Journal of Information Security, 2016, 7, 103-111
Published Online April 2016 in SciRes. http://www.scirp.org/journal/jis
http://dx.doi.org/10.4236/jis.2016.73007

Trusted Heartbeat Framework for Cloud Computing

Dipen Contractor[1], Dhiren Patel[1], Shreya Patel[2]

[1]Department of Computer Engineering, NIT, Surat, India
[2]Department of Computer Engineering, CKPCET, Surat, India
Email: contractor.dipen@gmail.com, dhiren29p@gmail.com, shreya218@gmail.com

Received 26 November 2015; accepted 29 March 2016; published 1 April 2016

Abstract

In cloud computing environment, as the infrastructure not owned by users, it is desirable that its security and integrity must be protected and verified time to time. In Hadoop based scalable computing setup, malfunctioning nodes generate wrong output during the run time. To detect such nodes, we create collaborative network between worker node (*i.e.* data node of Hadoop) and Master node (*i.e.* name node of Hadoop) with the help of trusted heartbeat framework (THF). We propose procedures to register node and to alter status of node based on reputation provided by other co-worker nodes.

Keywords

Trusted Heartbeat, Cloud Computing, Hadoop Framework, Reputation, Attestation

1. Introduction

Outsourcing computation to cloud can reduce IT expenditure spent by companies. Still, most of them are not willing to do so, due to security concerns with cloud computing environment and services. As per survey [1], it is found that despite of huge benefits, fear is still there about security threats like loss of control of data and integrity of systems. Computing nodes (virtual machines) can be tampered with or ill configured to produce wrong results. E.g. Assigned Hadoop task (related to financial data consolidations) may generate incorrect result due to few malfunctioning nodes [2]. Due the large size of data and its processing, the error is very hard to identify in collective results, and it may result in huge loss.

Malfunctioning Nodes and Infrastructure Attacks

In a public cloud infrastructure, malfunctioning nodes may infringe security requirements specified by service consumer. They may produce malicious outputs, which may violate the privacy and integrity ofcomputation. This may result in disclosure of users' confidential data, and profile users' behaviors (and preferences) for privacy analysis. Moreover, software flaws, bugs and mis-configurations can lead to incorrect results or unintended information leakage.

Malicious or tempered nodes may eavesdrop the communication between other nodes, in order to disclose confidential data, enforce malicious privacy profiling [3], launch replay attacks [3], and Man-In-the-Middle attacks [2] [4] in the cloud system. They may also impersonate the Master to steal other node's data, or *vice versa*. Moreover, malicious node can launch Denial of Service attacks [5]. Growing number of vulnerabilities uncovered in cloud platform has prompted the move towards implementing trust based solutions incorporated with hardware support. The Trusted Computing's (TC) [6] initiative and adoption of trusted platform module (TPM) [7] has been gaining attestion from industry as well as acadamic. Hardware manufacturer is also participating to accelerate the adoption of TC across varios platform.

We consider a cloud system, which takes a user task, distributes among computing nodes, and gathers its output as shown in **Figure 1**. We propose a framework for integrity verification of cloud. We use three procedure viz: for registration, for verification and for detection of virtual machine. TPM based node registration process will initially establish trust. Attested heartbeat procedure periodically verifies the trustworthiness of every node in the system. Tampered or misconfigured nodes can be identified quickly by reputation based decision procedure.

Rest of the paper is organized as follows: In Section 2, we discuss background and related work on Heartbeats and TPM. In section 3, we propose the Trusted Heartbeat infrastructure. Section 4 shows usage model of proposed framework with conclusion and references at the end.

2. Background and Related Work

2.1. Hadoop

Apache Hadoop [8] is framework that facilitates the data intensive distributed processing of massive data sets across clusters machines. It supports extension of processes from a single to thousands of machines. Designed with a fundamental assumption that hardware failure is common, making it the software's responsibility to identify and handle failures at the application layer. It replicates data across multiple nodes with rapid data transfer facility. Hadoop implementation essentially consists of two major components: (i) Hadoop Distributed File System (HDFS) [9]: A file system that manages all the nodes in a cluster for data storage, and (ii) MapReduce [10]: The framework that allocate work to nodes in a cluster. Hadoop Cluster can be designed in various ways. One of which includes a single master and multiple worker nodes. The master node consists of a JobTracker, TaskTracker, NameNode and DataNode. A worker node acts as both a DataNode and TaskTracker, it depends on avalability of physical or virtual resources (**Figure 2**).

