ISSN 2347 - 3983

Volume 8. No. 3, March 2020

International Journal of Emerging Trends in Engineering Research

Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter23832020.pdf

https://doi.org/10.30534/ijeter/2020/23832020

A Bigdata processing with Hadoop Map Reduce in Cloud Systems

K.Jose Triny[1], G.Anjuka[2], C.Dhanapal[3], S.Kavibharani[4],C.Kowsalya[5]

[1]Department of CSE,M.Kumarasamy College Of Engineering, Karur, India, kbjtriny@gmail.com
[2]Department of CSE,M.Kumarasamy College Of Engineering, Karur, India, anjukarenya23399@gmail.com
[3]Department of CSE,M.Kumarasamy College Of Engineering, Karur, India, dhanapalmhr@gmail.com
[4]Department of CSE,M.Kumarasamy College Of Engineering, Karur, India, kavirajisathish@gmail.com
[5]Department of CSE,M.Kumarasamy College Of Engineering, Karur, India, kowsalyac1404@gmail.com

ABSTRACT

The new ages of cell phones have high preparing force and capacity, yet they linger behind as far as programming frameworks for large information stockpiling and preparing. Hadoop is a versatile stage that gives disseminated capacity and computational capacities on groups of product equipment. Building Hadoop on a versatile system empowers the gadgets to run information escalated processing applications without direct information on basic conveyed frameworks complexities. In any case, these applications have serious vitality and unwavering quality requirements (e.g., brought about by startling gadget disappointments or topology changes in a powerful system). As cell phones are progressively helpless to unapproved get to, when contrasted with customary servers, security is additionally a worry for delicate information. Consequently, it is vital to think about unwavering quality, vitality effectiveness and security for such applications. The MDFS (Mobile Distributed File System) addresses these problems for huge information preparing in portable mists. We have built up the Hadoop Map Reduce structure over MDFS and have examined its exhibition by differing input outstanding burdens in a genuine heterogeneous portable bunch. Our assessment shows that the execution tends to all imperatives in handling enormous measures of information in versatile mists. In this manner, our framework is a suitable answer for satisfy the developing needs of information handling in a portable condition.

Key words: Hadoop Map Reduce, Mobile Distributed file system

1.INTRODUCTION

With progresses in innovation, cell phones are gradually supplanting conventional PCs. The new periods of phones are ground-breaking with gigabytes of memory and multi-focus processors. These gadgets have very good quality figuring equipment furthermore, complex programming applications that produce huge measures of information on the request for many megabytes. This information can extend from application crude information to images, sound, video or content documents. With the quick increment in the quantity of cell phones, large information preparing on cell phones has become a key developing need for giving capacities like those gave by customary servers [1].

Current versatile applications that perform gigantic figuring errands (huge information preparing) offload information and errands to server farms or ground-breaking servers in the cloud [2].The casing work parts the client work into littler undertakings and runs these undertakings in equal on various hubs, subsequently decreasing the in general execution time when contrasted and a successive execution on a solitary hub. This engineering however, fails without outer system network, as it is the situation in military or fiasco reaction operations. This engineering is additionally stayed away from in crisis reaction situations where there is constrained network to cloud, prompting costly information transfer and download tasks. In such circumstances, remote portable specially appointed systems are regularly sent [3]. The restrictions of the conventional distributed computing rouse us to examine the information preparing issue in an infrastructure less and versatile condition in which the web is inaccessible and all occupations are performed on cell phones. We expect that cell phones in the region are willing to share each other's computational assets.

There are numerous difficulties in bringing large information capabilities to the versatile condition: 1) cell phones are asset obliged as far as memory, processing force and vitality. Since most cell phones are battery fueled, vitality utilization during work execution must be limited. By and large vitality utilization relies upon the hubs chose for the activity execution. The hubs must be chosen dependent on every hub's remaining vitality, work recovery time, and vitality profile. As the employments are recovered remotely, shorter occupation recovery time demonstrates lower transmission vitality and shorter work fruition time. Contrasted with the conventional cloud processing, transmission time is the bottleneck for the work makespan and remote transmission is the major wellspring of the vitality utilization; 2) unwavering quality of information is a significant test in powerful systems with unpredictable topology changes. Association disappointments could cause cell phones to leave the system reach after restricted investment. Gadget disappointments may likewise occur because of vitality exhaustion or application explicit disappointments. Consequently, an unwavering quality model more grounded than those utilized by conventional static systems is basic; 3) security likewise a significant worry as the put away information frequently contains delicate client data [4] [5]. Conventional security instruments customized for static systems are lacking for dynamic systems. Gadgets can be caught by unapproved clients and information can be undermined effectively on the off chance that essential safety efforts are not given. To address the previously mentioned issues of vitality productivity, rely capacity and security of dynamic system topologies, the k out of n registering system was presented [8] [9].

