Journal of Theoretical and Applied Information Technology
30th June 2017. Vol.95. No 12

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

PREDICTIVE ANALYSIS OF LOCALITY-AWARE STORAGE-TIER DATA BLOCKS 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

The term 'Big Data analytics' refers to a large-scale solution for managing giant datasets in a parallel environment. Hadoop is an ecosystem that processes large datasets in distributed computing scenario. The ecosystem is further categorized into four sub-projects i.e. HDFS, MapReduce, YARN and Hadoop Commons. The Hadoop Distributed File System (HDFS) is a backbone of ecosystem, which helps storing and processing large datasets. Recently, HDFS is upgraded to heterogeneous storage-tier environment that cope with data block processing over multiple storage devices i.e. DISK, SSD and RAM. The block placement policy dispatches data blocks to the devices without calculating I/O transfer parameters and locality perspectives. Moreover, HDFS selects random Datanodes that could be located into the next rack having longer path than local rack. This increases the data block processing latency and results in a huge delay for replica management in heterogeneous storage-tier. To resolve this issue, we propose a predictive analysis that build a locality-aware storage-tier node summary and predict the most nearby available storage-tier for block job processing. The experimental evaluation depicts that the proposed approach reduces data block transfer time overhead, replica transfer time overhead and decreases node paths to an optimal accessibility over the cluster.

Keywords: *Hadoop, HDFS, Locality-aware, network distance, storage-tier.*

1. INTRODUCTION

Big data processing has resolved many complex issues of huge dataset processing [1]. There are many system softwares available, which address the dataset processing over parallel computing environment i.e. Cloudera [4], MapR [3] and Apache Hadoop [2]. Hadoop ecosystem is an open-source ecosystem, which processes large-scale huge datasets in a distributed computing phenomena. It is categorized into four main sub-projects i.e. HDFS [7], YARN [5], MapReduce [6] and Hadoop Commons. The Hadoop Distributed File System (HDFS) is a core sub-project, which stores and retrieves datasets over the cluster. Hadoop Commons is a built-in library for providing environment parameters to the cluster processing. MapReduce is a programming module, which processes large datasets in a parallel processing environment and YARN is a brain of the Hadoop that schedules and allocates resources to task processing.

HDFS consists of three components i.e. Namenode, Datanode and client. The Namenode receives a task request from client and allocates scheduling activity and resource parameters to Datanode. The Datanode returns job output and stores data blocks over the storage-tier [8] [9] as seen from Figure-1.

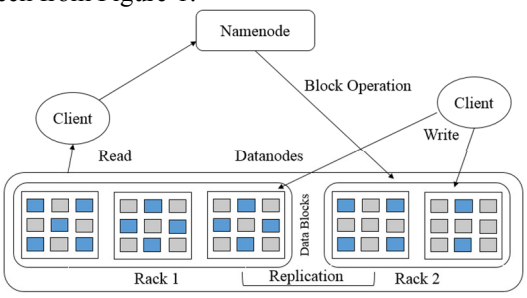

Figure 1: HDFS Architecture

By default, a Namenode processes data blocks to random Datanodes [11] [12]. This increases processing latency $Latency_P = \left(\frac{Path(n)}{Processing\ Time}\right)$ and produces replica latency

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

$Latency_R = (Latency_P \times node(n))$ over the cluster. Moreover, the latency increases with the heterogeneous storage-tier factor, where a single node consists of multiple storage devices i.e. SSD, DISK and RAM. As a result, we observe a data block placement latency $Latency_C = (Latency_R \times Storage(n))$ [10] [13] [14] over Hadoop cluster as observed from Figure-2.

