Journal of Theoretical and Applied Information Technology
31st May 2017. Vol.95. No 10
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

HSLA: HETEROGENEOUS STORAGE-TIER LOG ANALYZER 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

Hadoop ecosystem processes extremely large datasets in a parallel computing environment. The Hadoop Distributed File System (HDFS) manages operational aspects of processed, unprocessed and log archives. Recently, HDFS has adopted heterogeneous environment, that enables file system to cope with storage-tier data processing. This increases the functional utilization of storage devices and distributes node capacity to storage-tier unevenly. Thus, a job having high priority is affected with delay latency and storage devices i.e. Disk, SSD and RAM consumes individual time overhead to release a non-priority job data. To analyze the complexity of storage-tier, we present Heterogeneous Storage-tier Log Analyzer (HSLA) strategy, that collects control and data flow events to a central repository and performs an analysis over log datasets. The analytics metrics consists of pre-emptive measures observed through events traces. The experimental results depict that, HSLA presents a broad aspect of storage-tier contingency problem and proposes a node computing capacity share strategy to balance functional processing of HDFS blocks over storage-tier.

Keywords: *Hadoop, HDFS, Log analysis, Storage-tier, heterogeneous node.*

1. INTRODUCTION

The concept of Big Data has resolved the complexity of processing extremely large datasets in a distributed environment [1]. Among many data processing systems, we find Apache Hadoop [2], MapR [3] and Cloudera [4] to be the most popular and stable ecosystems. Hadoop is an open-source ecosystem, which processes huge datasets using generic four components i.e. Hadoop commons, YARN, HDFS and MapReduce. YARN is considered as the central controller of Hadoop, which allocates resources and schedules tasks over the cluster [5]. Hadoop commons represent a functional library, which supports environment processing. MapReduce is a programming module that processes large-scale datasets in distributed environment [7]. The Hadoop Distributed File System (HDFS) is a core backbone of ecosystem, which manages processed, unprocessed and logs over the cluster [6].

The architecture of HDFS consists of three main components i.e. client, Namenode and Datanode. The client submits a request to perform a task over Namenode. The Namenode allocates data blocks for the task and provides access over a Datanode. The Datanode processes the task through local resources and generates an output over storage-tier of HDFS [8] [9] as seen from Figure-1.

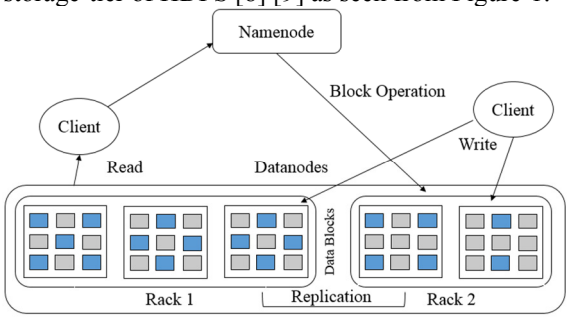

Figure 1: HDFS Architecture

Recently, Hadoop is equipped with heterogeneous storage-tier functionality, which provides a facility to store datasets over store media i.e. DISK, SSD and RAM [10] [11]. To observe a detailed view, storage-tier communicates through control and data flow event traces [12] [13]. Therefore, when data block placement occurs,

© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

control flow event trace is received over Datanode having data flow event trace [14] [16] [17]. The log traces are stored over repository and a reference index is kept into log container [15] as seen from Figure-2.

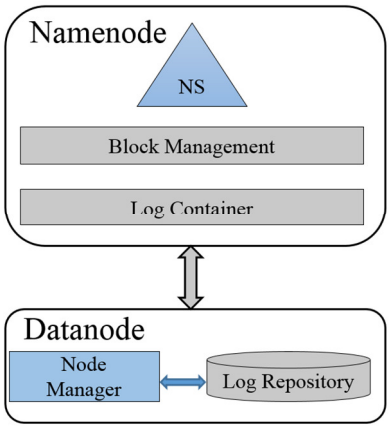

Figure 2: HDFS storage-tier Log Repository

By default, a Datanode places data blocks to a single storage device at a time [18]. Therefore, when a low-priority job placement occurs, it consumes node capacity till completion of the task. This affects the concept of parallel processing in cluster and high-priority jobs observes storage-tier delay latency and individual device produces time overhead till release of low-priority data job [19] [20] [21]. Thus, a cluster places data blocks below to its installed configurations.

