Hatman: Intra-cloud Trust Management for Hadoop

Safwan Mahmud Khan and Kevin W. Hamlen
Department of Computer Science
University of Texas at Dallas
Richardson, Texas, USA
{*safwan,hamlen*}*@utdallas.edu*

***Abstract*—Data and computation integrity and security are major concerns for users of cloud computing facilities. Many production-level clouds optimistically assume that all cloud nodes are equally trustworthy when dispatching jobs; jobs are dispatched based on node load, not reputation. This increases their vulnerability to attack, since compromising even one node suffices to corrupt the integrity of many distributed computations.**

This paper presents and evaluates Hatman: the first full-scale, data-centric, reputation-based trust management system for Hadoop clouds. Hatman dynamically assesses node integrity by comparing job replica outputs for consistency. This yields agreement feedback for a trust manager based on EigenTrust. Low overhead and high scalability is achieved by formulating both consistency-checking and trust management as secure cloud computations; thus, the cloud's distributed computing power is leveraged to strengthen its security. Experiments demonstrate that with feedback from only 100 jobs, Hatman attains over 90% accuracy when 25% of the Hadoop cloud is malicious.

***Keywords*-cloud computing; integrity; Hadoop; reputation-based trust management; security**

I. Introduction

Enormous progress in hardware, networking, middleware, and virtual machine technologies have led to an emergence of new, globally distributed computing platforms that provide computation facilities and storage as services accessible from anywhere via the Internet. At the fore of this movement, cloud computing [1], [2], [3] has been widely heralded as a new, promising platform for delivering information infrastructure and resources as IT services [4], [5]. Customers can access these services in a pay-as-you-go fashion while saving huge capital investment in their own IT infrastructure [6]. Thus, cloud computing is now a pervasive presence of enormous importance to the future of e-commerce.

Data integrity and privacy have emerged as major concerns for prospective users of clouds [7]. A survey by Fujitsu Research Institute reveals that 88% of prospective customers are worried about who has access to their data in the cloud and demand more trustworthiness [8]. Such surveys reveal an urgent need to meaningfully address these concerns for real-world cloud systems.

While cloud data privacy has received a great deal of popular attention in the literature (cf., [9], [10]), computation integrity also remains a significant problem for large-scale, production-level cloud architectures. An attacker who is able to compromise even one cloud node potentially gains the ability to corrupt the outcomes of all computations allocated to that node. Since clouds subdivide and distribute their computations as widely as possible across their nodes to achieve high performance and scalability, this means that a single compromised node can corrupt the integrity of many or even all of the jobs undertaken by the cloud.

As an example, consider a Hadoop MapReduce cloud [11] that performs military intelligence data mining, similar to the one currently under development by the U.S. National Security Agency [12]. An attacker who has compromised just one node in such a cloud can introduce nearly arbitrary error into simple computations, such as word counting or clustering computations, by simply forcing the compromised node to yield false, outlying answers to queries. These answers are summed or averaged into the answers returned by the other nodes, resulting in a final answer (delivered to the user) that is largely dictated by the attacker. Such computational integrity corruption could be applied to frustrate military intelligence data-mining efforts by masking important data correlations or introducing false ones.

For this reason, a large body of work on cloud security focuses on protecting nodes from being compromised in the first place (cf., [13], [14]). Data processing clouds [15], including Hadoop, execute untrusted, user-submitted code on trusted cloud nodes during job processing, and must therefore remain vigilant against malicious mobile code attacks. Virtualization technologies, including trusted hardware, hypervisors, secure OSes, and trusted VMs are the typical means by which such mobile code is secured (e.g., [16], [17]). However, a variety of studies have shown that clouds introduce significant new security challenges that make mobile code security a non-trivial, ongoing battle [18], [19], [13], [20]. For example, the Cloud Security Alliance has identified insecure cloud APIs, malicious insiders, shared technology issues, service hijacking, and unknown risk profiles all as top security threats to clouds [21].

We therefore examine trust management as a second line of defense for cloud computation integrity enforcement. Trust management systems [22] weather (rather than preclude) malicious behavior in distributed systems by tracking

reputations of untrusted agents (e.g., cloud nodes) over time. Agents who frequently exhibit behavior characterized as malicious by more trustworthy agents accrue poor reputations, and therefore become distrusted by the rest of the system. This facilitates detection and rejection of misbehaving agents without the need to modify the underlying hardware, software, or communication protocols of each agent in the system.

