2015 IEEE 17th International Conference on High Performance Computing and Communications (HPCC), 2015 IEEE 7th International Symposium on Cyberspace Safety and Security (CSS), and 2015 IEEE 12th International Conf on Embedded Software and Systems (ICESS)

A Framework for Privacy-Aware Computing on Hybrid Clouds with Mixed-Sensitivity Data

Xiangqiang Xu
School of Engineering and Computer Science
Washington State University
Vancouver, WA 98686
Email: xiangqiang.xu@wsu.edu

Xinghui Zhao
School of Engineering and Computer Science
Washington State University
Vancouver, WA 98686
Email: x.zhao@wsu.edu

***Abstract*—Security and privacy have long been the primary concerns of cloud computing platforms. Hybrid clouds provide potentials for handling data separately based on their sensitivity, harnessing the heterogeneous architecture. In this paper, we design and implement a privacy-aware framework to address data privacy challenges by supporting sensitive data segregation on hybrid clouds. Specifically, we model data sensitivity in a comprehensive and dynamic manner using a set of tagging mechanisms, which include a coarse-grained file level tagging, a fine-grained line level tagging, temporal and spatial tagging. The framework can also process data dynamically generated on-the-fly. We demonstrate the effectiveness of this framework using a big data application, and the experimental results show that the privacy-aware framework successfully enables data sensitivity protection while providing good performance.**

***Keywords*—Hybrid Clouds, Privacy, Data Sensitivity, Tagging, MapReduce**

I. Introduction

Cloud computing has recently emerged as a new computing paradigm which fundamentally changes the way resources are shared in a large scale system. The rapidly growing availability of public clouds has significantly increased the computational/storage capacity of regular users, implicitly leading to the popularity of big data applications. However, a number of inevitable risks and challenges have been introduced by this new computing paradigm, which – to a large extent – undermine the effectiveness of the traditional security protection mechanisms. Security and privacy issues, such as availability, confidentiality, data integrity, control and audit for security on clouds, create obstacles for delivering resources to users in a secure way. A recent security threat happened in August 2014, when hackers exploited Find My iPhone service, brute-forced users' Apple IDs and accessed their iCloud data. Large amounts of private information that had been synced to iCloud storage were revealed [12].

Besides the security issues, privacy also presents challenges for the adoption of cloud computing. The cloud techniques cannot be thoroughly exploited unless these privacy challenges can be addressed properly. On the one side, organizational data contains sensitive information which cannot be shared with cloud service providers without proper privacy protection. Yet on the other side, cloud service providers do not offer high-level security assurance. For instance, in a big data computation, a significant amount of computation workload does contain sensitive information. The exposure of these sensitive data in either the public storage service or the process of transmitting results can possibly violate the security and privacy standards a user would expect. Moreover, although the heterogeneous architecture of cloud computing, e.g. hybrid clouds, provides potentials for security and privacy protection, it inevitably increases the complexity of data processing. As a result, most data-intensive frameworks, including MapReduce [3], do not support hybrid clouds.

In this paper, we propose a privacy-aware framework on hybrid clouds to guarantee data privacy by segregating the sensitive data from the rest, and processing the sensitive data on the private cloud only. Specifically, the sensitive data can be tagged and retained on the private cloud so that data privacy can be protected, and a fraction of computations which only requires non-sensitive data can be off-loaded to the public cloud for better performance. We have developed a set of tagging mechanisms to represent data sensitivity at different granularities, in different scenarios. These include coarse-grained file-level tagging, fine-grained line-level tagging, temporal and spatial tagging. File-level tagging and line-level tagging are used to tag sensitivity at different granularities. Temporal and spatial tagging are used in dynamic scenarios where sensitivity changes over time and space. These mechanisms aim for providing a comprehensive privacy protection. In addition, we propose a simple performance optimization model for achieving better performance while the data privacy protection is guaranteed. Our framework is built using MapReduce.NET on Aneka hybrid cloud [7]. We use Amazon EC2 as the public clouds and a local VMware clouds as the private clouds. We have evaluated our framework using a big data application, and designed experiments to exercise both static tagging, where data are available for processing, and dynamic tagging, where data are generated at run-time. The experimental results show that our framework can effectively protect the data privacy for both static and dynamic data with only minimum overhead comparing to the public Amazon EC2 clouds. Our framework also shows good scalability in terms of sensitivity level.

The rest of the paper is organized as follows. Section II reviews related work. The design and implementation of our privacy-aware framework is presented in Section III. In Section IV, we use a big data application to evaluate our framework in different sensitivity settings. Finally, Section V concludes the paper and discusses future directions of this research.

978-1-4799-8937-9/15 $31.00 © 2015 IEEE
DOI 10.1109/HPCC-CSS-ICESS.2015.110

II. Related Work

Security and privacy issues have long been the major concern in cloud computing ever since it has emerged. A common approach to address the data security and privacy challenges is to encrypt data before transmission to the public cloud. However, traditional encryption techniques do not allow computation to be carried out on the encrypted data [13]. Therefore, users must download the data from the cloud, decrypt before they can run computations on them, which is obviously inefficient. Traditional cryptographic techniques could only perform limited operations on the encrypted data, therefore, these approaches can be only applied to a limited number of applications. Examples of these include secure and private sequence comparison [1] which is a protocol for sequence comparisons that no party could leak their own private sequence information to others, and encrypted data searching [11], which is a cryptographic scheme for searching encrypted data in a cryptographic system. In addition, homomorphic encryption is very expensive for large-scale computations [4]. Overall, the secret-sharing techniques can address the security and privacy issue to some extent, however, these approaches require significant amount of data to be transmitted over the network [6]. Therefore, they are not scalable when the size of the data to be processed increases.