2.2. Heartbeat in Hadoop Environment

Heartbeat [11] is a communication mechanism that provides a efficient; yet simple way for a Hadoop system to monitor performance and make that information available to external observers. Applications can use heartbeat

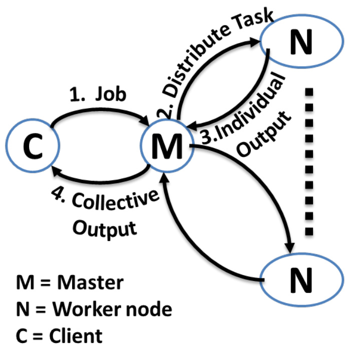

Figure 1. Cloud computing scenario.

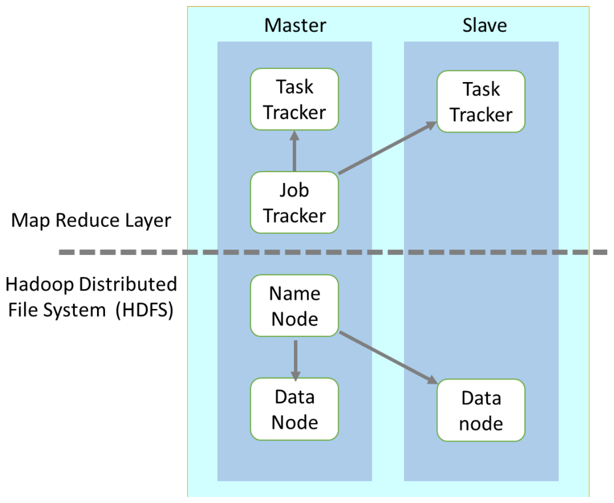

Figure 2. Hadoop cluster adopted from [10].

information to automatically add or subtract resources from their pool. HDFS replicates file blocks for fault tolerance. An application can specify the number of replicas of a file at the time it is created. The NameNode makes all decisions concerning block replication. Each DataNode sends heartbeat messages timely to its NameNode, so the later can identify loss of connectivity if it stops receiving these messages. The NameNode marks such node as dead DataNode (not responding to heartbeats) and desists from sending requests to it. Data stored on such node is no longer available to a client (**Figure 3**).

2.3. Trusted Platform Module (TPM)

The trusted computing group consortium has developed specifications for the trusted platform module. The TPM is a special purpose microcontroller on a motherboard. By incorporating a physical facility for secure generation and storage of cryptographic keys, the TPM becomes the core supporter for creating an interoperable "trusted computing" environment. These capabilities that every TPM provides include hashing by SHA-1 algorith, random number generation, asymmetric key generation as well as encryption and decryption by RSA algorithm. Following in **Table 1**; is the list of different types of keys can be created with TPM with their properties.

Integrity verification of the software components to support mitigation of security concerns related to cloud computing infrastructure. Though, it does not actually provide absolute assurance, trusted computing improves the complexity for attackers by operating at hardware level. With a correct implementation, an attacker would need physical access to the hardware in order to subvert the TPM [13]. In our proposed work, we use TPM to prevent Man-In-The-Middle attack and verify the integrity of Virtual machine via attestation.

2.4. Related Work

There have been many attempts to enhance the fault tolerance and trust based mechanisms to preserve integrity of cloud system in open distributed environment [14]. For sensitive data in open distributed systems, Airavat [15] is developed. It incorporates mandatory access control to detect privacy violation. Verification-based Integrity Assurance Framework [16] is based on the idea of replication and quiz related methods. It can detect malicious and normal task trackers in Hadoop system with the help of predefined set of questionnaires. Authors in article [17], proposed algorithm named Longest Approximate Time to End (LATE). LATE finds the slow tasks in a homogeneous environment. LATE first estimates the remaining time for each tasks, then assigns the speculative tasks for those with the longest remaining time to end and maintains integrity of the system. Terra [18] provides an attestation ability that allows a remote party to reliably detect whether the host is running a platform that the remote party trusts. As elaborated by Bercher *et al*. [19], for encrypted communication between all the nodes in the HDFS system, a key must be securely exchanged in advance. However, there are issues with how the key is shared. As seen in [20]; key exchange is done frequently by heartbeat messages and attacker can pretend to be a data node and can many chunks of data.