To evaluate reputation-based trust management in a realistic cloud environment, we augment a full-scale, production-level data processing cloud—Hadoop MapReduce [3], [11]—with a reputation-based trust management implementation based on EigenTrust [23]. The augmented system replicates Hadoop jobs and sub-jobs across the untrusted cloud nodes, comparing node responses for consistency. Consistencies and inconsistencies constitute feedback in the form of agreements and disagreements between nodes. These form a trust matrix whose eigenvector encodes the global reputations of all nodes in the cloud. The global trust vector is consulted when choosing between differing replica responses, with the most reliable response delivered to the user as the job outcome.

To achieve high scalability and low overhead, we show that job replication, result consistency checking, and trust management can all be formulated as highly parallelized MapReduce computations. Thus, the security offered by the cloud scales with its computational power. Our primary contributions are therefore as follows:

- We implement and evaluate intra-cloud trust management for a real-world cloud architecture—Hadoop.
- Our system adopts a data-centric approach that recognizes job replica disagreements (rather than merely node downtimes or denial of service) as malicious.
- We show how MapReduce-style distributed computing can be leveraged to achieve purely passive, full-time, yet scalable attestation and reputation-tracking in the cloud.

The remainder of the paper proceeds as follows. Section II begins with a system overview. Implementation details and experimental results are described and analyzed in §III and §IV, respectively. Related work is summarized in §V, and §VI concludes with recommendations for future work.

II. System Overview

A. Overview of Hadoop Architecture

The Hadoop Distributed File System (HDFS) [3] is a master/slave architecture designed to run on commodity hardware. Each HDFS cluster has a single *NameNode* master, which manages the file system namespace and regulates access to files by customers. In addition, there are a number of *DataNodes*, usually one per node in the cluster, which manage storage attached to the nodes on which they run. The DataNodes are arranged in racks for replication purposes. Customers communicate with the NameNode, which coordinates the services from the DataNodes.

MapReduce [11] is an increasingly popular distributed programming paradigm used in cloud computing environments. It expedites the processing of large datasets using inexpensive cluster computers. Additional advantages include load balancing and fault tolerance.

In this research work we use Hadoop's MapReduce framework [3]. In Hadoop, the unit of computation is called a *job*. Customers submit jobs to Hadoop's JobTracker component. Each job has two phases: Map and Reduce. The Map phase maps input key-value pairs to a set of intermediate key-value pairs. The Reduce phase reduces the set of intermediate key-value pairs that share a key to a smaller set of key-value pairs traversable by an iterator. When a job is submitted to the JobTracker, Hadoop attempts to place the Map processes near to the input data in the cluster to reduce the communication cost. Each Map process and Reduce process works independently without communication.

B. Hatman Architecture

Hatman (HAdoop Trust MANager) augments Hadoop NameNodes with reputation-based trust management of their slave DataNodes. The trust management system is centralized in the sense that NameNodes maintain a small, trusted store of trust and reputation information; however, all computation is decentralized in that trust matrix computations and user-submitted job code is all dispatched to DataNodes. NameNode computations therefore remain restricted to simple bookkeeping operations related to job dispatch. This keeps the system scalable and maintains high trustworthiness of NameNodes by minimizing their attack surfaces.

Hatman users submit Hadoop jobs J with two additional parameters: (1) a *group size* n and (2) a *replication factor* k. The NameNode distributes user-submitted computation J across kn DataNodes, as illustrated in Fig. 1. Each group of n nodes independently processes job J, with any sub-jobs being redistributed to the nodes of the group that spawned it. Different groups are permitted to have some common members (though this is unlikely when kn is small relative to the size of the cloud), but no two groups are identical. Increasing n therefore yields higher parallelism and increased performance, whereas increasing k yields higher replication and increased security.

Interpretation of the results of these replicated computations proceeds according to Algorithm 1. Line 3 first collects candidate results from each of the k replica groups. The collected results are compared pairwise by lines 7 and 9. Hadoop job results are simply files, which can be partitioned and compared in a highly distributed fashion; therefore, comparison of non-trivial results is implemented by line 9 as a second Hatman job over freshly selected nodes and