To streamline the discussed problem, we present Heterogeneous Storage-tier Log Analyzer (HSLA), which addresses priority job block placement. The proposed approach collects control and data flow event traces and analyze device-to-data placement priority. Moreover, it recommends processing optimal data blocks to suitable storage-tier device to reduce individual time overhead and balance parallel computing among storage-tier devices.

The main contributions of the proposed scheme are:

- An efficient control and data flow event trace container.
- An effective storage-tier analyzer to propose *n* data blocks to *m* device.

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

2. RELATED WORK

Many researchers have presented log analyzing architectures for Hadoop cluster. The significant contributions can be divided into three categories i.e. Network log, Cluster log and node log.

The network log management strategies include the research work of ELT and IOSIG+. The Efficient Log-based Troubleshooting (ELT) [25] strategy collects network log messages and perform troubleshooting for administrators. It is useful for network parameter analysis i.e. network congestion troubleshooting and fault diagnoses of homogeneous network nodes only. The IOSIG+ [26] is suitable for Hadoop network analysis having sub-projects i.e. Mahout and Hive. The prime purpose of this analyzer is to collect and analyze I/O traces between base Hadoop common and sub-project. The functional perspective lack heterogeneous node factors.

The cluster log management methods include Mochi and Diagnosing Heterogeneous Hadoop cluster. Mochi [27] is a cluster log analyzer that logs behavior in space, time and volume. It collects control and data flow events across the cluster to analyze performance and debugging issues. It is a suitable tool to analyze homogeneous storage activities across the cluster. The Diagnosing Heterogeneous Hadoop Clusters [28] approach focuses over heterogeneity-aware log analysis. It consists of diagnoses and fault analysis over CPU contention and DISK I/O contention. Therefore, when we consider heterogeneous storage-tier portion over DISK I/O, it is equipped to analyze only single media type i.e. DISK.

The node log management strategies include SALSA, Ganesha and Visual Log-based tracing. The SALSA [29] [22] logs control and data flow event traces of a node and analyses failure diagnosis of performance through workload processing. It is a homogeneous strategy that work over homogeneity-aware storage-tier and consider CPU contention only. The Ganesha [30] [23] is a black-box diagnosis strategy and focuses over fault analysis of Map task execution over a Datanode. It works over homogeneous storage-tier and consider OS-level parameters for logging event traces. The Visual Log-based tracing [37] [24] analyzer collects core Map and Reduce event traces from a Datanode and perform analysis over MapReduce logs. The in-depth MapReduce log analysis includes client, Namenode and Datanode behavior analysis for troubleshooting purpose.

Keeping in view, the above discussed contributions lack heterogeneous storage-tier log analysis. Therefore, we propose HSLA that uses

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

control and data flow event traces to log storage-tier logs. Moreover, it reduces individual media time overhead through resolving the storage media contention problem.

3. HETEROGENEOUS STORAGE-TIER LOG ANALYZER (HSLA)

The proposed analyzer works in three phases i.e. (i) Identification of Control and Data Flow events, (ii) Event Container and (iii) Performance analysis.

