Journal of Theoretical and Applied Information Technology
15th June 2017. Vol.95. No 11
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HIT-RATIO STORAGE-TIER PARTITION STRATEGY OF TEMPEATURE DATA OVER HADOOP

[1]NAWAB MUHAMMAD FASEEH QURESHI, [2*]DONG RYEOL SHIN, [3]ISMA FARAH SIDDIQUI, [4]ASAD ABBAS

[1,2] Department of Computer Science and Engineering, Sungkyunkwan University, Suwon, South Korea

[3,4] Department of Computer Science and Engineering, Hanyang University ERICA, Ansan, South Korea

[*]Corresponding Author

E-mail: [1]faseeh@skku.edu, [2*]drshin@skku.edu

ABSTRACT

Big data analytics refers to a simplified solution of managing huge datasets in a distributed computing environment. Apache Hadoop is an open-source solution that processes large datasets in parallel scenario. The ecosystem consists of three-tier architecture i.e. client, Namenode and Datanode. When a client requests a task processing, Namenode assigns resources and schedules the task over slave node. The Datanode processes the task and returns an output over heterogeneous storage-tier of the cluster. The dataset used for task processing can be categorized into four types i.e. Hot, Warm, Cold and Frozen data. The temperature feature indicates the frequency of accessibility to a dataset and facilitates storage areas i.e. DISK and ARCHIVE. Since the data temperature is identified through number of blocks used per day and data age i.e. 7 days, the storage-tier bears an additional task burden to monitor the changes. Moreover, the temperature data is not relocated but declared with a mark of DISK and ACHIVE proforma. Therefore, storage-tier suffers from data block contention and I/O accessibility latency issues. To overcome these issues, we propose Hit-Ratio Storage-tier Partition Strategy (HRSP), which create logical partitions i.e. Hot partition over a storage-tier media. The presented approach provides direct accessibility of temperature data and reduces I/O accessibility latency and block contention problem. The experimental results depict that the proposed approach disposes the concept of defragmentation and improves I/O accessibility of Hadoop cluster.

Keywords: *Hadoop, HDFS, Data Temperature, Network contention, Storage-tier.*

1. INTRODUCTION

Big data analytics has made simplification in accessing, storing and processing large-scale datasets over a parallel and distributed environment [1]. This activity involves an ecosystem i.e. Apache Hadoop [2], MapR [3] and Cloudera [4]. Hadoop is an open-source programming ecosystem that manages distributed datasets. The ecosystem consists of four core components i.e. YARN [5], Hadoop commons, MapReduce [6] and HDFS [7]. YARN is a central processing component that manages resource allocation and schedules jobs over the cluster. Hadoop commons are the library utilities for managing job executions. MapReduce is a programming module that processes large-scale datasets through assigned jobs. The Hadoop Distributed File System (HDFS) is the ecosystem's file system and provides a storage medium to store and retrieve large datasets over the cluster [8][9].

HDFS consists of three-tier architecture i.e. Namenode, Datanode and client. Therefore, when a client request a dataset operation, Namenode allocates resources and schedules task over a Datanode. The slave node process the dataset operation and returns an output over HDFS cluster [10] [11].

The data temperature is a frequency of accessing a file over a time. There are four types of data temperatures i.e. Hot, Warm, Cold and Frozen dataset [12] [13]. By default, a hot, warm and cold dataset is stored with a DISK identification mark and frozen dataset is marked with ARCHIVE identification [10] [14] as seen from Figure-1. Recently, Hadoop has transformed storage architecture to heterogeneous storage-tier. The heterogeneity-aware storage involves the usage of

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

multiple storage medias i.e. SSD, RAM and DISK [15] [16] [17]. In this way, HDFS scatters DISK and ARCHIVE marked datasets over storage media and produces data block contention and I/O accessibility latency issues [18] [19] [20].

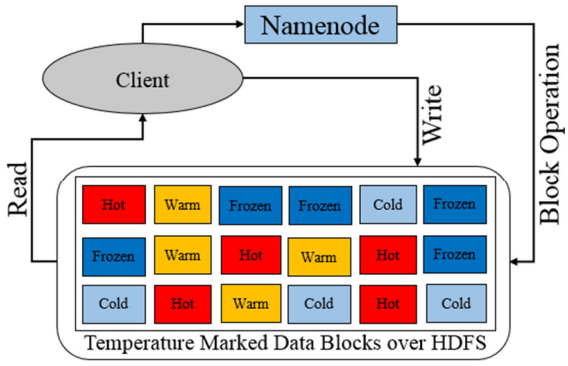

Figure 1: Default Temperature Data HDFS Cluster

To resolve this problem, we propose Hit-Ratio storage-tier partition strategy (HRSP) that aliens marked temperature data blocks to respective partitions and build custom metadata indexes for efficient accessibility. The proposed approach provides a contention-free data block access and decreases I/O accessibility latency.