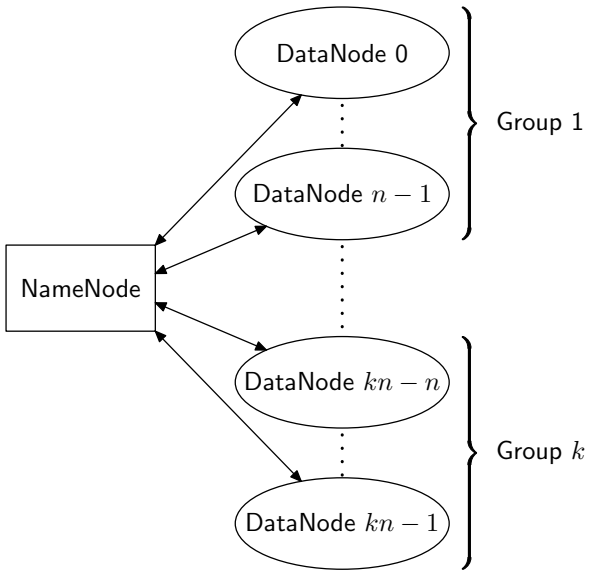

Figure 1. A Hatman job replicated k times and distributed across n data nodes per replica group.

groups. The comparison job recursively leverages the trust management system to ensure high data integrity, resulting in a reliable comparison.

Lines 11–16 tally agreements and disagreements between the groups in a local trust matrix. Lines 18–21 periodically use this to compute a global trust vector for all nodes in the system. Finally, line 22 uses the global trust vector to evaluate the most trustworthy result to return to the user. We next discuss the details of the local trust matrix, the global trust matrix computation, and the evaluation function, respectively.

A large category of Hadoop jobs tend to be stateless and deterministic [24], [25], [26]. Thus, when all nodes are reliable, all k replica groups yield identical results r_g with high probability. However, when some nodes are malicious or unreliable, the NameNode must choose which of the differing responses to deliver to the user, and it must decide how the reputations of members of disagreeing groups are affected by the disagreement.

To make this decision, we appeal to a trust model defined by a local trust matrix $T_{ij} = \alpha_{ij} t_{ij}$, where $t_{ij} \in [0,1]$ measures how much agent i trusts agent j, and $\alpha_{ij} \in [0,1]$ measures agent i's relative confidence in his choice of t_{ij} [27]. The confidence values are relative in the sense that $\sum_{i=1}^{N} \alpha_{ij} = 1$, where N is the total number of agents.

In Hatman, DataNode i trusts DataNode j proportional to the percentage of jobs shared by i and j on which i's group agreed with j's group. That is, $t_{ij} = A_{ij}/C_{ij}$ where C_{ij} is the number of jobs shared by i and j and A_{ij} is the number of those jobs on which their groups' answers agreed. DataNode i's relative confidence is the percentage

Algorithm 1 Hatman job processing

Input: job J, group size n, replication factor k
Output: job result r

1: Choose k unique groups G_g each of size n
2: **for all** groups G_g **do**
3: $r_g \leftarrow HadoopDispatch(G_g, J)$
4: **end for**
5: **for all** pairs (G_g, G_h) with $g \neq h$ **do**
6: **if** r_g and r_h are small **then**
7: $eq \leftarrow (r_g =? r_h)$
8: **else**
9: $eq \leftarrow HatmanDispatch(r_g =? r_h)$
10: **end if**
11: **for all** $(i,j) \in G_g \times G_h$ with $i \neq j$ **do**
12: $C_{ij} \leftarrow C_{ij} + 1$ (and $C_{ji} = C_{ij}$)
13: **if** $eq = true$ **then**
14: $A_{ij} \leftarrow A_{ij} + 1$ (and $A_{ji} = A_{ij}$)
15: **end if**
16: **end for**
17: **end for**
18: **if** time to update trust vector **then**
19: $T \leftarrow HatmanDispatch(tmatrix(A, C))$
20: $t \leftarrow HatmanDispatch(EigenTrust(T))$
21: **end if**
22: $m \leftarrow \arg\max_g \ eval(G_g)$
23: **return** r_m

of assessments of j that have been voiced by i:

$$\alpha_{ij} = \frac{C_{ij}}{\sum_{k=1}^{N} C_{kj}} \tag{1}$$

Multiplying $T_{ij} = \alpha_{ij} t_{ij}$ therefore yields matrix

$$T_{ij} = \frac{A_{ij}}{\sum_{k=1}^{N} C_{kj}} \tag{2}$$

This is the computation performed by $tmatrix(A, C)$ in line 19 of Algorithm 1.

This formula is well-defined whenever j has shared at least one job with another DataNode (making the denominator non-zero). When j has not yet received any shared jobs, we allow all DataNodes to initially trust j (so $t_{ij} = 1$) with uniform confidence ($\alpha_{ij} = 1/N$).