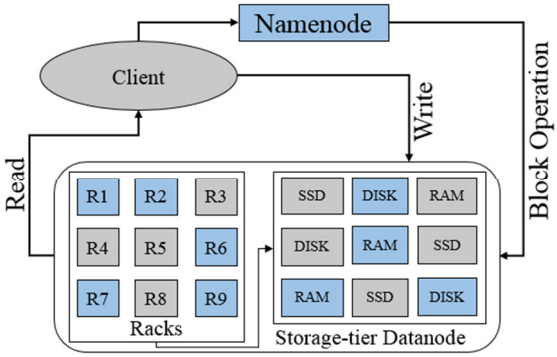

Figure 2: Default Storage-tier Rack Cluster

To solve this problem, we propose a predictive analysis that collects nearby storage-tier Datanodes summary and predicts nearest locality-aware storage-tier media for block job processing. The predicted storage-tier node reduces the $Latency_P$, $Latency_R$ and $Latency_C$ time overheads simultaneous and increases cluster block processing to an optimal state.

The main contributions of the proposed scheme are:

- A novel storage-tier summary collector.
- A novel prediction model for searching nearby storage-tier Datanodes.
- An effective latency reducer for storage-tier cluster.

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

2. RELATED WORK

Many researchers have contributed and presented locality-aware cluster processing strategies. The locality-aware processing can be categorized into two types i.e. (i) Scheduler-based locality-aware processing and (ii) Resources-based locality-aware processing [15] [16] [17]. The scheduler-based locality-aware processing includes scheduling techniques, which generates MapReduce job output as per nearby Datanodes. The resource-based locality-aware processing focuses over data placement technique, which can be executed after a MapReduce job is completed in a default manner [18] [19] [20]. Since, our proposed approach comes into the resource-aware strategy so we highlight contributions for this area as below:

ADAPT [25] is a data placement technique, which dispatches data blocks over availability of storage space in a Datanode. This is a simple technique, which focuses over storage space availability over homogeneous storage-tier. RDP [31] is a robust data placement strategy, which perform block placement over heterogeneous storage-tier but do not consider locality-aware processing. Purlieus [30] uses virtual heterogeneous storage-tier of virtual machines to process data blocks without keeping locality-awareness over cluster. Data pre-loading and data placement technique [29] manages a cache to storage data blocks before dispatching towards the Datanodes. Keeping in view, the performance is better than default approach but lack locality-awareness in block job processing [21] [22] [23] [24]. Dynamic energy efficient data placement [27] strategy uses Datanodes to dispatch data blocks and detach them so to save energy perspective. This strategy focuses over homogeneous storage-tier only and do not consider locality-awareness in block job processing. Investigation of Data Locality [26] technique processes multiple instances of data placement over homogeneous storage-tier cluster.

Since, the discusses techniques lack the concept of locality-aware data placement over storage-tier cluster, we present a predictive analysis of locality-aware data placement technique, which collects heterogeneous storage-tier processing information and predicts most suitable storage node of a Datanode to receive block placement function. The proposed approach reduces three-tier latencies and increases the performance of block placement over Hadoop cluster.

3. PREDICIVE ANALYSIS OF LOCALITY-AWARE STORAGE-TIER DATA BLOCKS

The predictive analysis consists of two phases i.e. (i) Storage-tier summary container and (ii) Predict the most nearby Datanode.

Journal of Theoretical and Applied Information Technology
30th June 2017. Vol.95. No 12

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

The storage-tier summary container collects all the Datanode and storage media information i.e. computing capacity and storage-tier devices with volume statistics. Moreover, the media predictor performs training sessions over the dataset and predicts the most nearby Datanode with available storage-tier media as seen from Figure-3.

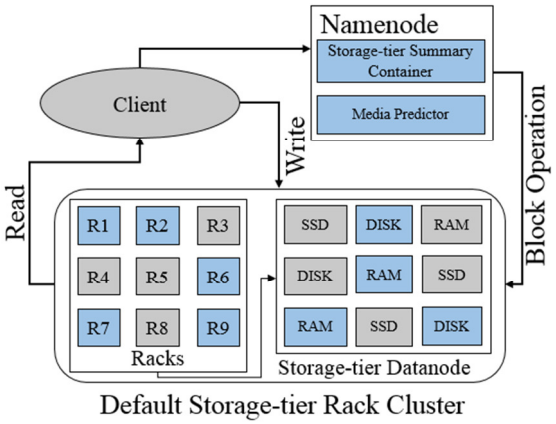

Figure 3: Storage-tier Predictive Analysis over HDFS Cluster

3.1 Storage-tier summary container

The summary container consists of Datanode information i.e. CPU, storage medias, accessibility time, 1 MB data block receiving time and volume sizes of each storage.