The main contributions of the proposed scheme are:

- A novel temperature data partition blocks.
- A novel temperature data search strategy.
- A compact hit-ratio observer

The remaining paper is organized as follows. Section II discusses related work. Section III briefly explains proposed approach HRSP. Section IV depicts experimental environment and evaluation result for HRST partition strategy. Finally, section V shows conclusion and future research directions.

2. RELATED WORK

The data temperature mark strategy is relatively a new phenomenon, which categorizes data block accessibility for general task processing [21] [22] [23] [24]. Initially, this concept was introduced by Rohith Subramanyam [25] and later Apache Hadoop adopted with few changes in their architecture [10]. Moreover, many researchers presented their approaches to optimize temperature marked data block accessibility. Jianjiang Li [26] uses data temperature property to back up hot data blocks through an encoding decoding compression technique. This space optimization strategy reduces redundancy problems but do not consider index accessibility problem. Moreover, the backup strategy consumes a huge time overhead over searching hot data blocks. PRESIDIO [30] strategy uses ARCHIVE data temperature property and reduces data redundancy problem over active part of secondary storage area. Moreover, the approach accesses ARCHIVE section and matches storage objects to reduce space workload. The approach creates a virtual content addressable storage to manage block indexes and generates a huge time overhead to access, copy and paste alike ARCHIVE blocks. Fabien Andre [27] presents a reliable data temperature storing strategy with erasure coding approach but limits it over homogeneous storage-tier functionality [28].

Keeping in view of limited schemes to alien temperature data over a heterogeneous storage-tier cluster, we present HRSP that manages temperature-aware data blocks over a partition scheme in storage media. The proposed approach systematically resolves data temperature block accessibility latency and contention issue of multiple storage-tier devices over Hadoop cluster.

3. HIT-RATIO STORAGE TIER PARTITION STRATEGY (HRSP)

The proposed approach works in three steps i.e. (i) Temperature Data Block locator, (ii) Hit-ratio observer, (iii) Temperature-based Data Block partition table.

The proposed approach HRSP perform a custom partition of storage media i.e. Hot partition, warm partition, cold partition and frozen partition in a Datanode as observed from Figure-2.

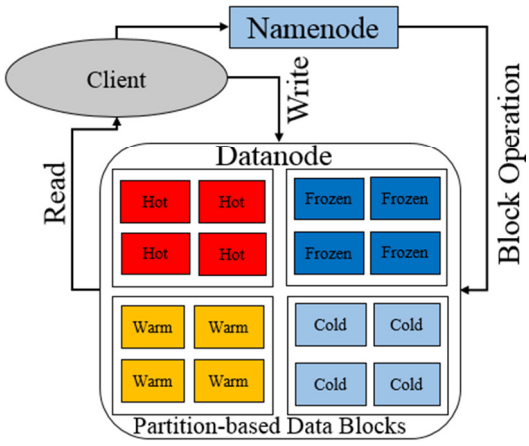

Figure 2: Partition-based Temperature Data blocks over HDFS Cluster

Journal of Theoretical and Applied Information Technology
15th June 2017. Vol.95. No 11
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

3.1 Temperature Data Block Locator

The locator fetches element from dataset having set of temperature mark $Temp_m$ and data block size D_s. The collection of marked data blocks can be represented as,

$$Data_m = (Temp_m, D_s) \quad (1)$$

A HDFS-aware algorithm TSPLFC [29] is used to scan temperature categorized i.e. Hot, Warm, Cold and Frozen elements over $Data_m$. There are two changes in current algorithm i.e. (i) Scanning Data block metadata at a time and (ii) Simultaneous identification of dataset elements $Temp_m$ and D_s.