This differs from EigenTrust [23], which initially distrusts new agents because it targets networks with potentially uncontrolled churn. A default distrust of new peers disincentivizes leaving and rejoining such a network to reset reputation. In contrast, clouds typically have more controlled churn; new cloud nodes undergo some form of validation and authorization by organization personnel at installation and cannot leave and rejoin the network arbitrarily. Therefore, Hatman trusts new nodes by default and reduces that trust in response to evidence of compromise.

Following EigenTrust, line 20 computes the left eigenvector of local trust matrix T to obtain a vector t of global reputations for all DataNodes. Once again, this computation is formulated as a distributed Hatman job across a fresh set of nodes and replica groups. The trust vector is recomputed at regular intervals and at idle periods rather than after every job to avoid overburdening the network.

Reputation vector t is used as a basis for evaluating the trustworthiness of each group's response. We employ evaluation function

$$eval(G) = w\frac{|G|}{|S|} + (1-w)\frac{\sum_{i\in G} t_i}{\sum_{i\in S} t_i} \quad (3)$$

where $S = \cup_{j=1}^{k} G_j$ is the complete set of DataNodes involved in the activity, and weight $w \in [0,1]$ defines the relative importance of group size versus group collective reputation in assessing trustworthiness. In §IV we observe highest accuracy with $w = 0.2$, demonstrating that reputation is about 4 times more effective than simple majority voting for identifying integrity violations. The result yielded by the group with the highest evaluation score is the one returned to the user.

C. Activity Types

An *activity* is a tree of sub-jobs whose root is a job J submitted to Algorithm 1. There are three different types of activities that Hatman undertakes: *user-submitted activities*, *bookkeeping activities*, and *police activities*.

User-submitted activities are jobs submitted by cloud customers. These receive highest priority in the system, with parameters n and k chosen by the user (and perhaps entailing higher customer cost in response to demands for greater parallelism and replication).

Bookkeeping activities are result-comparison and trust matrix computation jobs submitted by lines 9, 19, and 20 of Algorithm 1. These inherit the priority of the user-submitted job with which they are associated, and receive high replication factors k to ensure their integrity.

Police activities are dummy jobs (e.g., replayed user-submitted activities or stock jobs) whose sole purpose is to exercise the system. These are undertaken during periods of low load to help trust matrix T converge more quickly. The results of police activities are discarded, providing a safe means to assess reliability of low-reputation nodes without risking delivery of low-integrity results to users.

D. Attacker Model and Assumptions

We assume that attackers can compromise DataNodes but not NameNodes. NameNodes do not execute any user-submitted code, and have a substantially simpler computing architecture relative to DataNodes, reducing their vulnerability to attack. We also assume that communication between NameNodes and DataNodes is cryptograhpically protected, so that a man-in-the-middle cannot forge or replay messages.

Table I
HATMAN CODE BREAKDOWN

Component	Description	Lines
ActivityGen	Generate police activities.	300
NameNode Additions:		
TrustMatrix	Compute local trust matrix and dispatch eigenvector job computations.	3500
Evaluator	Evaluate group responses and select results.	4000
Interface	Initialize and finalize jobs.	1500
Job Code:		
Compare	Test results for equivalence.	600
EigenVector	Compute left eigenvector of local trust matrix.	300
Clustering	Police activity code.	600
Total		11000

Following prior work [25], [26], we assume that most (but not necessarily all) jobs are deterministic and stateless, so that inconsistencies are indicative of integrity violations. This assumption is valid for a large category of data processing jobs that dominate Hadoop and similar cloud frameworks. Non-determinism based on random number generation can be adapted to our system by requiring job-authors to expose random number generator seeds as job inputs, so that they can be duplicated across replica groups.

Attacker-compromised nodes in our model occasionally (or always) submit incorrect results for jobs that they process. Confidentiality and denial of service attacks are outside our scope, but are addressed by a large body of other work (cf., [9], [10]).

III. IMPLEMENTATION

Our implementation of Hatman consists of about 11,000 lines of Java code added to the open source release of Hadoop v0.20.3. Table I reports a size breakdown of each component's programming.

The majority of the implementation modifies Hadoop's JobTracker and NetworkTopology modules to adjust the distribution of input and output files, and modify the scheduling of jobs during Map and Reduce phases in accordance with Algorithm 1. This accounts for about 82% of the implementation.