Here, in this paper, we execute Hadoop Map Reduce system over MDFS and assess its exhibition on a general heterogeneousbunch of gadgets. We implement the nonexclusive document framework interface of for MDFS which makes our framework interoperable with other Hadoop structures like HBase. There are no progressions required for existing HDFS applications to be sent over MDFS. As far as we could possibly know, this is the primary work to bring Hadoop Map Reduce structure for portable cloud that genuinely addresses the difficulties of the dynamic system condition. Our framework gives an appropriated figuring model to processing of huge datasets in versatile condition while guaranteeing solid certifications for vitality productivity, information unwavering quality and security.

2.LITERATURE SURVEY

There have been a few researches contemplates that attempted to bring Map Reduce structure to the heterogeneous bunch of gadgets, because of its effortlessness and ground-breaking deliberations [11].

Marinelli presented the Hadoop based stage Hyrax for distributed computing on cell phones. Hadoop Task Tracker and Data Node forms were ported on Android cell phones, while a solitary case of Name Node and Job Tracker were run in a customary server. Porting Hadoop forms straightforwardly onto versatile gadgets doesn't moderate the issues looked in the mobile condition[12]. As exhibited before, HDFS isn't well appropriate for dynamic system situations. There is a need for a progressively lightweight document framework which can satisfactorily address dynamic system topology concerns. Another Map Reduce system dependent on Python, Misco was executed on Nokia cell phones[13]. It has a comparative server-customer model where the server follows along of different client occupations and doles out them to laborers on request. One more server customer model based Map Reduce framework was proposed over a bunch of portable gadgets [14] where the versatile customer actualizes Map Reduce rationale to recover work and produce results from the ace hub. The above arrangements, be that as it may, don't comprehend the issues associated with information stockpiling and handling of enormous datasets in the dynamic system[6].

P2P Map Reduce depicts a model implementation of a Map Reduce system which utilizes a distributed model for equal information handling in dynamic cloud topologies. It depicts components for overseeing hub and occupation disappointments in a decentralized way[15].

The past research concentrated uniquely on the equal handling of undertakings on cell phones utilizing the Map Reduce structure without tending to the genuine difficulties that happen when these gadgets are sent in the portable condition. Huchton et al. [1] proposed a k-Versatile Mobile Distributed File System (MDFS) for cell phones focused on essentially for military operations. Chen et al. proposed another asset allocation plot dependent on k-out-of-n structure and executed an increasingly solid and vitality effective Mobile Dispersed File System for Mobile Ad Hoc Networks (MANETs) with critical upgrades in vitality utilization over the customary MDFS engineering. [16].

3.PROPOSED SYSTEM

3.1 Hadoop Architecture

The two essential segments of Apache Hadoop are the Map Reduce system and HDFS, as appeared in Figure 1. Map Reduce is a versatile equal preparing system that sudden spikes in demand for HDFS. It alludes to two particular assignments that Hadoop occupations play out the Map Task and the Reduce Task. The Map Task takes the information set and creates a lot of middle of the road

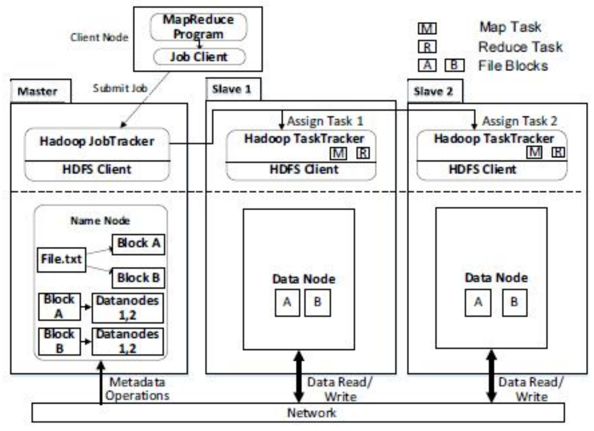

Figure 1: Hadoop architecture

<key, value>sets which are arranged and parceled per reducer. The map yield is then passed to Reducers to deliver the last yield. The client applications execute mapper furthermore, reducer interfaces to give the guide and lessen capacities[7]. In the Map Reduce system, calculation is constantly drawn nearer to hubs where information is found, rather than moving information to the figure hub. In the perfect case, the figure hub is likewise the capacity hub limiting the system blockage and along these lines maximizing the general throughput[10].