Suppose $Data_m$= [B_1, B_2…m-1,0…n-1] temperature blocks are processed over marked temperature $Temp_m$ = [Hot, Warm, Cold, Frozen]. The enhanced TSPLFC algorithm compare $Data_m$ and D_s over pattern of $Temp_m$ and fetch result set in array of $Mark_s$. The scanning process continues till $Data_m$ [0] and D_s [m-1] do not find $Temp_m$ [i+1…s-1] and exit algorithm execution by generating data block indexes. In the same way, $Data_m$ [0] and D_s [n-1] continues reverse order scanning until $Temp_m$ [i+1…s-1] and produces an array of scanned elements S_m to compile temperature data blocks as seen from Figure-4.

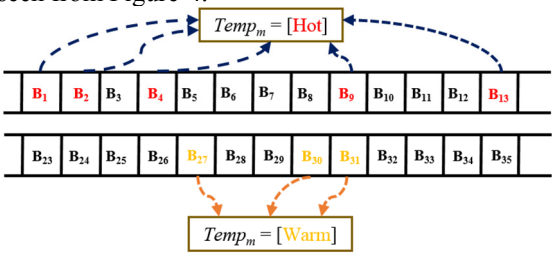

Figure 3: Temperature Block Scanning Over HDFS

3.2 Hit-ratio Observer

When a client request an operation over data block array S_m, Namenode buffers hit-ratio of a block and store it over a over a DFS file. A single hit-ratio can be collected as,

$$Hit_{Ratio} = \frac{S_{m_{Block}}}{Operation\ Task} \quad (2)$$

The collection of hit-ratio for temperature data blocks can be represented as,

$$Temp_{Blocks} = \sum S_{m_{Block}}(H, C, W, F) \quad (3)$$

The DFS write socket fetch $Temp_{Block}$ index and perform data block placement function to relocate temperature-aware data blocks over partition-based storage media as seen from Figure-5.

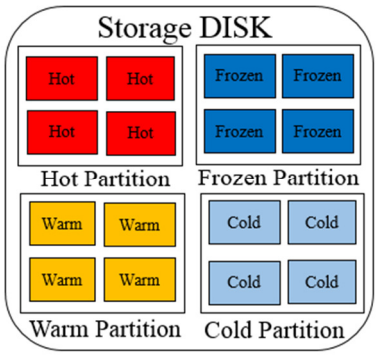

Figure 5: Temperature Data Partition Over DISK

3.3 Temperature Data Block Partition Table

The partition table consists of four tables i.e. Hot partition, Warm partition, Cold partition and Frozen partition. The hot partition reserves a space volume of T_s $\frac{1}{4}$ size and index tag I_H over the storage media. The table configuration of hot, cold, warm and frozen data blocks can be expressed as,

$$Table_{Hot} = \sum_{(T_S \in Data_m(DISK))}^{I_H \leq S_m} (N_c, M_D, I_S) \quad (4)$$

$$Table_{Cold} = \sum_{(T_S \in Data_m(DISK))}^{I_C \leq S_m} (N_c, M_D, I_S) \quad (5)$$

$$Table_{Warm} = \sum_{(T_S \in Data_m(DISK))}^{I_W \leq S_m} (N_c, M_D, I_S) \quad (6)$$

$$Table_{Frozen} = \sum_{(T_S \in Data_m(DISK))}^{I_F \leq S_m} (N_c, M_D, I_S) \quad (7)$$

Where N_C, M_D and I_S represents cluster, Datanode and storage-tier parameter information.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate HRSP over cluster configuration as seen from Table-1.

Table 1: Hadoop Cluster.

Machine	Specifications	No. of VM	
Intel Xeon E5-2600 v2	8 CPUs, 32GB memory, 1T Disk and 128 GB SSD	3	1 Master Node, 2 Datanodes
Intel core i5	4 Core, 16GB memory, 1T Disk and 128 GB SSD	2	2 Datanodes
Hadoop	Hadoop-2.7.2 (stable)		
Virtual Machine Management	VirtualBox 5.0.16		

4.1 Environment

The Hadoop cluster comprises of Intel Xeon with 8 CPUs, 32GB memory and storage devices i.e. 1TB Hard disk drive and 128GB Samsung SSD. Moreover, we also used Intel core

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

i5 with 4 Core, 16GB memory and storage devices i.e. 1TB Hard disk drive and 128 GB Samsung SSD. We used virtualbox 5.0.16 for installing 5 virtual machines on depicted cluster configurations as observed from Table- 2.

Table 2: Hadoop Cluster Virtual Machines Configuration.