Approximately 1500 lines of additional MapReduce code implement distributed algorithms for result comparison, eigenvector computation, and police activity jobs. For our police activities we used a K-means clustering algorithm that partitions randomly generated data sets of 10,000 data points into 2 clusters. A separate ActivityGen module submits police activities to NameNodes during idle times or other periods of low activity. To maximize the effectiveness of the police jobs, they are submitted with parameters $n = 1$ and $k = 3$. Group size $n = 1$ helps the trust manager reliably

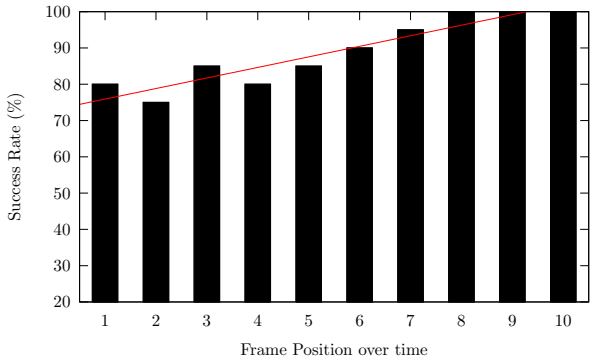

Figure 2. Success rate over time

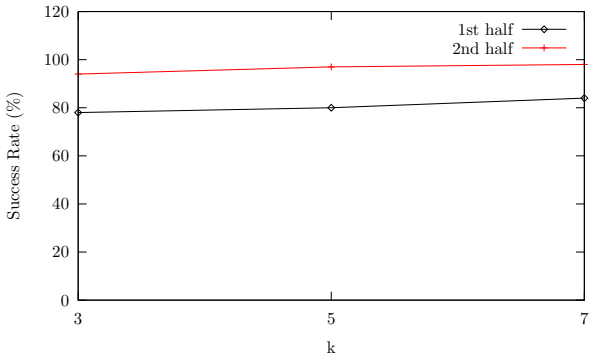

Figure 3. Two-frame comparison for $n = 2$

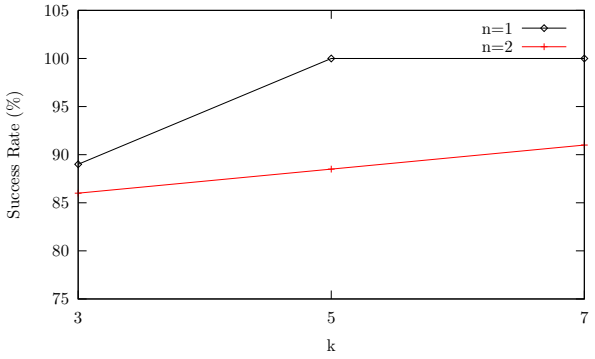

Figure 4. k versus success rate

trace inconsistencies to one misbehaving node per group, and small, odd replication factor k helps break potential ties with low overhead.

Our test architecture is a Hadoop cluster consisting of 8 DataNodes and 1 NameNode. Node hardware consists of Intel Pentium IV 2.40–3.00GHz processors with 2–4GB memory each, running Ubuntu operating systems. In each test, 2 of the 8 nodes (25%) are malicious, randomly returning correct or incorrect results for each job they are assigned.

IV. Results and Analysis

In all experiments we used Equation 3 for evaluation with group size weighted at $w = 0.2$ and group reputation weighted at $1 - w = 0.8$. This yielded the best success rates in all cases. Police activities were submitted at regular intervals between user-submitted jobs, and account for 30% of the network's overall load.

Fig. 2 illustrates Hatman's success rate in selecting correct job outputs in a Hadoop cloud of 25% malicious nodes, with user-submitted jobs having group size $n = 1$ and replication factor $k = 3$. Each data point reports the average success rate over a frame consisting of 20 user activities. Initially the success rate is 80% because there is initially no reputation information for the nodes. However, by frame 8 all the malicious nodes have been identified and the success rate rises to 100%. The average success rate over all frames is 89%.

Fig. 3 considers the same experiment but with the results divided into only two frames (1st half and 2nd half) of 100 activities each, an increased group size of $n = 2$, and an increased replication factor k ranging from 3 to 7. The plot for the 2nd half of the experiment is substantially above the one for the 1st half in all cases, illustrating that after 100 activities the trust algorithm has converged to 96.33% accuracy on average. As expected, higher replication factors push the success rate even higher—near 100% with $k = 7$.

Figures 4–5 examine the impact of replication factor k and group size n more directly. Fig. 4 shows that increasing the replication factor can substantially increase the success rate for any given frame on average. The impact is more pronounced when n is small because feedback from replica inconsistencies is more node-specific with smaller group sizes, leading to more accurate blame assigned by the trust manager. In fact, with sufficiently high k and low n, accuracy can be pushed almost arbirarily high, as seen by the 100% success rate at $k = 7$ and $n = 1$.