2 significant modules in Map Reduce are the Job Tracker and the Task Tracker. Job Tracker is the Map Reduce ace daemon that acknowledges the client employments also, parts them into numerous errands[17]. It at that point doles out these errands to Map Reduce slave hubs in the group called the Task Trackers. Task Trackers are the processing hubs in the group that run the errands Map and Diminish[26]. The Job Tracker is liable for planning errands on the Task Trackers and re-executing the fizzled errands. Task Trackers report to Job Tracker at

customary interims through heartbeat messages which convey the data with respect to the status of running errands and the quantity of accessible spaces. HDFS is a solid, shortcoming tolerant appropriated record framework intended to store huge datasets. Its key highlights incorporate burden adjusting for greatest effectiveness, configurable square replication systems for information protection, recuperation systems for adaptation to internal failure and auto versatility. In HDFS, each record is part into squares and each square is recreated to a few gadgets over the bunch[27].

3.2 MDFS Overview

As appeared in Figure 2, each document is scrambled utilizing a mystery key and divided into n1 record pieces utilizing deletion encoding (Reed Solomon calculation). Unlike customary plans, the mystery key isn't put away locally. The key is part into n2 sections utilizing Shamir's mystery key sharing calculation. Document creation is finished at the point when all the key and record sections are disseminated over the bunch. For record recovery, a hub needs to recover in any event k1 (<n1) document sections and k2 (<n2) key pieces to remake the first document. MDFS system gives high security by ensuring that information can't be decoded except if an authorized client acquires k2 unmistakable key pieces[18]. It likewise guarantees versatility by permitting the approved clients to reproduce the information considerably in the wake of losing n1-k1 fragments of information. Unwavering quality of the document increments when the proportion k1/n1 diminishes, however it additionally brings about higher information repetition[19]. The information sections are set on a set of chosen stockpiling hubs thinking about every hub's disappointment likelihood and its separation to the potential[20]
customers. A hub's disappointment likelihood is assessed based on the rest of the vitality, arrange availability, and application-subordinate elements. Information sections are at that point dispensed to the system with the end goal that the normal information moving vitality for all customers to recover/recoup the document is limited.

MDFS has a completely circulated registry administration in which every gadget keeps up data with respect to the rundown of accessible documents and their comparing key and document sections. Every hub in the system occasionally synchronizes the catalog with different hubs guaranteeing that the catalogs of all gadgets are constantly refreshed[21].

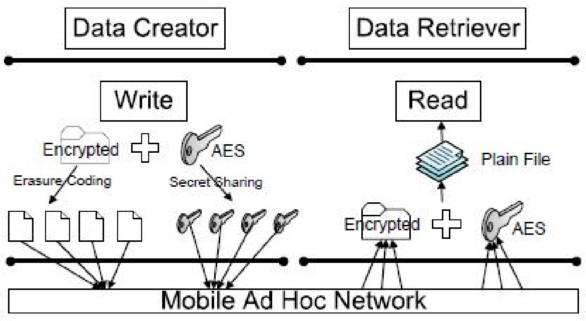

Figure 2: Overview of MDFS Architecture

3.3 System Architechture

We propose two methodologies for our MDFS design
A Distributed design where there is no focal element to deal with the bunch and a Master-slave architecture, as in HDFS. In spite of the fact that the tradeoffs between the circulated engineering and the incorporated architecture in a circulated framework is well-considered; this paper is the first to actualize and analyze Hadoop system on these two structures.