Node	CPU	Memory	Disk	Configuration
Master Node	6	16 GB	HDD & SSD	Intel Xeon
Slave1	2	4GB	HDD & SSD	Intel Xeon
Slave2	2	4GB	HDD & SSD	Intel Core i5
Slave3	2	4GB	HDD & SSD	Intel Core i5
Slave4	2	4GB	HDD & SSD	Intel Core i5

4.2 Experimental Dataset

The dataset used to perform experimental work includes: (i) 200 marked Hot data blocks of 64MB (12.8GB size), (ii) 200 marked Cold data blocks of 64MB (12.8GB size) (iii) 200 marked Warm data blocks of 64MB (12.8GB size) and (iv) 200 marked Frozen data blocks of 64MB (12.8GB size) [31].

4.3 Experimental Results

The experiments performed to evaluate HRSP strategy are: (i) Data Temperature Block I/O processing and (ii) Contention measures over Temperature-aware Data blocks.

4.3.1 Data Temperature Block I/O processing

The temperature-aware data block i.e. hot, cold, warm and frozen requires an input operation INVOKE_BLOCK over a client function and returns a metadata file of block index to Namenode [32] [33]. Moreover, hit-ratio calculates temperature type and performs block placement procedure over respective partition table of HDFS storage-tier. In this way, we processed '200' marked hot blocks over default and partition table-based I/O processing and evaluated that Table-based hot blocks consumed 52.62% less processing time than default block processing as observed from Figure-6.

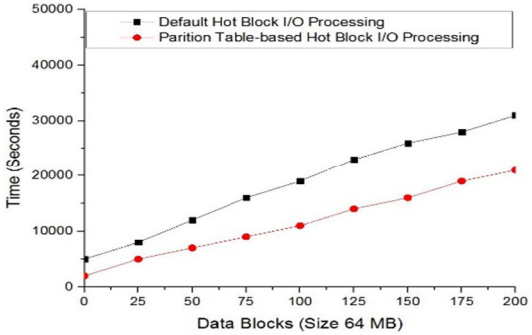

Figure 6: Hot Marked Data Blocks I/O Processing

Similarly, we processed '200' marked cold blocks over default and partition table-based I/O processing and evaluated that Table-based cold blocks consumed 69.3% less processing time than default block processing as observed from Figure-7.

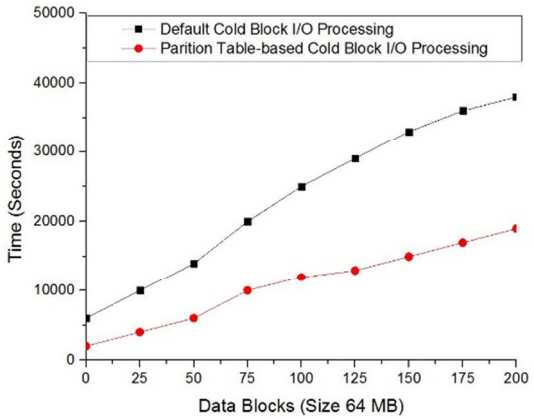

Figure 7: Cold Marked Data Blocks I/O Processing

Moreover, we processed '200' marked warm blocks over default and partition table-based I/O processing and evaluated that Table-based warm blocks consumed 54.7% less processing time than default block processing as observed from Figure-8.

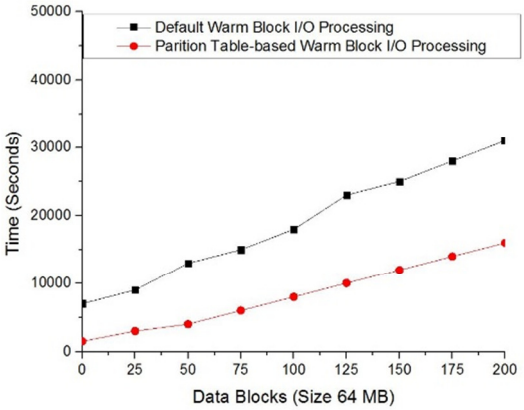

Figure 8: Warm Marked Data Blocks I/O Processing

Furthermore, we processed '200' marked frozen blocks over default and partition table-based I/O processing and evaluated that Table-based frozen blocks consumed 49.5% less processing time than default block processing as observed from Figure-9.

