Tracker is the service within Hadoop that farms out Map Reduce tasks to specific nodes in the cluster, ideally the nodes that have the data, or at least are in the same rack. Every cluster carry only one Job Tracker sometimes called a **"daemon service"** for submitting and tracking MapReduce jobs in Hadoop and hence also responsible for the occurrence of single point of failure so if it goes out of the service all the currently running jobs Will also be halted. The Job Tracker is a point of failure for the Hadoop Map Reduce service. If it goes down, all running jobs are halted.

b) Task Tracker

A Task Tracker is a node in the cluster that accepts tasks - Map, Reduce and Shuffle operations - from a Job Tracker. The slaves configured perform tasks as directed by the Job Tracker. Every Task Tracker is configured with a set of *slots*; these indicate the number of tasks that it can accept. Each slave node has only one Task Tracker which maintains the track of task instances and each time it give notification to the Job Tracker about the status of the level of task implemented. When the Job Tracker 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. After this the job report is submitted to the client. The report consists of the status and diagnostic statistics of the tasks.

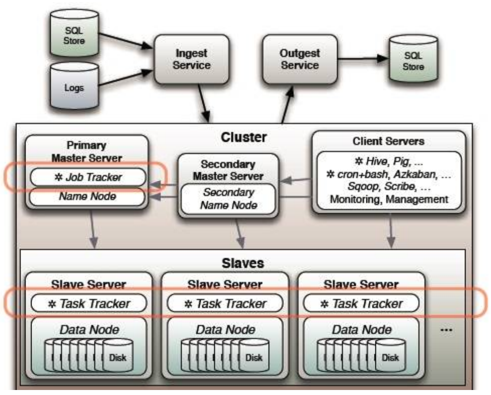

Fig 3: Role of job tracker and task tracker

VI. RESULTS

Hadoop clusters can be benchmarked in several ways and with several options. In addition to this, IBSG predicts that there will be 30 billion devices that will connect to the Internet by 2016and approximately 50 billion by 2020. Thus, the information stored in such solutions will be statistically representative making the data analytics performed on it reliable. Selecting benchmarks depends on the nature of the workload and the cluster infrastructure characteristics. When setting up a Hadoop cluster we always want to confirm if the cluster is precisely and accurately configured and this can be concluded by running the tasks linearly and then analysing the results. The following benchmarks are reviewed for Hadoop.

- **Test DFSIO**: helps in Testing the Distributed File System input and output performance evaluation of Hadoop Distributed File System.
- **Map Reduce sort** : This actually test the Map Reduce performance in the Hadoop by creating Map Reduce tasks to accomplish the initial partial sorting of input and then transfer the input by the shuffle. This test is performed in three steps: generation of the random data, sorting of the data, and validation of the result.
- **Grid mix suit:** It utilizes the mixture of the synthetic jobs, and assembles the realistic cluster workloads by resembling the all patterns obtained on real time systems in which the data is accessed.

Experimentally it is found that Hadoop cluster that runs in virtual scenario significantly perform low than the one that runs over the physical environment (physical machine). Other factors also affect the performance but in this study we focused on the environment effect which is 68% affecting the performance of the Hadoop cluster. A comparison chart is prepared between the executions of the Hadoop cluster in both the environments. Figure represents the difference in the performance between the clusters running in different environments.

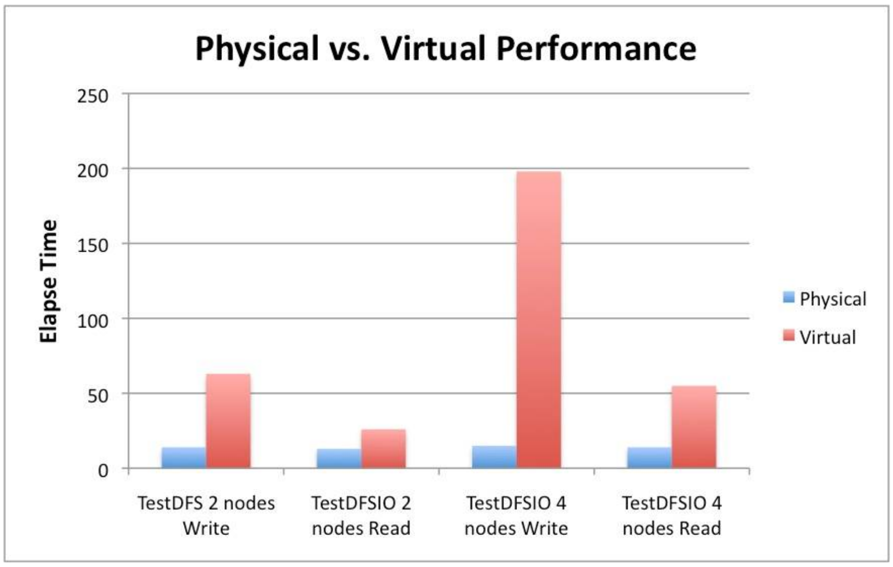

Fig 4. Physical Vs Virtual Performance of Hadoop

From the above observation it can be concluded that virtual Hadoop cluster performance is notably lower than the cluster running on physical machine because of the overhead of the virtualization on the CPU of the physical host. The various factors like (RAM size, network bandwidth), were considered in our experiment. However, our aim was directed towards the effect of environmental aspects on the performance.