a) Distributed Architecture

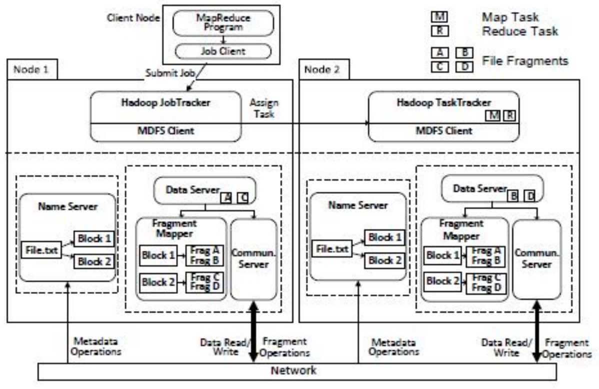

Figure 3: Distributed Architecture of MDFS

Here in this system, each participating hub runs a Name Server and a Fragment Mapper. After each record framework activity, the update is communicated in the system with the goal that the neighborhood reserves of all hubs are synchronized. Also, every hub periodically synchronizes with different hubs by sending communicate messages. Any new hub entering the system gets these communicate messages and makes a neighborhood reserve for further tasks. In the equipment execution, the updates are communicated utilizing UDP parcels. We expect every utilitarian hub can effectively get the communicate messages. The more complete and vigorous circulated catalog administration is left as future work[22].

This system has no single purpose of disappointment and no imperative is forced on the system topology. Each hub can work autonomously, as every hub stores a different duplicate of the namespace and section mapping. The heap is uniformly appropriated over the group as far as metadata stockpiling when contrasted with the brought together design. Be that as it may, organize data transfer capacity is squandered because of the messages communicate by every hub for refreshing the nearby store of each other hub in the organize. As the quantity of hubs associated with handling expands, this issue turns out to be increasingly extreme, prompting higher reaction time for every client activity[23].

b) Master Slave Architecture

In this design portrayed, the Name Server and the Fragment Mapper are singleton in positions over the total group. These daemons can be run in any of the hubs in the group. The hub that runs these daemons is known as the ace hub. MDFS stores metadata on the ace hub like other disseminated frameworks like HDFS, GFS and PVFS[24].

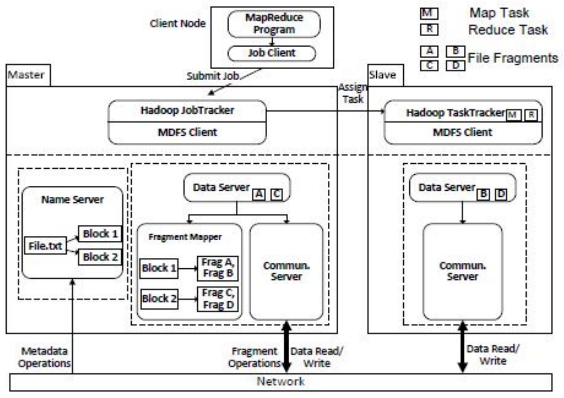

Figure 4: Master Slave Architecture

c)System Operation

i) File read operation

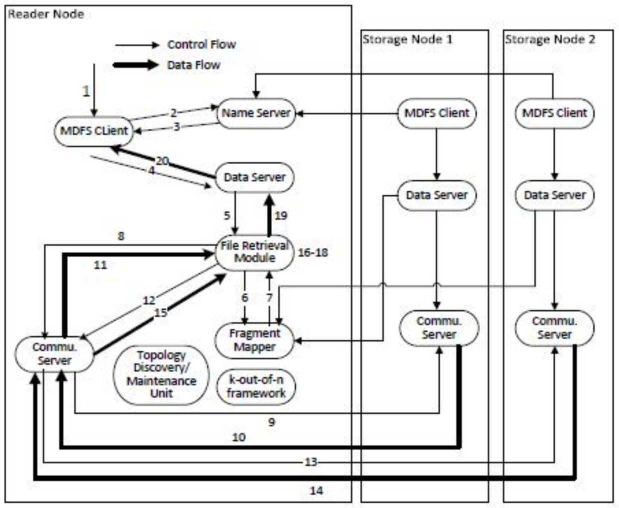

Figure 5: Dataflow of read operation

Figure outlines the control stream of a read activity through these numbered advances.

Stage 1: The client gives a read solicitation for record squares of length L at a byte balance O.

Stages 2,3: As in HDFS, the MDFS customer inquiries the Name Server to restore all squares of the document that range the byte balance run from O to O+L. The Name Server looks the nearby store for the mapping from the record to the rundown of squares. It restores the rundown of hinders that contain the mentioned bytes.