The performance metrics for calculating processing timestamp depends over two factors i.e. (i) Local index search timestamp and (ii) Fetch and block placement timestamp

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

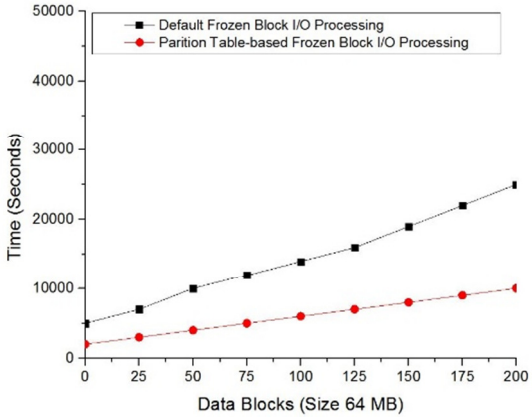

Figure 9: Frozen Marked Data Blocks I/O Processing

4.3.2 Contention observation over Temperature-aware Data blocks

The temperature-aware data blocks are accessed through indexes of metadata file over a Datanode. The data block involves a huge delay of accessing single index among millions of indexes in Namenode. The sole reason of this latency is related to random placement of temperature data blocks [34]. After the blocks are placed over partition table, the index accessibility latency is reduced to optimal level. To observe the discussed difference, we processed '200' hot blocks indexes over default and partition table-based index accessibility and found that Table-based hot block indexes consumes 47.24% less accessibility time than default block accessibility timestamp as seen from Figure-10.

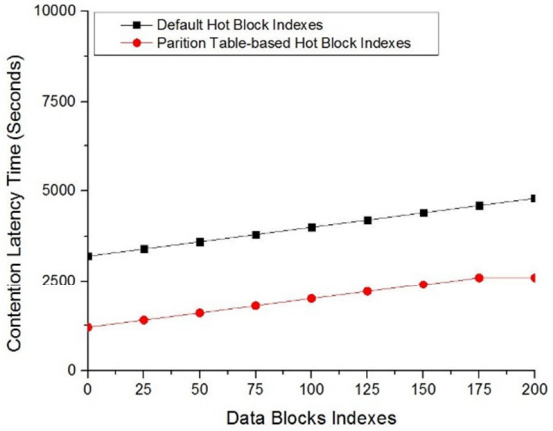

Figure 10: Hot Data Blocks Indexes Accessibility

Similarly, we processed '200' cold blocks indexes over default and partition table-based index accessibility and evaluated that Table-based hot block indexes consumes 61.71% less accessibility time than default block accessibility timestamp as seen from Figure-11.

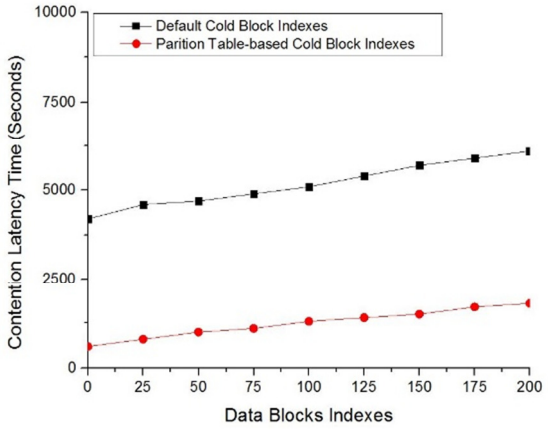

Figure 11: Cold Data Blocks Indexes Accessibility

Moreover, we processed '200' warm blocks indexes over default and partition table-based index accessibility and observed that Table-based warm block indexes consumes 43.73% less accessibility time than default block accessibility timestamp as seen from Figure-12.

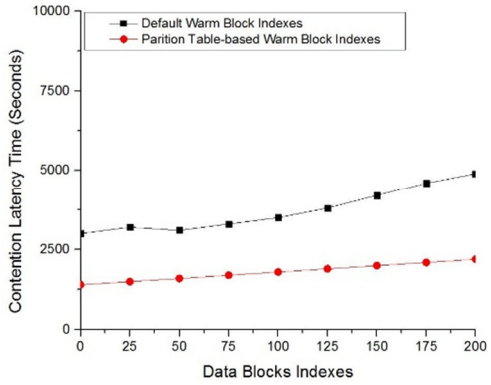

Figure 12: Warm Data Blocks Indexes Accessibility

Furthermore, we processed '200' frozen blocks indexes over default and partition table-based index accessibility and evaluated that Table-based frozen block indexes consumes 56.51% less accessibility time than default block accessibility timestamp as seen from Figure-13.