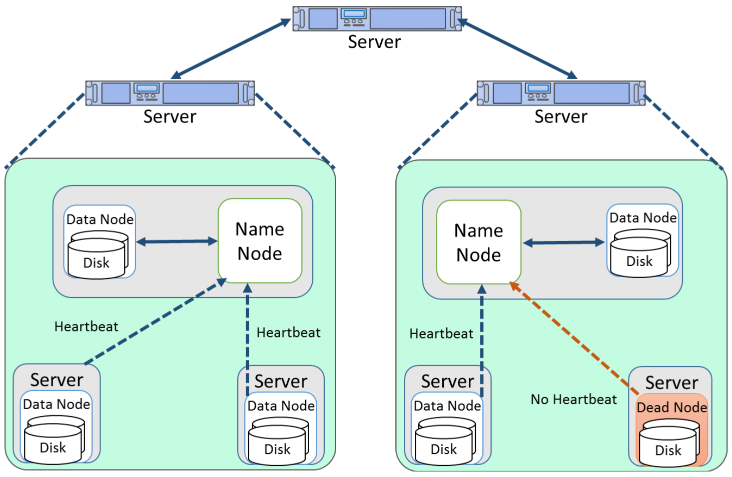

Figure 3. Heartbeat in Hadoop environment adopted from [12].

Table 1. TPM Key types with their pupose [7].

Key Name	Purpose
Endorsement Key (EK)	A key-pair based on RSA algorithm; imposed by TPM manufacturer to identify uniquely TPM.
Storage Root Key (SRK)	A non-transferable key generated by the platform owner to serve as the root key in the hierarchy of keys associated with the TPM.
Attestation Identity Key (AIK)	Used for attestation and identification of a TPM (*i.e.* activated mode). Trusted third party can create identity certificate by signing public key part of AIK.
Signing Key	Used by the system to sign messages.
Storage Key	Used to encrypt and decrypt other keys. (using RSA)
Identity Key	Used for operations that requires TPM identity.
Binding Key	Used for Unbind operations to decrypt a data.

We propose a scheme to determine whether a particular VM is trustworthy or not. Only attested and trusted VMs can get the tasks and collaborate in network. Negative reputation is assigned if node does not generate output (or produce malicious output).

3. Trusted Heartbeat Framework

In this framework, we assume TPM communication cannot temper, and storage is not exposed. The main intention of TPM is to repel most of the attacks on the software, we presume that trusted platform can assess each and every software module loaded on platform in terms of hash code [7]. In addition, we assume Master node works as a trusted party which performs attestations as suggested by TC [21]. For better understanding of our framework, we denote job tracker as a *master node* and task tracker as *simple node*. Proposed framework is as shown in **Figure 4**. The Task scheduler present at every node executes tasks assign to them. Trust collector is attached to each node to manage assessments and support the attestation service. In master node, the task scheduler deploys jobs to nodes and collects their outcomes. Task manager stores node information and their assign task information. In addition, trust and reputation collector manages the trust information of nodes, and Trust Verifier performs attestations to them. The collected security properties (Endorsement Key and Attestation Identity Key) are stored in the Trust storage.

Trust manger binds evidence generated with accordance to TC's notation as trusted data. Moreover, users can get such information to assess the security properties of the worker node at any time, for the entire processing cycle. The Trust & reputation collector collects such properties of nodes and stores them with corresponding values. These values are kept in the trust storage for future score calculation. Following are the three main procedures for our proposed system.