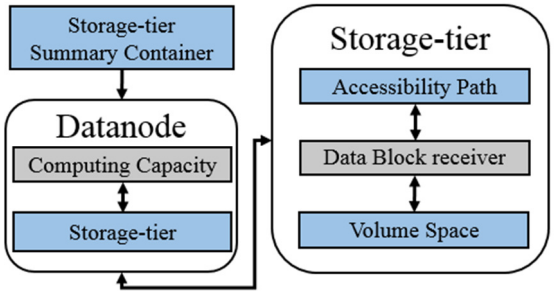

Figure 3: Storage-tier summary Architecture

The container messages are collected through Belief Propagation method [32], which stores information messages in concatenated form. Thus, it reduces complete message read time overhead. Moreover, we perform inference on belief propagation through Message Propagation Model [33] that a message m of a variable component i having value $\varkappa_i$ with a belief $b_i(\varkappa_i)$ can be propagated from source component a to destination component i represents likeliness of random variable X_i where $\varkappa_i \in X_i$ by,

$$message\ m_{a\to i}(\varkappa_i) \quad (1)$$

We store information messages of a Datanode having computing capacity *CC* and Storage-tier *ST* as,

$$message\ m_{CC_i\to D_i}(\varkappa_{D_i}) \quad (2)$$

$$message\ m_{ST_i\to D_i}(\varkappa_{D_i}) \quad (3)$$

We calculate belief of component Datanode *D* having *CC* and *ST* information messages as,

$$b_i(\varkappa_{D_i}) \propto \prod_{(CC_i,ST_i)\in N(D_i)} m_{(CC_i,ST_i)\to D_i}(\varkappa_{D_i}) \quad (4)$$

Similarly, we store information messages of Storage-tier having Accessibility Path *AP*, Datablock receiver *DR* and Volume Space *VS* as,

$$message\ m_{AP_i\to ST_i}(\varkappa_{ST_i}) \quad (5)$$

$$message\ m_{DR_i\to ST_i}(\varkappa_{ST_i}) \quad (6)$$

$$message\ m_{VS_i\to ST_i}(\varkappa_{ST_i}) \quad (7)$$

We calculate belief of component Storage-tier *ST* having *AP* and *ST* information messages as,

$$b_i(\varkappa_{ST_i}) \propto \prod_{(AP_iDR_iVS_i)\in N(ST_i)} m_{(AP_i,DR_i,VS_i)\to ST_i}(\varkappa_{ST_i}) \quad (8)$$

We normalize eq (4) and eq (8) with a constant Z and get,

$$b_i(\varkappa_{D_i}) = \frac{1}{Z}\left(\prod_{(CC_i,ST_i)\in N(D_i)} m_{(CC_i,ST_i)\to D_i}(\varkappa_{D_i})\right) \quad (9)$$

and

$$b_i(\varkappa_{ST_i}) = \left(\prod_{(AP_iDR_iVS_i)\in N(ST_i)} m_{(AP_i,DR_i,VS_i)\to ST_i}(\varkappa_{ST_i})\right) \quad (10)$$

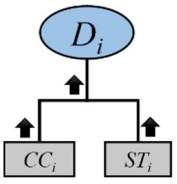

Figure 4(a): Belief of D_i

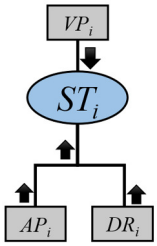

Figure 4(b): Belief of ST_i

Therefore, belief of component D and ST can be expressed from Figure-3(a) and Figure-3(b).

To collect information messages together at a single container i.e. Storage-tier summary container, we calculate joint belief *'L'* of component D_i *and* ST_i as,

$$b_L(\varkappa_L) = b_A(X_A) \quad (11)$$

Where $X_A = \{\varkappa_{D_i}, \varkappa_{ST_i} : D_i, ST_i \in N(A)\}$ and $\varkappa_L$ is the domain space related to component L as seen from Figure-5.