At first, the analyzer identifies control and data flow events in HDFS storage-tier logs. The marked event traces are collected into Event container through belief propagation method and storage-tier log analysis is carried out through performance analysis of extracted metrics and dataset as seen from Figure-3

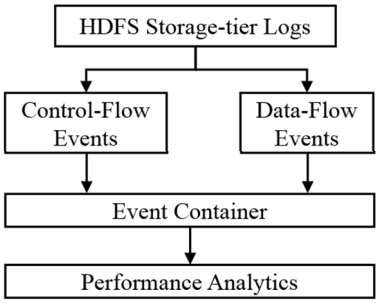

Figure 3: Heterogeneous storage-tier Log Analyzer Workflow

3.1 Identification of Control and Data Flow events

The HDFS block management receive control flow event trace of storage-tier processing. The message includes StorageTID, DeviceID, BlockID and Timestamp to be processed over the cluster. Based on the storage-tier log, HSLA derives functional communication between HDFS and active storage devices of the Datanode i.e. DISK, RAM and SSD. Moreover, the control flow events over HDFS can be collected as,

$$C_{fs} = \sum HDFS\ Control\ events \quad (1)$$

The Datanode control-flow initiates *BlockPlacement* method to respective storage devices and the collection of control flow events over a Datanode can be observed as,

$$C_{Dn} = \sum Datanode\ Control\ events \quad (2)$$

The collection control events can be observed as,

$$C_E = (C_{fs}, C_{Dn}) \quad (3)$$

Where C_E represents collection of control-flow event traces over storage-tier HDFS.

A token represents the start and end of an activity. There are three type of activities i.e. StorageType, StorageDevice and BlockPlacement. Initially, control-flow activate StorageTID, which generate a table having storage types over the cluster. Secondly, DeviceID is linked to the storage type and confirms the availability of the storage medium. Finally, BlockPID places data block over given DeviceID with Timestamp. In this way, HSLA calculates control-flow consumption time through storage-tier processing log activities as seen from Table-1.

Table 1: Tokens in HDFS storage-tier for identifying start and end of states

Activity	Start Token	End Token
StorageType	Launch (StorageTID)	Task (StorageTID) completed
StorageDevice	Launch (DeviceID)	Task (DeviceID) completed
BlockPlacement	Launch (BlockPID)	Task (BlockPID) completed

The HDFS block management calculates data-flow event traces through transferring data blocks from HDFS to storage device of Datanode. The transfer of data blocks can be observed through *ReadBlock* and *WriteBlock* states. Therefore, when a control-flow event completes the cycle, read/write block event trace is generated. In this way, a Datanode comes to know that a data block placement is completed as observed from Table-2.

Table 2: Tokens in Datanode for identifying end of states

Activity	End Token
ReadBlock	Process BlockPID to storage device
WriteBlock	Process BlockPID from storage device

The collection data-flow events can be observed as,

$$D_{Dn} = \sum Data - flow\ events \quad (4)$$

Where D_{Dn} represents collection of data-flow event traces over storage-tier HDFS.

3.2 Event Container

HSLA collects control and data flow event traces through Belief Propagation method [32], which stores log messages to destination component. To perform inference, we use Message Propagation Model [33], which states that, a log message *m* of a variable component *i* having value $\varkappa_i$ with a belief b_i ($\varkappa_i$) is stored over component *i*

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

having likeliness of random variable X_i where $\varkappa_i \in X_i$ by,

$$message\ m_{a \to i}(\varkappa_i) \quad (5)$$

The control-flow event trace log message can be passed towards event container as,

$$message\ m_{C_i \to EC_i}(\varkappa_{EC_i}) \quad (6)$$

Similarly, the data-flow event trace log message can be passed towards event container as,

$$message\ m_{D_i \to EC_i}(\varkappa_{EC_i}) \quad (7)$$

The event container receives component messages and belief of *EC* component can be calculated as,

$$b_i(\varkappa_{EC_i}) \propto \prod_{(C_i,D_i) \in N(EC_i)} m_{(C_i,D_i) \to EC_i}(\varkappa_{EC_i}) \quad (8)$$

To simplify equation (7), we use contact Z and the belief can be obtained as,

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

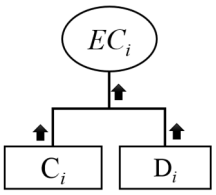

Figure 4: Belief of component EC

After applying factor F_A to EC container, we receive a close-form solution as,

$$m_{(C_i,D_i) \to L}(\varkappa_L) = \sum f_A(X_A) \prod_{(C_i,D_i) \in N(L)} m_{(C_i,D_i) \to L}(\varkappa_L) \quad (10)$$

Where $m_{(C_i,D_i) \to L}(\varkappa_L)$ represents collection of control and data flow event traces in Event Container to factor F_A at respective control and data flow events.