Fig. 5 demonstrates the high scalability of our approach by showing how activity times remain almost completely flat as k increases. This is because all significant computations associated with replica management are fully parallelized across the entire cloud. Note that $k = 7$ and $n = 2$ results in a total load $kn = 14$ that is almost twice the size of our cloud. Nevertheless, activity time remains comparable to $k = 3$ and $n = 2$.

Based on this preliminary evidence, we believe that Hatman will scale extremely well to larger Hadoop clusters with larger numbers of data nodes. Each additional data node adds to the size of the trust matrix, but since all trust matrix operations are distributed across all available data nodes, each additional node also increases the power of the cloud to manage the larger trust computations. This agrees with experimental evidence from prior work showing that EigenTrust and similar distributed reputation-management systems scale well to large networks [28].

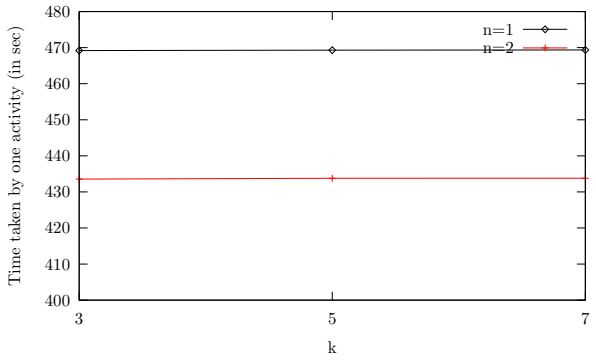

Figure 5. k versus activity time

V. Related Work

Cloud computing security has exploded into a vast research area in recent years [14], with much of the work focusing on data privacy [9]. Data integrity is a second major concern that involves challenges related to secure storage (e.g., [29], [30]) and secure integrity attestation of computation results. The latter is the subject of our work.

AdapTest [25] and RunTest [26] implement cloud service integrity attestation for the IBM System S stream processing system [24] using attestation graphs. Always-agreeing nodes form a clique in the graph, facilitating detection of malicious collectives.

In contrast, our work considers a reputation-based trust management approach to integrity violation detection in Hadoop clouds. Trust management systems probabilistically anticipate future misbehavior of untrusted agents based on their histories of past behavior. Reputation-based trust managers, such as EigenTrust [23], NICE [31], and DCRC/CORC [32], assess trust based on reputations gathered through personal or indirect agent experiences and feedback.

Opera [33] employs reputation-based trust management to improve Hadoop computation efficiency. It tracks node trust as a vector of efficiency-related considerations, such as node downtime and failure frequency. However, malicious behavior in the form of falsified computation results are not considered, making it unsuitable for protecting against data integrity attacks.

Policy-based trust management [34] has been used as a basis for allowing cloud users to intelligently select reliable cloud resources for their computations, and to provide accountability of cloud providers to their customers [35], [36]. These approaches necessarily involve re-architecting clouds to expose some or all of their internal resources to users, so that users can make informed choices regarding those resources.

Peer-to-peer, distributed, decentralized trust management has also been recognized as a natural means of providing inter-cloud security guarantees [37], [38]. Each cloud acts as an individual peer in a super-cloud with no central authority. Inter-cloud computations are then partitioned and distributed based in part on cloud reputations.

As an alternative to trust management, traditional byzantine fault tolerance has been used extensively to detect and isolate malicious behavior in networks of replicated services, including clouds [39], [40]. However, these solutions typically involve implementation of new communication protocols for untrusted agents, complicating their application to existing, large-scale cloud implementations such as Hadoop.

Although there is strong evidence that EigenTrust and similar trust management approaches scale well to networks with large numbers of data nodes [28], NameNode scalability is not something we studied. Hadoop's use of a single NameNode has been identified in the literature as a potential bottleneck [41], and several current works are exploring the feasibility of distributing NameNode computations and metadata [42], [43]. Hatman benefits from these advancements, since they offer opportunities to more widely distribute its trust matrix metadata.

VI. Conclusion

Hatman extends Hadoop clouds with reputation-based trust management of slave data nodes based on EigenTrust [23]. To obtain high scalability, all trust management computations are formulated as distributed cloud computations. This leverages the considerable computing power of the cloud to improve the data integrity of cloud computations. Experiments show that Hatman consistently obtains over 90% reliability after just 100 jobs even when 25% of the network is malicious, and scales extremely well with increased job replication rates.