Journal of Theoretical and Applied Information Technology
30th June 2017. Vol.95. No 12

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

The domain space $\varkappa_L$ *is equal to* $\{(\varkappa_{D_i}, \varkappa_{ST_i}) \mid f_A(\varkappa_{D_i}, \varkappa_{ST_i}) = 1,\ \varkappa_{D_i} \in \chi_{D_i}, \varkappa_{ST_i} \in \chi_{N_i}\}$ where factor f_A represents a bipartite string between D_i and ST_i components. Therefore, a joint belief can be observed as,

$$b_L(\varkappa_L) = \frac{1}{Z} \prod_{(D_i,ST_i) \in N(L)} m_{(D_i,ST_i)\to L}(\varkappa_L) \quad (12)$$

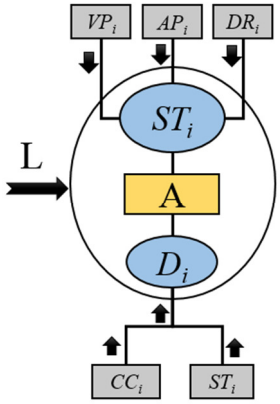

Figure 5: Belief of Factor A (L)

Finally, we simplify eq (12) to close-form solution as,

$$m_{(D_i,ST_i)\to L}(\varkappa_L) = \sum f_A(X_A) \prod_{(D_i,ST_i) \in N(L)} m_{(D_i,ST_i)\to L}(\varkappa_L) \quad (13)$$

Where $m_{(D_i,ST_i)\to L}(\varkappa_L)$ depicts Storage-tier summary container over Namenode.

3.2 Media predictor

To predict locality-aware storage media for data block processing, we use Hidden Markov Model (HMM) [34]. The model works in two steps i.e. (i) Training phase and (ii) Prediction phase. By default, the model uses hidden states $X = \{x_1, x_2\}$, conditional transition probability $A = a_{ij} = \{P[q_{t+1} = x_j | q_t = x_j]\}$, observation state $Y = \{y_1, y_2, y_3, y_4\}$ and emission probability $B = b_{ij}$. Therefore, according to definition of model HMM (λ) we get,

$$\lambda = (\pi, A, B) \quad (14)$$

Where A is the transition matrix, B is the emission matrix and π is initial state transition probability.

To apply media predictor scenario, we process observations $O = \{B_1, B_2, B_3, B_4, B_N\}$ over hidden states $H = \{SSD_n, DISK_n, RAM_n\}$ and generate an association for training phase as seen from Figure-6.

The model completes prediction cycle in time length of Δt and returns storage status and time to predict storage media t_{ST}.

3.2.1 Model Training

The media predictor model gets training through Storage-tier summary container elements. At first, it transits from initial state x_1 to end state x_2 with seed probability π=0.33 and calculates hidden state transition λ. Moreover, it fetches container parameters and train the model through Expectation-Maximization [35] algorithm. In this way, blocks are processed several times until sequence generates best fit resultset for model and produces media names in the dataset.

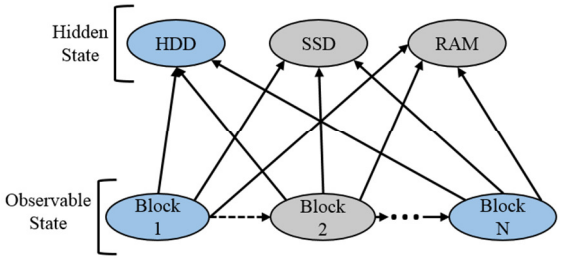

Figure 6: HMM States

The Expectation-Maximization (EM) works in two steps i.e. (i) Expectation and (ii) Maximization. The first step calculates media likelihood from present estimations and maximization step calculates maximizing expected media likelihood. Algorithm-1 shows a procedural workout, where seed blocks are processed over hidden media storages. At first, we calculate state path probability and update the transition and emission probability simultaneous. Moreover, we process path probability with obtained values through EM algorithm and generates pair of likelihood i.e. [(Block$_1$, SSD), (Block$_2$, RAM), (Block$_3$, DISK)].