Stage 4: For each square in the rundown returned by the Name Server, the customer gives a recovery solicitation to the Information Server. Each document framework activity is distinguished by a particular opcode in the solicitation.

Stage 5: The Data Server recognizes the opcode and starts up the File Retriever module to deal with the square recovery.

Stages 6-7: The Data Server demands the Fragment Mapper to give data in regards to the key and record pieces of the document. The Fragment Mapper answers with the character of the pieces and the areas of the parts in the systems.

Stages8to15: The DataServer demands the Communication Server to get the necessary number of pieces from the areas which are recently returned by the Section Mapper. Sections are brought in equal also, put away in the nearby document arrangement of the mentioning customer. In the wake of bringing each solicitation, the Communication Server recognizes the Data Server with the area where the sections are put away in the nearby document framework.

Stage 16: The above activities are rehashed for bringing the key sections. These subtleties are not included in the chart for quickness. The mystery key is built from the key sections.

Stage 17: Once the necessary grind parts are downstacked into the nearby document framework, they are decoded and at that point unscrambled utilizing the mystery key to get the first square.

Stage 18: The key and document parts which were downloaded into the nearby document framework during the retrieval process are erased for security reasons.

Stage 19: The Data Server recognizes the customer with the area of the square in the neighborhood record framework.

Stage 20: The MDFS customer peruses the mentioned number of bytes of the square. Stages 4-19 are rehashed on the off chance that there are different squares to be perused. When the read activity is finished, the square is erased for security motivations to reestablish the first condition of the group.

ii) File write operation

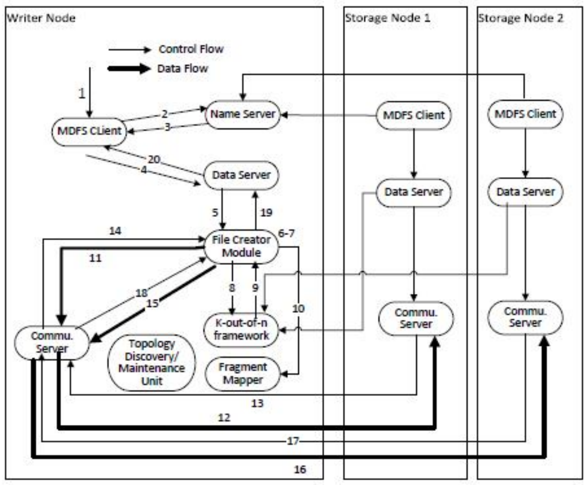

Figure 6: Data flow of write operation

Figure shows the control stream of a compose activity through these numbered advances

Stage 1: The client gives a compose demand for a document of length L. The record is part into squares of size [L/B] where B is the client designed square size. The last square may not be finished relying upon the record length. The client solicitation can likewise be a gushing compose where the client keeps in touch with the document framework byte by byte. When the square limit is come to or when the document is shut, the square is kept in touch with the system. In the two situations, the information to be composed is thought to be available in the nearby record framework.

Stage 2: Similar to HDFS square allotment conspire, for each square to be composed, the MDFS customer demands the Name Server to designate another square Id which is a one of a kind identifier for each square. As every one of the identifiers are created by a solitary Name Server in a concentrated engineering, there won't be any identifier. Be that as it may, in the dispersed engineering, a fitting hashing work is required to produce the novel worldwide identifier. In our usage, the outright way of each record is utilized as the hash key.

Stage 3: The Name Server restores another square id in light of the assignment calculation and includes the square identifier in its nearby store. The mapping of record to list of squares is put away in the Name Server.

Stages 4to5: The MDFS customer gives a creation demand to the Data Server which contains a particular opcode in the solicitation message. The Data Server distinguishes the opcode and starts up the File Creator module to handle the square creation.

Stage 6: The square put away in the neighborhood record framework is encoded utilizing the mystery key. The encoded square is parceled into n parts utilizing deletion encoding.

Stage 7: The key is likewise part into pieces utilizing Shamir's mystery key sharing calculation.

Stages 8,9: The Data Server demands the k out of n structure to give n stockpiling hubs with the end goal that aggregate expected transmission cost from any hub to k nearest capacity hubs is insignificant.