3.3 EC Analyzer

The EC analyzer reviews container datasets and proposes non-preemptive data block processing over heterogeneous storage-tier. It is categorized into three phases i.e. (i) Control and Data filter, (ii) Node capacity estimator and (iii) Job priority over analyzer.

The Control and Data filter extracts Timestamp of selecting StorageTID, assigning DeviceID and processing BlockPID statistics. Moreover, the filter extracts following concurrent time-series events:

a) Node capacity at the time of DISK job processing (50% Core).

b) Node capacity at the time of SSD job processing (30% Core).

c) Node capacity at the time of RAM job processing (20% Core).

Keeping in view, the storage devices generates *ReadBlock* and *WriteBlock* timestamps according to assigned core capacity. Therefore, we observe a shared block job processing over a Datanode as,

$$Shared_P = \left(SSD\frac{3}{10}, DISK\frac{1}{2}, RAM\frac{1}{5}\right) \quad (11)$$

The $Shared_P$ node capacity is a metrics parameter to assign data block job processing over a node. Therefore, it reduces the frequency of high-priority and low-priority data block processing and balances the cluster performance.

4. EXPERIMENTAL EVALUATION

In this section, we evaluate HSLA scheme over Hadoop configuration as seen from Table-3.

Table 3: 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 cluster consists of Intel Xeon with 8 CPUs, 32GB memory and storage devices i.e. 1TB Hard disk drive and 128GB Samsung SSD. Moreover, 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 have created 5 virtualbox 5.0.16 virtual machines on discussed cluster configurations as seen from Table- 4.

Table 4: Hadoop Cluster Virtual Machines Configuration.

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

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

4.2 Experimental Dataset

The dataset used to process experimental work includes: (i) 10 SSD wordcount data blocks of 64MB (10GB size), (ii) 10 DISK wordcount data blocks (10GB size), (iii) 10 RAM wordcount data blocks (10GB size) [31].

4.3 Experimental Results

The experiments performed to evaluate HSLA strategy are: (i) Control and Data flow events, (ii) Node Capacity percentile and (iii) Processing Percentile.

4.3.1 Control and Data flow events

As we know that control and data flow event traces are inter-connected to each other. At first, HSLA trigger '7' StorageTID events to return DeviceID INFO message. We observe that Datanode respond back with INFO event traces having availability of storage devices. Moreover, the approach extends triggering *'7+n'* DeviceID events to obtain suitable volume space for block job processing. Finally, data block placement occurs over *'7+n+m'* BlockPID events traces.

After receiving control-flow successfully, Datanode perform block placement operation and returns data event traces i.e. *ReadBlock* and *WriteBlock*. Keeping in view, that event *ReadBlock* returns data block read operation INFO message and timestamp, whereas, *WriteBlock* generates write operation INFO message to another Datanode or cluster. Therefore, we observe that a timestamp of '55' and '86' seconds of return time as seen from Figure-6.

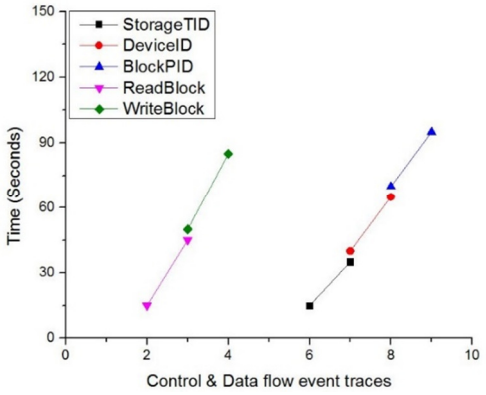

Figure 6: Control and Data flow event traces over Datanode

4.3.2 Node Capacity utilization

The proposed scheme HSLA transforms data block placement to parallel processing. After filter event traces through *EC*, we obtain data block processing timestamp and divide node capacity to $Shared_P$ parameters for parallel block job processing. We observe that, $Shared_P$ node configuration permits *'3+n'* block job processing to respective storage media. Moreover, the device capacity strengthens I/O operations over $Shared_P$ and balances block job processing as seen from Figure-7.