Algorithm-1. Media Prediction Algorithm

Initialize O = {Block$_1$, Block$_2$, Block$_3$}
S = {SSD, RAM, DISK}
M=3
N=3
Initialize Transprobability(A_{ij}), EmissionProbability(B_{ij})
X = {X_1, X_2, X_3}
Initialize Probability(StatePath)
Update B_{ij},A_{ij}
Probability(Path) = B_{ij} * Probability(StatePath)
  For Each state S_j do
    Probability(StatePath)[j,i] ← max$_k$ (Probability(StatePath)[k,i-1].A_{kj}.B_{jyi})
    Probability(Path)[j,i] ← arg max$_k$ (Probability(Path)[k,i-1].A_{kj}.B_{jyi})
  End For
  C_i ← arg max$_k$ (Probability(StatePath))
  Z_i ← S_i
  For I ← T-1, …, 1 do
    C_i ← Probability(Path)[Z_i,i]
  Z_i ← C_i
  End For
Produce (Time,Z)

3.2.2 Prediction

Media predictor uses Viterbi algorithm [36], which calculates hidden media storage. Initially, the

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

algorithm generates optimal state sequence and declares hidden states with observations. In the last, it calculates state sequence S_{states}= {S_1,S_2,… S_n}as,

$$S_{opt} = argmax_s P(Sstates; O; \lambda) \quad (15)$$

Where S_{opt} is optimal state sequence. The algorithm returns optimal path at step *t* over end state *N*. Moreover, it returns an increase *S* over *t+1* and maxima likelihood at *t+2*. The optimal path gets updated at each step and returns hidden media storage in the resultset.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate our proposed approach 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 ecosystem consists of Intel Xeon processor with 8 CPUs, 32GB memory and storage devices i.e. 1TB Hard disk drive and 128GB Samsung SSD. In addition to that, we use Intel core i5 with 4 Core, 16GB memory and storage devices i.e. 1TB Hard disk drive and 128 GB Samsung SSD. We install 5 virtual machines having virtualbox 5.0.16 as seen 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 experimental dataset consists of: (i) 250 random SSD wordcount data blocks of 64MB (40GB size), (ii) 250 random DISK wordcount data blocks (40GB size) and (iii) 250 random RAM wordcount data blocks (40GB size).

4.3 Experimental Results

The evaluation and simulations performed for evaluating proposed approach are: (i) Storage-tier summary collector, (ii) Media prediction analysis and (iii) Processing Latency optimization.

4.3.1 Storage-tier summary collector

The collector fetches event traces of computing capacity, storage-tier I/O, Accessibility Path timestamp, Datablock receiver timestamp and Volume Space statistics over container. The message length varies between 0.5 ≤ size ≥ 5 KB and consumes a resource between 0.2 ≤ Bandwidth ≥ 500 KB/s. The summary container stores 2.7 GB of log information over 120GB data blocks as observed from Figure-7.

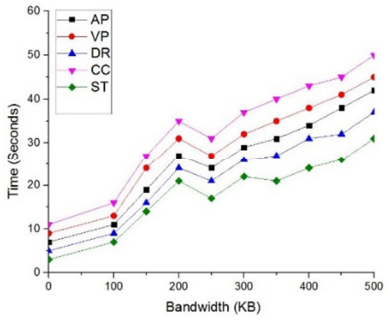

Figure 7: Storage-tier Summary Container Message Collector

4.3.2 Media prediction analysis

After generating container messages, we perform prediction simulations over three '250' random data blocks. In the first hour of simulation, we observe that predictor detects pattern of '109' SSD data blocks, '78' DISK data blocks and '63' RAM data blocks. In the second hour of simulation, we use '500' random data blocks and analyze that predictor observes pattern of '211' SSD data blocks, '192' DISK data blocks and '97' RAM data blocks. In the third hour of simulation, we evaluate '750' random data blocks and evaluate that predictor observes pattern of '322' SSD data block, '219' DISK data blocks and '109' RAM data blocks as observed from Figure-8.