5. CONCLUSION

This paper proposes Hit-Ratio storage-tier partition strategy (HRSP) that generates temperature-aware data block partitions over storage-tier HDFS. The proposed approach measures hit-ratio of marked temperature data blocks and performs block placement to alien temperature-aware blocks over partition-table. The

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

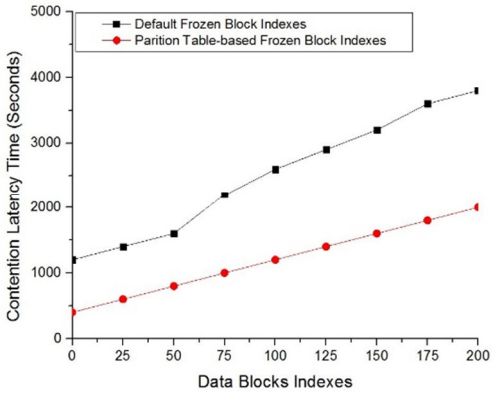

Figure 13: Frozen Data Blocks Indexes Accessibility

results depict that HRSP reduces client operation timestamp for accessing marked data blocks and decreases contention latency between storage-tier indexes accessibility.

In future, we focus to expend partition tables accessibility to multihoming nodes.

ACKNOWLEDGEMENT

This work was supported by Institute for Information & communications Technology Promotion(IITP) grant funded by the Korea government(MSIP) (No. R0113-15-0002, Automotive ICT based e-Call standardization and after-market device development)

REFRENCES:

[1] LaValle, Steve, et al. "Big data, analytics and the path from insights to value." MIT sloan management review 52.2, 2011, pp. 21.

[2] "Welcome to Apache™ Hadoop®!" 2014. [Online]. Available: http://hadoop.apache.org/. Accessed: Mar. 13, 2017.

[3] M. Technologies, "Featured customers", 2016. [Online]. Available: https://www.mapr.com/. Accessed: Mar. 13, 2017.

[4] Cloudera, "The modern platform for data management and analytics," Cloudera, 2016. [Online]. Available: http://www.cloudera.com/. Accessed: Mar. 13, 2017.

[5] "Apache Hadoop 2.7.2 – Apache Hadoop YARN," 2016. [Online]. Available: https://hadoop.apache.org/docs/r2.7.2/hadoop-yarn/hadoop-yarn-site/YARN.html. Accessed: Mar. 13, 2017.

[6] "Apache Hadoop 2.7.2 – MapReduce Tutorial," 2016. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html. Accessed: Mar. 13, 2017.

[7] "Apache Hadoop 2.7.2 – HDFS users guide," 2016. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsUserGuide.html. Accessed: Mar. 13, 2017.

[8] A. Kala Karun and K. Chitharanjan, "A review on Hadoop — HDFS infrastructure extensions," 2013 IEEE CONFERENCE ON INFORMATION AND COMMUNICATION TECHNOLOGIES, Apr. 2013.

[9] Abbas, A., Wu, Z., Siddiqui, I. F., & Lee, S. U. J. (2016). An approach for optimized feature selection in software product lines using union-find and Genetic Algorithms. Indian Journal of Science and Technology, 9(17)

[10] "Apache Hadoop 2.7.2 – HDFS storage-tier," 2016. [Online]. Available: https://hadoop.apache.org/docs/r2.7.3/hadoop-project-dist/hadoop-hdfs/ArchivalStorage.html Accessed: Mar. 13, 2017.

[11] Abbas, A., Siddiqui, I. F., & Lee, S. U. J. (2016). Multi-Objective Optimization of Feature Model in Software Product Line: Perspectives and Challenges. Indian Journal of Science and Technology, 9(45).

[12] Y. Tsuruoka, "Cloud computing - current status and future directions," Journal of Information Processing, vol. 24, no. 2, 2016, pp. 183–194.

[13] ABBAS, A., SIDDIQUI, I. F., & LEE, S. U. J. (2017). CONTEXTUAL VARIABILITY MANAGEMENT OF IOT APPLICATION WITH XML-BASED FEATURE MODELLING. Journal of Theoretical & Applied Information Technology, 95(6).