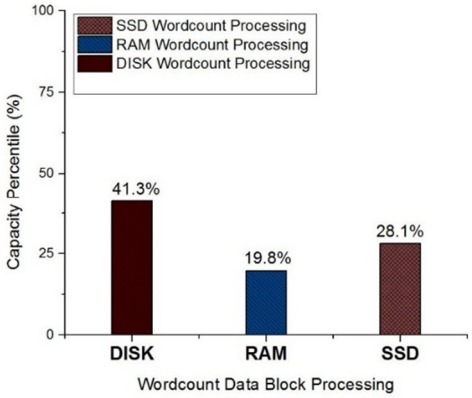

Figure 7: $Shared_P$ based Data Block Processing

4.3.3 Storage-tier Block Job Performance

The presented approach HSLA reduces the factor of high and low priority jobs by sharing computing resource between storage-tier devices. Therefore, we evaluated that 30 storage-tier block jobs having $Shared_P$ configuration returns consumption of 43.7% of high priority block jobs with 39.3% of low priority block jobs over a Datanode as seen from Figure-8.

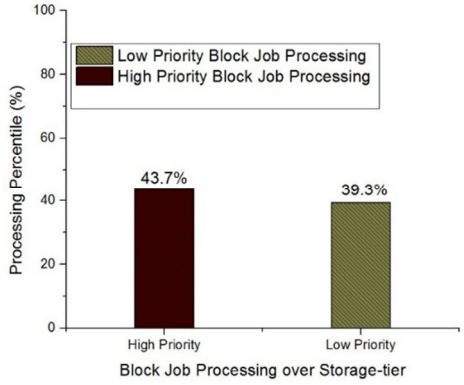

Figure 8: Storage-tier Block Job Performance

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

5. CONCLUSION

This paper proposes a novel Heterogeneous Storage-tier Log Analyzer (HSLA), which addresses parallel block job placement issue. The proposed approach identifies control and data flow event traces through tokens of start and end processing. Moreover, Event Container (EC) significantly collects the token identifiers and passes through EC Analyzer. The analyzer builds a hierarchy of control and data flow events and assigns node capacity for block job parallel processing. Finally, the job priority issue is resolved through sharing node capacity for parallel heterogeneous storage-tier data block processing.
In the future, we focus to work over multihoming event traces 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-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.

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

[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] Kc, Kamal, and Xiaohui Gu. "ELT: Efficient log-based troubleshooting system for cloud computing infrastructures." *Reliable Distributed Systems (SRDS), 2011 30th IEEE Symposium on*. IEEE, 2011.

[26] Feng, Bo, et al. "IOSIG+: on the Role of I/O Tracing and Analysis for Hadoop Systems." *Cluster Computing (CLUSTER), 2015 IEEE International Conference on*. IEEE, 2015.

[27] Tan, Jiaqi, et al. "Mochi: Visual Log-Analysis Based Tools for Debugging Hadoop." *HotCloud*. 2009.

[28] Gupta, Shekhar, et al. "Diagnosing heterogeneous hadoop clusters." *Workshop on Principles of Diagnosis*. 2012.

[29] Tan, Jiaqi, et al. "SALSA: Analyzing Logs as StAte Machines." *WASL* 8 (2008): 6-6.

[30] Pan, Xinghao, et al. "Ganesha: blackBox diagnosis of MapReduce systems." *ACM SIGMETRICS Performance Evaluation Review* 37.3 (2010): 8-13.

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

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

[37] Tan, Jiaqi, et al. "Visual, log-based causal tracing for performance debugging of mapreduce systems." *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*. IEEE, 2010.