Although our implementation augments a full-scale, production-level cloud system, our evaluation is preliminary. In future work we plan to extend our analysis to consider more sophisticated data integrity attacks (e.g., malicious collectives) against larger clouds. We also plan to investigate the impact of job non-determinacy on integrity attestations based on consistency-checking.

Acknowledgment

This material is based upon work supported by the National Science Foundation under Grant No. NSF-0959096. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] Amazon, "Amazon elastic compute cloud," http://aws.amazon.com/ec2.

[2] Microsoft, "Windows Azure," http://www.windowsazure.com.

[3] Apache, "Hadoop," http://hadoop.apache.org.

[4] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg, and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," in *Future Generation Computer Systems (FGCS)*, 2009, pp. 599–616.

[5] A. Weiss, "Computing in the cloud," in *ACM Networker*, 2007, pp. 16–25.

[6] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A Berkeley view of cloud computing," U.C. Berkeley, Tech. Rep. UCB/EECS-2009-28, 2009.

[7] S. Pearson, Y. Shen, and M. Mowbray, "A privacy manager for cloud computing," in *Proc. IEEE Int. Conf. on Cloud Computing (CLOUD)*, 2009, pp. 90–106.

[8] Fujitsu, "Personal data in the cloud: A global survey of consumer attitudes," Fujitsu Research Institute, Tech. Rep., 2010.

[9] M. D. Ryan, "Cloud computing privacy concerns on our doorstep," *Communications of the ACM (CACM)*, vol. 54, no. 1, pp. 36–38, 2011.

[10] D. Chen and H. Zhao, "Data security and privacy protection issues in cloud computing," in *Proc. Int. Conf. on Computer Science and Electronics Engineering (ICCSEE)*, 2012, pp. 647–651.

[11] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Communications of the ACM (CACM)*, vol. 51, no. 1, pp. 107–113, 2008.

[12] B. Iannotta, "Securing the cloud: The intel community's high-stakes bid to keep its data safe," *C4ISR Journal, DefenseNews*, November 2011.

[13] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *Journal of Network and Computer Applications*, vol. 34, no. 1, pp. 1–11, 2011.

[14] K. W. Hamlen, M. Kantarcioglu, L. Khan, and B. Thuraisingham, "Security issues for cloud computing," *Int. Journal of Information Security and Privacy (IJISP)*, vol. 4, no. 2, pp. 36–48, 2010.

[15] J. Du, W. Wei, X. Gu, and T. Yu, "Towards secure dataflow processing in open distributed systems," in *Proc. ACM Workshop on Scalable Trusted Computing (STC)*, 2009, pp. 67–72.

[16] S. Berger, R. Caceres, D. Pendarakis, R. Sailer, E. Valdez, R. Perez, W. Schildhauer, and D. Srinivasan, "TVDc: Managing security in the trusted virtual datacenter," *ACM SIGOPS Operating Systems Review (OSR)*, vol. 42, no. 1, pp. 40–47, 2008.

[17] A. S. Ibrahim, J. Hamlyn-Harris, J. Grundy, and M. Almorsy, "CloudSec: A security monitoring appliance for virtual machines in the IaaS cloud model," in *Proc. 5th Int. Conf. on Network and System Security (NSS)*, 2011, pp. 113–120.

[18] M. Christodorescu, R. Sailer, D. L. Schales, D. Sgandurra, and D. Zamboni, "Cloud security is not (just) virtualization security," in *Proc. ACM Cloud Computing Security Workshop (CCSW)*, 2009, pp. 97–102.

[19] Y. Chen, V. Paxson, and R. H. Katz, "What's new about cloud computing security?" Electrical Engineering and Computer Sciences, University of California at Berkeley, Tech. Rep. UCB/EECS-2010-5, January 2010.

[20] M. A. Morsy, J. Grundy, and I. Müller, "An analysis of the cloud computing security problem," in *Proc. 1st Asia-Pacific Workshop on Cloud Computing (APSEC-CLOUD)*, 2010.

[21] Cloud Security Alliance, "Top threats to cloud computing v1.0," March 2010, http://cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.

[22] M. Blaze, J. Feigenbaum, and J. Lacy, "Decentralized trust management," in *Proc. IEEE Security & Privacy (S&P)*, 1996, pp. 164–173.

[23] S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina, "The EigenTrust algorithm for reputation management in P2P networks," in *Proc. Int. World Wide Web Conf. (W3C)*, 2003, pp. 640–651.