Stage 10: The Data Server demands the Fragment Mapper to include the part data of each record which incorporates the part identifier with the new areas returned by the k out of n system. On the off chance that the arrange topology changes after the underlying calculation, k-out-of-n system recomputes the capacity hubs for each record put away in the system and updates the Piece Mapper. This guarantees Fragment Mapper is consistently in a state of harmony with the present system topology.

Stages 11-18: The document pieces are circulated in equal over the group. The key pieces are additionally put away in a similar way. These subtleties are definitely not remembered for the graph for quickness.

Stages 19to20: Once the record and key sections are disseminated over the group, the Data Server educates the customer that the record has been effectively composed to the hubs. For security purposes, the first square put away in the neighborhood document arrangement of the author is erased after the compose activity as it is never again required.

iii) File Append Operation

MDFS underpins append activity which was introduced in Hadoop 0.19. In the event that a client needs to keep in touch with an existing record, the document must be open in annex mode. On the off chance that the client adds information to the document, bytes are added to the last square of the record. Thus, for square affix mode, the last square is added something extra to the neighborhood record arrangement of the essayist and the record pointer is refreshed properly to the last composed byte. At that point, composes are executed in a comparative route as portrayed in the past segment.

iv) File Delete Operation

For a record to be erased, all document sections of each square of the record must be erased. At the point when the client gives a document erase demand, the MDFS customer inquiries the Name Server for every one of the squares of the document. It at that point demands the Information Server to erase these squares from the system. The Data Server accumulates data about the record pieces from the Fragment Mapper and solicitations the Communication Server to send erase solicitations to every one of the areas returned by the Fragment Mapper. Once the erase demand has been effectively executed, the comparing section in the Fragment Mapper is expelled. If there should be an occurrence of the appropriated engineering, the update must be communicated to the system so that the passage is erased from all hubs in the system.

v) File Rename Operation

The File Rename activity requires just an update in the namespace where the document is referenced with the new way name rather than the old way. At the point when the client issues a document rename demand, the MDFS customer demands the Name Server to refresh its namespace. The Name Server refreshes the current inode structure of the record in view of the renamed way.

vi) Directory Create/Delete/Rename Operations

At the point when the client gives the record directions to make, erase or then again rename any catalog, the MDFS customer demands the Name Server to refresh the namespace. The namespace keeps a mapping of each record to its parent registry where the highest level is the root index ('/'). All ways from the root hub to the leaf hubs are one of a kind. Recursive activities are likewise took into account erase and rename activities.

4.PERFORMANCE EVALUATION

4.1 Effect of Cluster Size on Job Completion Time

The cluster size decides th degree of conceivable parallelization in the bunch. As the group size increments, more errands can be run in equal, in this manner diminishing the work consummation time. Fig 7(a) shows the impact of cluster size on work consummation time. For bigger documents, there are a few guide assignments that can be worked in equal contingent upon the arranged square size. So the presentation is improved essentially with increment in group size as in the figure. For littler documents, the execution isn't influenced much by the group size, as the presentation gain got as a major aspect of parallelism is similar to the extra expense acquired in the errand arrangement[25].

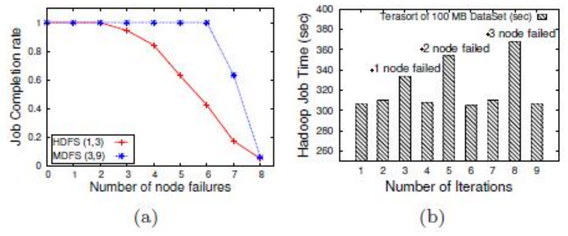

Figure 7: Effect of (a) Comparison of job completion rate Between HDFS and MDFS (b) Job time vs. Number of failures.

4.2 Effect of Input Data Size

Fig 8(a) and Fig 8(b) show the impact of info dataset size on MDFS throughput. The investigation measures the normal peruse and compose throughput for different record sizes. The square size is set to 4 MB. The outcome shows that the framework is less effective with little records because of the overhead in arrangement of creation and recovery errands. Greatest throughput is watched for record sizes that are products of square size. This will decrease the total number of subtasks expected to peruse/compose the entirety document, diminishing the general overhead. In Figure 8(b), the throughput bit by bit increments when the information dataset size is expanded from 1 MB to 4 MB since more information can be moved in a solitary square read/compose demand. In any case, when input dataset size is expanded further, one extra solicitation is required for additional information and along these lines throughput drops unexpectedly. The outcomes show that most extreme MDFS throughput is around 2.83 MB/s for peruses and 2.12 MB/s for composes for document measures that are products of square size.