[14] C. Rodríguez-Quintana, A. F. Díaz, J. Ortega, R. H. Palacios, and A. Ortiz, "A new Scalable approach for distributed Metadata in HPC," in Algorithms and Architectures for Parallel Processing. Springer Nature, 2016, pp. 106-117.

[15] T. White, Hadoop: The definitive guide, "O'Reilly Media, Inc.", 2012.

[16] Abbas, A., Siddiqui, I. F., & Lee, S. U. J. (2016). GOAL-BASED MODELING FOR REQUIREMENT TRACEABILITY OF SOFTWARE PRODUCT LINE. Journal of Theoretical and Applied Information Technology, 94(2), 327.

[17] Abbas, A., Siddqui, I. F., Lee, S. U. J., & Bashir, A. K. (2017). Binary Pattern for Nested Cardinality Constraints for Software Product

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Line of IoT-based Feature Models. IEEE Access.

[18] N.M.F Qureshi, et al. "KEY EXCHANGE AUTHENTICATION PROTOCOL FOR NFS ENABLED HDFS CLIENT", *Journal of Theoretical and Applied Information Technology*, vol. 95, no. 7, pp. 1353-1361, 2017.

[19] I.F Siddiqui, et al. "Comparative Analysis of Centralized Vs. Distributed Locality-based Repository over IoT-Enabled Big Data in Smart Grid Environment", *Proceedings of the Korean Society of Computer Information Conference*, vol. 25, pp. 75-79, 2017.

[20] I.F. Siddiqui, et al. "A HIDDEN MARKOV MODEL TO PREDICT HOT SOCKET ISSUE IN SMART GRID", *Journal of Theoretical and Applied Information Technology*, vol. 94, no. 2, pp. 408-415, 2016.

[21] I.F. Siddiqui, et al. "A Comparative Study of Multithreading APIs for Software of ICT Equipment," J. Indian Journal of Science and Technology, vol. 9, no. 48, pp. 1-5, Dec. 2016.

[22] I.F. Siddiqui, et al. "A Framework for Verifying Consistency of SQL-DB Ontology using Alloy," In Proc. 16th Korea Computer Congress, 2014, pp.497-499.

[23] I.F. Siddiqui, et al. "Access Control as a Service for Information Protection in Semantic Web based Smart Environment," J. Journal of Korean Society for Internet Information, vol. 17, no. 5, pp. 9-16, Oct. 2016.

[24] I.F. Siddiqui, et al. "Privacy-Aware Smart Learning: Providing XACML as a Service in Semantic Web based Smart Environment," In Proc. 7th International Conference on Internet Symp., 2015, pp.97-101.

[25] Subramanyam, R. (2015, September). HDFS Heterogeneous Storage Resource Management Based on Data Temperature. In *Cloud and Autonomic Computing (ICCAC), 2015 International Conference on* (pp. 232-235).

[26] Li, J., Zhang, P., Li, Y., Chen, W., Liu, Y., & Wang, L. (2017). A data-check based distributed storage model for storing hot temporary data. *Future Generation Computer Systems*.

[27] André, Fabien, et al. "Archiving cold data in warehouses with clustered network coding." *Proceedings of the Ninth European Conference on Computer Systems*. ACM, 2014.

[28] A. Rasheed, and M. Mohamed. "Fedora Commons with Apache Hadoop: A Research Study", 2013.

[29] Zubair, Muhammad, et al. "Improved text scanning approach for exact String matching." *Information and Emerging Technologies (ICIET), 2010 International Conference on*. IEEE, 2010.

[30] You, Lawrence L., et al. "PRESIDIO: a framework for efficient archival data storage." *ACM Transactions on Storage (TOS)* 7.2 (2011): 6.

[31] N. M. F. Qureshi, and D. R. Shin, "RDP: A storage-tier-aware Robust Data Placement strategy for Hadoop in a Cloud-based Heterogeneous Environment", KSII Transactions on Internet and Information Systems, vol. 10, no. 9, 2016, pp. 4063-4086.

[32] Lee, Kisung, and Ling Liu. "Efficient data partitioning model for heterogeneous graphs in the cloud." *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*. ACM, 2013.

[33] S. Mazumder, "Big Data Tools and Platforms." Big Data Concepts, Theories, and Applications. Springer International Publishing, 2016, pp. 29-128.

[34] S. Magana-zook, et al. "Large-scale seismic waveform quality metric calculation using Hadoop." Computers & Geosciences, 2016.