Whenever a Hadoop System is modelled and checked for various factors affecting the whole processing it is observed that more than 50% the external factors affect the system mainly the environment. Errors are also evaluated and always sum of the modelled errors should be equal to zero, as they are the difference between measured response and estimated response.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

VII. PERFORMANCE CONSIDERATIONS

For Big Data, performance is an intrinsic part of storage and data retrieval within Hadoop. There are the principle guidelines that are used to examine the performance in contrast between the systems to distinguish between likely alternatives. For measuring the performance of the cluster, the preferred choice is to isolate the cluster and execute benchmarks so that we can evaluate the resources utilized and the cluster speed of the whole processing. This section describes several performance factors that come into play for a Hadoop cluster.

Cluster Sizing
The distributed architecture of Hadoop allows the cluster to be expanded as the data and processing requirements grow. As the nodes are added to the cluster, the corresponding network infrastructure and Name Node needs to be sized accordingly. Generally, the key metric of job completion time decreases as the number of nodes increases in the cluster.

MapReduce Algorithm
The algorithmic details, data model of the algorithm, and the size of the input dataset have to be taken into account. One such consideration can be placing too much of the computation into Mappers and not enough in the reducers or vice versa, making the algorithm inefficient. For example, workload that has very little processing in the mapping function potentially generates more data for reducers to process and therefore, more network traffic between mappers and reducers. It is important to divide the algorithm efficiently, keeping in mind the functional differences of the map and reduce phases to achieve optimal results. Also, it is possible for certain workloads to create a pipeline of smaller MapReduce jobs instead of a single large MapReduce job, with the entire pipeline completing faster than the single job.

Input Data Set
For the same algorithm, the larger the input dataset, the longer it takes to complete the processing and produce the results.

Data Node
This requires the proper sizing of CPU, memory, network, and storage elements of data nodes. A more detailed review of this topic is provided later in this paper in the Capacity Planning section. MapReduce jobs can vary in CPU and memory intensiveness. The capacity planning aspects (CPU architecture, number of cores, and processing power of the cores) and memory (amount of memory and memory latency) available for such jobs determine how quickly the Map or Reduce phase can be completed. Faster processing and more memory generally decrease the amount of time needed for job completion. Regarding storage, typical disk data transfer rates and the time required to read the HDFS blocks needed by the MapReduce algorithm can contribute to the total job time.

Data Locality
If the data is locally available on the data node, the task can complete faster. The Hadoop Job Tracker is intelligent enough to schedule jobs optimally on nodes where the data is available. However, there are instances where all the nodes that host a particular data block might be experiencing heavy load. In such data miss instances, the Job Tracker is forced to schedule tasks in any node. Because the data blocks needed by the task are now unavailable locally, the task must request the data block from a node that has the data. Each such data transfer consumes resources on the node requesting the data, the node providing the data, and the network; and can lead to slower job completion.

Concurrent Activity
Hadoop clusters typically run MapReduce jobs in parallel, in addition to background activities such as importing/exporting data from HDFS or internal HDFS operations such as data rebalancing when nodes are added or removed. Therefore, all the running jobs in a cluster determine the available resources for the newly submitted job. One option is to increase the priority of the new job, at the cost of other existing jobs. Other options include strategies such as speculative execution, where the Hadoop infrastructure can speed up job completion by intelligently spawning multiple instances of the same task. Lesser sharing of the cluster among Map Reduce jobs will typically lead to less contention and better performance for the scheduled jobs.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