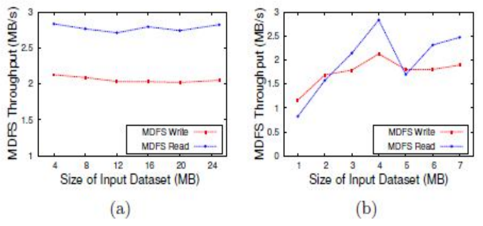

Figure 8: MDFS Read/Write Throughput of (a) Large files(b) Small files

Fig 9 shows the impact of information dataset size on work consummation time. The trial quantifies the activity fulfillment time for various document sizes going from 5 MB to 100MB. Documents produced in cell phones are far-fetched to surpass 100 MB. In any case, MDFS doesn't have any hard point of confinement on input dataset size and it can take any information size permitted in the standard Hadoop discharge. The outcome shows that the activity fulfillment time fluctuates in not exactly direct time with input dataset size. For bigger datasets, there is an adequate number of errands that can be executed in equal over the bunch coming about in better hub usage and improved execution

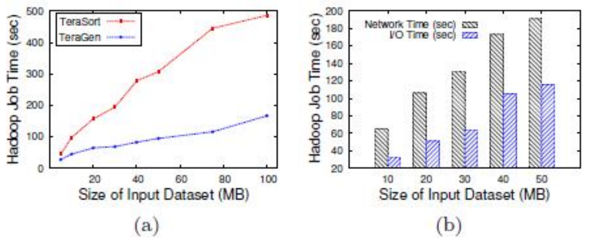

Figure 9 (a) Job Completion time v.s. input dataset size (b) Processing time vs Transmission time

5.CONCLUSION

The Hadoop Map Reduce structure over MDFS shows the abilities of cell phones to capitalize on the relentless development of huge information in the portable condition. Our framework tends to every one of the limitations of information preparing in versatile cloud - vitality effectiveness, information unwavering quality and security. The assessment results show that our framework is skilled for huge information investigation of unstructured information like media documents, content and sensor information. Our presentation results look encouraging for the organization of our framework in genuine groups for large information scientific of unstructured information like media records, content and sensor information.

REFERENCES

[1] S. Huchton, G. Xie, and R. Beyerly, "**Building and evaluating a k-resilient mobile distributed file system resistant to device compromise**," in Proc. MILCOM,2011.
[2] G. Huerta-Canepa and D. Lee, **"A virtual cloud computing provider for mobile devices,"** in Proc. of MobiSys, 2010.
https://doi.org/10.1145/1810931.1810937
[3] K. Kumar and Y.-H. Lu**, "Cloud computing for mobile users: Can offloading computation save energy?"** Computer, 2010.
[4] S. George, Z. Wei, H. Chenji, W. Myounggyu, Y. O. Lee, A. Pazarloglou, R. Stoleru, and P. Barooah, **"Distressnet: a wireless ad hoc and sensor network architecture for situation management in disaster response,"** Comm. Mag., IEEE, 2010.
[5] J.-P. Hubaux, L. Butty´an, and S. Capkun**, "The quest for security in mobile ad hoc networks,"** in Proc. of MobiHoc, 2001.
[6] H. Yang, H. Luo, F. Ye, S. Lu, and L. Zhang, **"Security in mobile ad hoc networks: challenges and solutions,"** Wireless Communications, IEEE, 2004.
[7] C. A. Chen, M. Won, R. Stoleru, and G. Xie, **"Resource allocation for energy efficient k-out-of-n system in mobile ad hoc networks,"** in Proc. ICCCN, 2013.