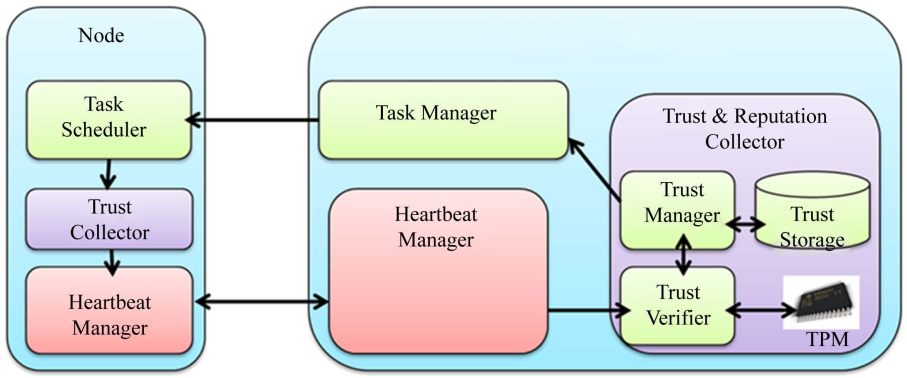

Figure 4. Architecture of trusted heartbeat framework.

(a) Initial Node Registration

Initially, when a data node joins a network, node registration takes place. It identifies a genuineness of TPM and exchanges keys for sealing and binding operations. The genuineness of TPM is identified by its public EK key.

Every time a worker node initiate connection request to the master node, an initial attestation procedure will be executed by master node. Verifier has collected all the properties of each node whose information is stored at the storage. Therefore, only registered node with allowed properties will be included to the list of the task Manager (for completing tasks). TC credentials and public session keys are stored at trust storage.

In Trusted Heartbeat framework, every node (N) is identified with its corresponding and unique AIK, and the Master (M) facilitates as the Privacy-CA defined by TC infrastructure [22], for registering and identifying all these AIKs. Whenever a new node is included to the Hadoop based cloud system, it is first get registered at the Master and assigned with AIK key credentials. **Figure 5** shows steps for node registration. As suggested earlier, only registered node can communicate with master and can get tasks with genuine TC credentials. Node registration procedure is more elaborated in **Figure 6**. In our Trusted Heartbeat framework implementation, the Trust & Reputation Collector is added to Master node and Trust collector to every node. Their exchanged messages are incorporated into the heartbeat protocol via Heartbeat manager. To simplify our protocol, we assign manually AIK credentials to node.

Time to time collector and trust storage updates the nonce information, and initiate the attestation procedure by invoking the TPM Quote from TPM instruction with the fresh nonce. As shown in **Figure 9(a)**, latest generated nonce and reputation is then added to the request and sent to the Master. The verifier collects and maintains the public credentials of individual nodes. When a request in heartbeat message; with genuine AIK credentials is received, verifier will first perform verification followed by registration of that node. The properties indicated from the worker node's Stored Message Logs (SML) are inspected with security policies which are defined earlier and stored in Trust storage (shown in **Figure 9(c)**), and only the expected worker node can be added in future. As shown in node registration procedure (**Figure 6**), Trust verifier maintains nonce information in a cache (*i.e.* for faster execution) and revise cache value by execution the SHA-1 hash operation to get the new value. Time interval between each received messages and the stored information of the cache together determines the longer time for a heartbeat to be valid (**Figure 7**).

(b) Verification of Heartbeats

The verifier from trust and reputation collector is invoked each time, when a heartbeat message with attestation request reaches to the master node. It examines the nonce value in the cache; received through recent heartbeat (last_nonce) message. If verifier does not find that nonce value, it invalidates the connection request through heartbeat message. Once more when worker node sends heartbeat message with valid new_nonce, it can continues to communicate the master and get the task. The trust verifier can verifies the received signature and quote of PCR values using the TPM_Verify [7] set of instructions of TPM. If the verifier finds any mismatch in hash value, it will put that node to gray list and that node has to again start with node initialization procedure (As shown in **Figure 9**). Difference in PCR values shows variation in node software status, therefore a new assessment requires to be initiated. After successful completion of verification, the new worker node's assessment information is updated for further communication.

Procedure Node Registration

$N(Node)_i \rightarrow M(\text{Master node}) : Certificate(K_{EKi}),\ \{K_{AIKi}\}$

$M\ (\text{Master node}) \rightarrow N(Node)_i{:}\{Certificate(K_{AIKi}),N(Node)ID_i\},Certificate(K_M),\ N_a$

Figure 5. Node registration procedure.