Network Considerations
The interconnections of the cluster nodes are heavily used during reading and writing data to the HDFS. Also, the network interconnects are highly utilized during the shuffle step, and Map Reduce metadata communications among the nodes. Additionally, there are several network characteristics that need to be considered. For network availability and resiliency, it is important to deploy a network that provides the required redundancy and can also scale as the cluster grows. Technologies that allow network designs with multiple redundant paths between the data nodes are better suited for the cluster nodes. Burst handling and Queue Depth are also important characteristics to consider. Some steps in MapReduce, such as the read/write function in HDFS and the shuffle step can be bursty in nature for the network traffic. If the network cannot handle bursts effectively, it drops packets. Therefore, an optimal buffer is needed in network devices to absorb bursts. Any dropped packets due to buffer non-availability result in retransmission, leading to longer job completion times. Therefore, network architecture with proper buffer and queues must be considered. The network oversubscriptions ratio is also an important network consideration. A higher oversubscription ratio can cause dropped packets and lower performance. Conversely, lower oversubscription can be costly to implement. Finally, network latency is important; specifically, end-to-end latency should be considered. Ultralow latency is not typically a major factor, but consistently and predictably low latency will benefit the cluster and application developers. This includes OS, JVM, and network latency parameters.

VIII. CONCLUSION

In this paper it is tried to obtain issues that mainly comes while discussing the performance of HDFS on heterogeneous clusters. Both name node and job tracker are critical nodes in Hadoop. Performance degradation happens because of the heterogeneity. For this we presented a brief overview of privacy capabilities of common social media services regarding their capability of protecting users from other peoples' activities. Our conclusion is that the use of Hadoop for academic research is in the stage of adolescence. We also conducted an analysis of privacy related metadata, particularly location data contained in social media and analysed over 28k real world images from popular social media sites. Optimization techniques for HDFS discussed in this paper will definitely help in boosting up the efficiencies of the running algorithms in the background and hence motivate the use of this parallel computing paradigm. Based on this survey we analysed the Big Data privacy implications and potential of the emerging trend of geo-tagged social media. We then presented three concepts how this location information can actually help users to stay in control of the flood of potentially harmful or interesting social media uploaded by others. Hadoop is ultimately a "mission critical" at this platform and a productive environment where it is widely used and adopted.

REFERENCES

[1] S. Ahern, D. Eckles, N. Good, S. King, M. Naaman, and R. Nair. Overexposed?: privacy patterns and considerations in online and mobile photo sharing. In Proceedings of the SIGCHI conference on Human factors in computing systems, pages 357–366, 2007.
[2] Feng Wang, Bo Dong, Jie Qiu, Xinhui Li, Jie Yang, Ying Li "Hadoop High Availability through Metadata Replication" 2009 ACM 978-1-60558-802-5/09 Pg 37-44.
[3] C. a. Ardagna, M. Cremonini, S. De Capitani di Vimercati, and P. Samarati. An Obfuscation-Based Approach for Protecting Location Privacy. IEEE Transactions on Dependable and Secure Computing, 8(1):13–27, June 2009.
[4] K. Bakshi, "Considerations for Big Data: Architecture and Approach", Aerospace Conference IEEE, Big Sky Montana, March 2012.
[5] A. Beresford and F. Stajano. Location privacy in pervasive computing. IEEE Pervasive Computing, 2(1):46–55, Jan. 2003.
[6] M. Smith, C. Szongott, B. Henne and G. Voigt , "Big Data Privacy Issues in Public Social Media", Digital Ecosystems Technologies (DEST), 6th IEEE International Conference on, Campione d'Italia, June 2012.
[7] A. Besmer and H. Richter Lipford. Moving beyond untagging. In Proceedings of the 28th international conference on Human factors in computing systems - CHI '10, page 1563, Apr. 2010.
[8] E. Baldeschwieler and D. Cutting. 2009. State of Hadoop. In Hadoop Summit 2009 (Santa Clara, US, June 10, 2009).
[9] D. Boyd. Facebook's privacy trainwreck: Exposure, invasion, and social convergence. Convergence: The International Journal of Research into New Media Technologies, 14(1):13–20, 2008.
[10] Apache Zookeeper. http://hadoop.apache.org/zookeeper/
[11] Borthapur, D., 2010, Hadoop Avatar Node High Availability, http://hadoopblog.blogspot.com/2010/02/hadoop-namenode high-availability.html, Facebook.

ISSN(Online) : 2320-9801
ISSN (Print) : 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 4, Issue 4, April 2016

BIOGRAPHY

Ashwini A. Somnathe is an Assistant Professor in the Information Technology Department, KBP College of Arts, Science and Commerce, Mumbai University. She received M.Sc. (Computer Science) in 2006 from RTMNU, Nagpur, MS, India. Her research interests are Image Processing, Embedded Systems, Operating Systems, etc.

Abhijit T. Somnathe is an Assistant Professor in Electronics Engineering Department, Shree L. R. Tiwari College of Engineering, Mumbai University. He received M.E. (Electronics Engineering) in 2014 from University of Mumbai, MS, India. His research interests are Image Processing, Computer Networking, Embedded Systems, etc.