[24] B. Gedik, H. Andrade, K.-L. Wu, P. S. Yu, and M. Doo, "SPADE: The System S declarative stream processing engine," in *Proc. ACM SIGMOD Int. Conf. on Management of Data*, 2008, pp. 1123–1134.

[25] J. Du, N. Shah, and X. Gu, "Adaptive data-driven service integrity attestation for multi-tenant cloud systems," in *Proc. IEEE Int. Workshop on Quality of Service (IWQoS)*, 2011, pp. 1–9.

[26] J. Du, W. Wei, X. Gu, and T. Yu, "RunTest: Assuring integrity of dataflow processing in cloud computing infrastructures," in *Proc. ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, 2010, pp. 293–304.

[27] M. Jakubowski, R. Venkatesan, and Y. Yacobi, "Quantifying trust," Microsoft Corporation, Tech. Rep. MSR-TR-2009-119, 2009.

[28] A. G. West, S. Kannan, I. Lee, and O. Sokolsky, "An evaluation framework for reputation management systems," in *Trust Modeling and Management in Digital Environments: From Social Concept to System Development*, Z. Yan, Ed. IGI Global, 2010, pp. 282–308.

[29] S. Nepal, C. Shiping, Y. Jinhui, and D. Thilakanathan, "DIaaS: Data integrity as a service in the cloud," in *Proc. IEEE Int. Conf. on Cloud Computing (CLOUD)*, 2011, pp. 308–315.

[30] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in *Proc. ACM Conf. on Computer and Communications Security (CCS)*, 2009, pp. 187–198.

[31] S. Lee, R. Sherwood, and B. Bhattacharjee, "Cooperative peer groups in NICE," in *Proc. Annual Joint Conf. of the IEEE Computer and Communications Societies (INFOCOM)*, 2003, pp. 1272–1282.

[32] M. Gupta, P. Judge, and M. H. Ammar, "A reputation system for peer-to-peer networks," in *Proc. ACM Int. Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV)*, 2003, pp. 144–152.

[33] T. Nguyen and S. Weisong, "Improving resource efficiency in data centers using reputation-based resource selection," in *Proc. Int. Conf. on Green Computing (ICGREEN)*, 2010, pp. 389–396.

[34] M. Blaze, J. Feigenbaum, and M. Strauss, "Compliance checking in the PolicyMaker trust management system," in *Proc. Int. Financial Cryptography Conf. (FC)*, 1998, pp. 254–274.

[35] P. D. Manuel, S. Thamarai Selvi, and M.-E. Barr, "Trust management system for grid and cloud resources," in *Proc. Int. Conf. on Advanced Computing (ADCONS)*, 2009, pp. 176–181.

[36] R. Ko, P. Jagadpramana, M. Mowbray, S. Pearson, M. Kirchberg, Q. Liang, and B. Lee, "TrustCloud: A framework for accountability and trust in cloud computing," in *Proc. IEEE World Congress on Services (SERVICES)*, 2011, pp. 584–588.

[37] J. Abawajy, "Determining service trustworthiness in intercloud computing environments," in *Proc. Int. Sym. on Pervasive Systems, Algorithms, and Networks (I-SPAN)*, 2009, pp. 784–788.

[38] W. Li and L. Ping, "Trust model to enhance security and interoperability of cloud environment," in *Proc. Int. Conf. on Cloud Computing (CLOUD)*, 2009, pp. 69–79.

[39] Y. Zhang, Z. Zheng, and M. R. Lyu, "BFTCloud: A byzantine fault tolerance framework for voluntary-resource cloud computing," in *Proc. IEEE Int. Conf. on Cloud Computing (CLOUD)*, 2011, pp. 444–451.

[40] R. Kotla, L. Alvisi, M. Dahlin, A. Clement, and E. Wong, "Zyzzyva: Speculative byzantine fault tolerance," *ACM Trans. on Computer Systems (TOCS)*, vol. 27, no. 4, 2009.

[41] K. V. Shvachko, "HDFS scalability: The limits of growth," *;login:*, vol. 52, no. 2, pp. 6–16, 2010.

[42] E. Molina-Estolano, C. Maltzahn, B. Reed, and S. A. Brandt, "Haceph: Scalable metadata management for Hadoop using Ceph," Poster at the 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2010.

[43] A. S. Talwalkar, "HadoopT – breaking the scalability limits of Hadoop," Master's thesis, Rochester Institute of Technology, Rochester, New York, January 2011.