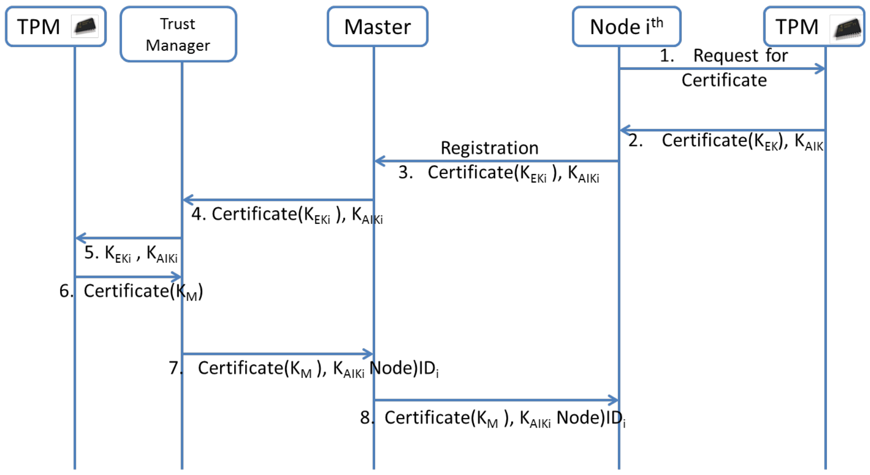
TPM
Trust Manager
Master
Node i^th
TPM
1. Request for Certificate
2. Certificate(K_EK), K_AIK
Registration
3. Certificate(K_EKi), K_AIKi
4. Certificate(K_EKi), K_AIKi
5. K_EKi , K_AIKi
6. Certificate(K_M)
7. Certificate(K_M), K_AIKi Node)ID_i
8. Certificate(K_M), K_AIKi Node)ID_i

Figure 6. Step by step node registration procedure.

```
Procedure for Verification of Heartbeat
Variable definition:
   new_nonce: Represent the received nonce of the node from heartbeat.
   last_nonce: Represent the received nonce of the node from previous heartbeat.

procedure vah
//input: Heartbeat_Received: Symbolize the received heartbeat
//output: Grey list
   new_nonce = Nounce.get(Heartbeat_Received)
   new_nonce ← $nonce_{last}$
   updated_nonce ← (current_time – last_time)
   for i=1 → updated_nonce do
   new_nonce ← compute hash of new_nonce
        insertnew_nonce into nonce_cache
   end for
   if Heartbeat.nonce ∉nonce_cache then
        return false (Heartbeat_Received   is Invalid)
   end if
   if Verify_Sig(nonce, AIK)= false then
        return   false (Heartbeat_Received Invalid)
   end if
   if hash(PCRs)∉node_PCRHash values then perform attestation
        if Attestation is correct then
   insert PCRHash values
   add(time _current, PCRHash) to Master_PCRHash
    return true
        else
            return false (node may be malicious)
   Put that node credentials in gray list.
        end if
   end if
end Procedure
```

Figure 7. Verifying heartbeat procedure.

(c) Reputation based detection

Reputations are gathered with each Heartbeat message received from Master. Calculated reputations, which are lower than a pre-defined threshold, master node, will unregister that node or mark it as a lost one. The threshold value can be computed based on the number of nodes and previously stored information available at trust storage. The Black list is one that contains a list of all such failed nodes. Similarly, Gray list is one that contains a probable list of nodes that have faced some decrements in reputations (**Figure 8**). Every susceptible node comes in Graylist first and then after inactivity it will be in Blacklist (shown in **Figure 9(d)**). Since reputations

```
Procedure for reputation based decision
Variable definition:
From_rep: Symbolize the reputation of the sender of a worker node.
To_rep: Symbolizethe reputation of the malicious node.

procedure rbd
//input: Heartbeat_Received: Represent the received heartbeat
//Output: Heartbeat_Sent: Represent heartbeat to sent
From_rep = Reputation.get(Heartbeat_Received)
If From_rep<Threshold
From_rep = From_rep + 1      //gain reputation
endif
If found heartbeat in Greylist
        for each node in Greylist
              Sent Heartbeat_To with penalty
                      If not received any Heartbeat_From from that node
                              If (To_rep< Minimum Reputation)
                                      Put node in Black list and Inform other nodes in same cluster
                      Lostnode(Heartbeat_Sent)
                      endif
              endif
        end foreach
    endif
end procedure
```

Figure 8. Procedure for reputation based decision.