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

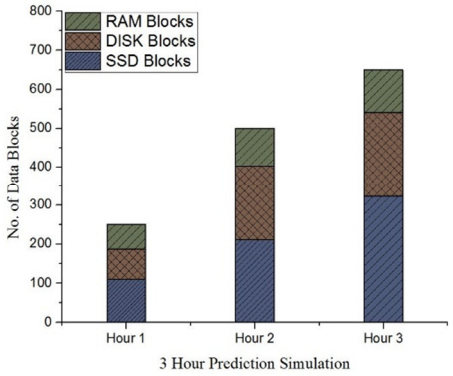

Figure 8: Storage-tier media block job prediction

4.3.3 Processing Latency Optimization

The media predictor depicts locality-aware nearby Datanode statistics with available functional media. The predictor lists processing timestamp, accessibility timestamp and completion timestamp of data blocks over respective storage media. As a result, we calculate locality-aware path and observes that proposed processing, node and storage-tier approaches are 39.1%, 54.7% and 22.9% efficient than default data block processing. This reduces storage-tier latency, node latency and overall processing latency as observed from Figure-9

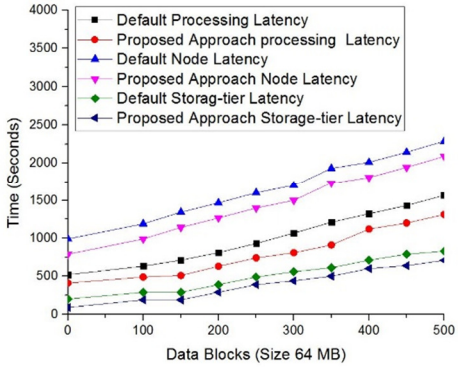

Figure 9: Latency optimization over HDFS Cluster

5. CONCLUSION

This paper proposes a predictive analysis, which collects storage-tier summary data into a container and predicts storage-tier, node and cluster processing timestamp than default approach. The proposed approach effectively stores storage-tier information messages with low message overhead. The predictive analysis depicts a clear significant difference of data block processing improvement than default approach.

In future, we would focus to work over inter-media contention issue over Hadoop cluster.

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-

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

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 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] Jin, Hui, et al. "Adapt: Availability-aware mapreduce data placement for non-dedicated distributed computing." *Distributed Computing Systems (ICDCS), 2012 IEEE 32nd International Conference on*. IEEE, 2012.

[26] Guo, Zhenhua, Geoffrey Fox, and Mo Zhou. "Investigation of data locality in mapreduce." *Cluster, Cloud and Grid Computing (CCGrid), 2012 12th IEEE/ACM International Symposium on*. IEEE, 2012.

[27] Maheshwari, Nitesh, Radheshyam Nanduri, and Vasudeva Varma. "Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework." *Future Generation Computer Systems* 28.1 (2012): 119-127.

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

[29] A. Spivak and D. Nasonov, "Data Preloading and Data Placement for MapReduce Performance Improving", *Procedia Computer Science*, vol. 101, pp. 379-387, 2016.

[30] Palanisamy, Balaji, et al. "Purlieus: locality-aware resource allocation for MapReduce in a cloud." *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011.

[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] J. S. Yedidia, "Message-passing algorithms for inference and optimization," Journal of

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

Statistical Physics, vol. 145, no. 4, pp. 860-890, 2011

[33] M. Khosla, "Message Passing Algorithms," PHD thesis, 9 , 2009

[34] Z. Ghahramani, "An introduction to hidden Markov models and Bayesian networks," International Journal of Pattern Recognition and Artificial Intelligence, vol. 15, no. 1, pp. 9-42, 2001.

[35] Ajit Singh, EM Algorithm, 2005.

[36] G. D. Forney, "The viterbi algorithm," in Proc. of the IEEE, vol. 61, no. 3, pp. 268-278, 1973