[8] C. Chen, M. Won, R. Stoleru, and G. Xie, **"Energy-efficient fault-tolerant data storage and processing in dynamic net- work,"** in MobiHoc, 2013.
https://doi.org/10.1145/2491288.2491325
[9] K. Shvachko, H. Kuang, S. Radia, and R. Chansler**, "The hadoop distributed file system,"** in Proc. of MSST, 2010.
[10] J. Dean and S. Ghemawat, **"Mapreduce: Simplified data processing on large clusters,"** Commun. ACM, 2008.
[11] E. E. Marinelli**, "Hyrax: Cloud computing on mobile devices using mapreduce,"** Master's thesis, School of Computer Science Carnegie Mellon University, 2009.
[13] T. Kakantousis, I. Boutsis, V. Kalogeraki, D. Gunopulos, G. Gasparis, and A. Dou, **"Misco: A system for data analysis applications on networks of smartphones using mapreduce,"** in Proc. Mobile Data Management, 2012.
https://doi.org/10.1109/MDM.2012.75
[14] P. Elespuru, S. Shakya, and S. Mishra **"Mapreduce system over heterogeneous mobile devices,"**, in Software Technologies for Embedded and Ubiquitous Systems, 2009.
[15] F. Marozzo, D. Talia, and P. Trunfio**, "P2p-mapreduce: Parallel data processing in dynamic cloud environments,"** J. Comput. Syst. Sci., 2012.
[16] C. A. Chen, M. Won, R. Stoleru, and G. G. Xie, **"Energy- efficient fault-tolerant data storage and processing in dynamic networks,"** in Proc. of MobiHoc, 2013.
[17] Saravanan, V. Venkatachalam, **"Advance Map Reduce Task Scheduling algorithm using mobile cloud multimedia services architecture"** IEEE Digital Explore, pp.21-25,2014.
https://doi.org/10.1109/ICoAC.2014.7229736
[18] K.Sindhanaiselvan, T.Mekala **, "A Survey on Sensor Cloud: Architecture and Applications",** International Journal of P2P Network Trends and Technology (IJPTT), Vol.6, PP.1-6,2014.
[19] Allae Erraissi, Abdessamad Belangour, **"Meta-modeling of Big Data management layer",** International Journal of Emerging Trends in Engineering Research (IJETER),Vol.7,PP.36-43,2019**.**
https://doi.org/10.30534/ijeter/2019/01772019
[20] Amita Dhankhar, Kamna Solanki "**A Comprehensive Review of Tools & Techniques for Big Data Analytics",** International Journal of Emerging Trends in Engineering Research (IJETER),Vol.7,PP.556-562,2019.
https://doi.org/10.30534/ijeter/2019/257112019
[21].**Pandiaraja, P,** Vijayakumar, P, Vijayakumar, V & Seshadhri, R, '**Computation Efficient Attribute Based Broadcast Group Key Management for Secure Document Access in Public Cloud'**, Journal of Information Science and Engineering, 33, No. 3, pp. 695-712
[22]. S.Sravanan , T.Abiramai , and P.Pandiaraja, '**Improve Efficient Keywords Searching Data Retrieval Process in Cloud Server** ", 2018 International Conference on Intelligent Computing and Communication for Smart World (I2C2SW) .PP 219 -223.
[23]. P.RajeshKanna and P.Pandiaraja," **An Efficient Sentiment Analysis Approach for Product Review using Turney Algorithm**", International Conference on Recent Trends in Advanced Computing (ICRTAC 2019). **Journal of Procedia Computer Science , Elsevier ,**Vol 165 ,Issue 2019, Pages 356-362.
[24]. S.Thilagamani , N. Shanthi,**Object Recognition Based on Image Segmentation and Clustering**, Journal of Computer Science, Volume 7,No.11,pp. 1741-1748, 2011.
https://doi.org/10.3844/jcssp.2011.1741.1748
[25]. P.santhi, G. Mahalakshmi, ,**Classification of magnetic resonance images using eight directions gray level co-occurrence matrix (8DGLCM) based feature extraction**, International journal of engineering and advanced technology, volume 8,No.4,pp.839-846,2019.
[26]. W N Hussein1 , L M Kamarudin2 , M R Hamzah3 , H N.Hussain4 , K J Jadaa, **A Methodology for Big Data Analytics and IoT-Oriented Transportation System for future implementation ,** International Journal of Emerging Trends in Engineering Research, International Journal of Emerging Trends in Engineering Research ,Vol 7,No11,pp 449-453.
https://doi.org/10.30534/ijeter/2019/087112019
[27]Amita Dhankhar and Kamna Solanki, **A Comprehensive Review of Tools & Techniques for Big Data Analytics**, International Journal of Emerging Trends in Engineering Research, International Journal of Emerging Trends in Engineering Research ,Vol 7,No11,pp 556-562.
https://doi.org/10.30534/ijeter/2019/257112019