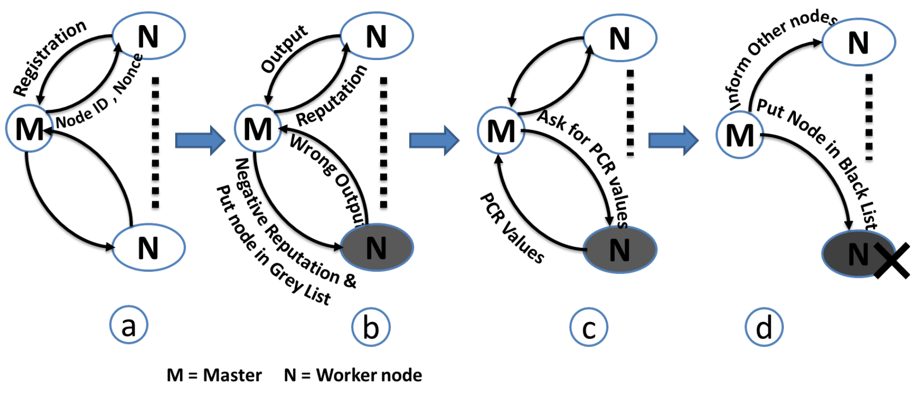

Figure 9. Working of our framework.

are collected in same cluster only, detecting a failed or malicious node is faster compared to collecting all reputations from all the nodes as depicted in Reputation based decision procedure.

A worker node can receive its reputation or penalties through heartbeat messages. Master node increases reputation of a worker node each time when it gets heartbeat messages with hash values. The Trust & reputation based detector has a upper bound for the maximum reputation, After reaching that value, initialization process begins. However, when node comes in graylist then it starts receiving penalties if it does not reply.

Figure 9 shows general step by step working of Trust and Reputation Collector with the procedures. All tasks performed by the Master node are indicated in it.

4. Conclusion

In this paper, we propose Trusted Heartbeat framework; that creates a collaborative network among virtual machines. With remote attestations and heartbeat messages, a Master node can define the exact status (working or malfunctioning) of its nodes. This proposed framework identifies the genuine worker node using trusted computing facilities. Heartbeat interval time is very important parameter in our system. Trust and reputation based detector improve Hadoop like distributed systems in detecting malicious nodes quickly. This framework shows utilization of common messages to establish trust among all the corresponding nodes in distributed environment.

References

[1] (2012) What's Holding Back the Cloud. http://www.intel.in/content/dam/www/public/us/en/documents/reports/whats-holding-back-the-cloud-peer-research-report2.pdf

[2] Khan, S.M. and Hamlen, K.W. (2012) Hatman: Intra-Cloud Trust Management for Hadoop. *CLOUD*'12: *Proceedings of* 5*th International Conference on Cloud Computing*, Honolulu, 24-29 June 2012, 494-501. http://dx.doi.org/10.1109/cloud.2012.64

[3] Feng, J., Chen, Y., Ku, W. and Liu, P. (2010) Analysis of Integrity Vulnerabilities and a Non-Repudiation Protocol for Cloud Data Storage Platforms. *ICPPW*'10*: Proceedings of* 39*th International Conference on Parallel Processing Workshops*, San Diego, 13-16 September 2010, 1-8. http://dx.doi.org/10.1109/icppw.2010.42

[4] Sujitha, G., Varadharajan, M., Rao, Y.V., Sridev, R., Gauthaum, M.K.S., Narayanan, S., Raja, R.S. and Shalinie, S.M. (2013) Improving Security of Parallel Algorithm Using Key Encryption Technique. *Information Technology Journal*, **12**, 2398. http://dx.doi.org/10.3923/itj.2013.2398.2404

[5] Contractor, D. and Patel, D. (2012) Trust Management Framework for Attenuation of Application Layer DDoS Attack in Cloud Computing. *IFIPTM* 2012: 6*th IFIP WG* 11.11 *International Conference on Trust Management*, **374**, 201-208. http://dx.doi.org/10.1007/978-3-642-29852-3_14

[6] Futral, W. and Greene, J. (2013) Introduction to Trust and Intel Trusted Execution Technology. *Intel Trusted Execution Technology for Server Platforms*, 1-14.

[7] TPM Software Stack (TSS) Specification, Version 1.2. http://www.trustedcomputinggroup.org/resources/tcg_software_stack_specification_tss_12_faq

[8] White, T. (2012) Hadoop: The Definitive Guide. O'Reilly.

[9] Dean, J. and Ghemawat, S. (2008) MapReduce: Simplified Data Processing on Large Clusters. *Communications of the ACM*, **51**, 107-113. http://dx.doi.org/10.1145/1327452.1327492

[10] Borthakur, D. (2007) The Hadoop Distributed File System: Architecture and Design. In hadoop.apache.org, 2007. http://hadoop.apache.org/docs/r0.18.0/hdfs_design.pdf

[11] Hoffmann, H., Eastep, J., Santambrogio, M.D., Miller, J.E. and Agarwal, A. (2010) Application Heartbeats for Software Performance and Health. *ACM Sigplan Notices*, **45**, 347-348. http://dx.doi.org/10.1145/1837853.1693507

[12] Hanson, J. (2011) Hadoop Heartbeat. http://www.ibm.com/developerworks/library/wa-introhdfs/

[13] Krautheim, F.J., Phatak, D.S. and Sherman, A.T. (2010) Introducing the Trusted Virtual Environment Module: A New Mechanism for Rooting Trust in Cloud Computing. *TRUST* '10: *Proceedings of* 3*rd International Conference on Trust and Trustworthy Computing*, Berlin, 21-23 June 2010, 211-227. http://dx.doi.org/10.1007/978-3-642-13869-0_14

[14] Scales, D.J., Nelson, M. and Venkitachalam, G. (2010) The Design of a Practical System for Fault-Tolerant Virtual Machines. *ACM SIGOPS Operating Systems Review*, **44**, 30-39. http://dx.doi.org/10.1145/1899928.1899932

[15] Roy, I., Setty, S.T.V., Kilzer, A., Shmatikov, V. and Witchel, E. (2010) Airavat: Security and Privacy for MapReduce. *Proceedings of the* 7*th USENIX Symposium on Networked Systems Design and Implementation* (*NSDI*), **10**, 297-312.

[16] Wang, Y. and Wei, J. (2011) VIAF: Verification-Based Integrity Assurance Framework for MapReduce. *CLOUD*'11: *Proceedings of IEEE International Conference on Cloud Computing*, Washington DC, 4-9 July 2011, 300-307. http://dx.doi.org/10.1109/cloud.2011.33

[17] Zaharia, M., Konwinski, A., Joseph, A.D., Katz, R.H. and Stoica, I. (2008) Improving MapReduce Performance in Heterogeneous Environments. *OSDI*, **8**, 7.

[18] Garfinkel, T., Pfaff, B., Chow, J., Rosenblum, M. and Boneh, D. (2003) Terra: A Virtual Machine-Based Platform for Trusted Computing. *SIGOPS Operating System Review*, **37**, 193-206. http://dx.doi.org/10.1145/1165389.945464

[19] Becherer, A. (2010) Hadoop Security Design Just Add Kerberos? Really. *iSEC PARTNER*, 1-10.

[20] Devi, S. and Kamaraj, K. (2014) Architecture for Hadoop Distributed File Systems. *International Journal of Enhanced Research in Management & Computer Applications*, **3**, 13-19.

[21] Ko, S.Y., Hoque, I., Cho, B. and Gupta, I. (2010) Making Cloud Intermediate Data Fault-Tolerant. *Proceedings of the* 1*st ACM Symposium on Cloud Computing*, Indianapolis, 2010, 181-192. http://dx.doi.org/10.1145/1807128.1807160

[22] Schiffman, J., Moyer, T., Vijayakumar, H., Jaeger, T. and McDaniel, P. (2010) Seeding Clouds with Trust Anchors. *Proceedings of the* 2010 *ACM Workshop on Cloud Computing Security Workshop*, Chicago, 2010, 43-46.