Data segregation is another type of approach besides encryption. Airavat [9] addresses data privacy challenges by setting a mathematical boundary for potential leakage and violations. It ensures mandatory access control and provides different policies for processing sensitive data. Airavat protects data privacy from violations or leakage by confining users' computations, and ensures that the computation outputs do not violate the privacy of inputs [5]. The limitation of Airavat is that it cannot confine all kinds of computations which are performed by untrusted code [9].

Unlike Airavat, Sedic [14] trusts the cloud platform, and focuses on protecting sensitive data from the public cloud. A security mechanism is proposed to protect sensitive data on hybrid clouds. Sedic addresses data sensitivity challenges by pre-labeling input data, duplicating all data to the public cloud and the private cloud, while excluding sensitive data from the public cloud. In a computation process, mappers operate data on both public cloud and private cloud, but it sends all temporary results back to the private cloud to avoid leaking sensitive data. However, Setic may still leak the locations and the length of the sensitive data [13]. In addition, iterative MapReduce is not supported by Sedic [3].

Among all data segregation based approaches, Tagged-MapReduce [13] is closely related to our work. Tagged-MapReduce addresses the data privacy challenges by extending MapReduce to support secure computing with mixed-sensitivity data on hybrid clouds. Different from Sedic, Tagged-MapReduce presents a general security framework for addressing data sensitivity challenges by using explicit tagging. This tagging mechanism protects data sensitivity since sensitive data are separated from the public cloud. Sensitive data will be tagged and retained on the private cloud only. As a result, the risk of leaking data privacy can be reduced for privacy-aware applications. Comparing to Tagged-MapReduce, our work supports more comprehensive types of tags, including coarse-grained file tags, fine-grained line tags, temporal and spatial tags, as well as combinations of those types.

III. Privacy-Aware Hybrid Cloud Framework

In a large number of applications, privacy issues are associated with users' data. A key to secured execution of these applications is to protect the *sensitive* data from any malicious or unauthorized access. A hybrid cloud system, which synthesizes both open public clouds and more secured private clouds, provides a natural platform for address the privacy issues of big data applications. Specifically, we develop tagging mechanisms to retain sensitive data on private clouds during the computation, in order to guarantee privacy while enjoying the capacity of the public clouds.

A. Sensitivity Tags

Many existing works view sensitivity as a fixed property of data files. Unlike these approaches, we model data sensitivity in a more comprehensive and dynamic manner. First, we model sensitivity at different granularity levels, e.g., file level and line level. This enables finer-grained control on data segregation. Second, we allow the sensitivity property of a data file (or data item) change over time. This is critical for applications executing in a dynamic environment where users' requirements may change over time. Specifically, we use sensitivity tags to differentiate data in an application. These tags can be used to specify sensitivity property at different granularities over time. Table I shows four types of sensitivity tags.

TABLE I. Sensitivity Tags

Sensitivity Tag Type	Description
TAG_f	File level sensitivity tag
TAG_l	Line level sensitivity tag
$TAG_x^{[t_1,t_2]}$	Temporal sensitivity tag ($x \in \{f, l\}$)
$TAG_x^{[loc]}$	Spatial sensitivity tag ($x \in \{f, l\}$)

File-level sensitivity tags (TAG_f) can be used to mark sensitive data files. A data file with TAG_f attached is considered sensitive indefinitely, and should be retained on private clouds for processing. Unlike the file-level tags, a line-level sensitivity tag (TAG_l) is associated with a specific line in a file, instead of a file as a whole. Line-level sensitivity tags provide a finer-grained control on the sensitivity of data. Both TAG_f and TAG_l are permanent sensitivity tags, which indicate a file or part of a file contains sensitive data indefinitely. However, in certain circumstances, the sensitivity property of a data file/item may change over time. This change, in most of the cases, reflects changes in either the data or users' requirements. For example, a credit card number may only be sensitive before its expiration date. To represent this type of temporary sensitivity, we use a temporal sensitive tag $TAG_x^{[t_1,t_2]}$, where the data type x can be either f (file) or l (line). This tag specifies that the data associated with it is sensitive only during the time interval $[t_1, t_2]$. Besides temporal information, the location of a data file may also be associated to its sensitivity. For example, an organization's employee list may not be considered sensitive data within its own internal network, but it should be protected if located in public storage space. The spatial sensitivity tag, $TAG_x^{[loc]}$, can be used in this scenario. Here, $[loc]$ is a list of locations, and x can be either f or l.

This tag specifies that the data it is associated with is sensitive if it resides on one of the locations in $[loc]$.

Temporal and spatial sensitivity tags can be combined and attached to the same data. For example, a data file with both $TAG_f^{[t_1,t_2]}$ and $TAG_f^{[loc]}$ is considered to be sensitive during the time interval $[t_1, t_2]$ no matter where it is located. In addition, it is also sensitive when it is located on one of the locations in $[loc]$ despite the time. A hybrid tag, $TAG_f^{[t_1,t_2],[loc]}$, means that a data file is sensitive if during time interval $[t_1, t_2]$ it is located in $[loc]$.

These sensitivity tags, when attached to data, can precisely represent sensitivity information over time and space, at different granularities. In addition, different sensitivity tags can be logically reduced based on their semantics. Suppose F is a file which consists n lines, $L_1, L_2, ..., L_n$, some logical operations on sensitivity tags of F are shown as follows.

$$L_1 \cdot TAG_l + L_1 \cdot TAG_l + ... + L_n \cdot TAG_l = F \cdot TAG_f$$

$$X \cdot TAG_x^{[t_1,t_2]} + X \cdot TAG_x^{[t_2,t_3]} = X \cdot TAG_x^{[t_1,t_3]}$$

$$X \cdot TAG_x^{[loc_1]} + X \cdot TAG_x^{[loc_2]} = X \cdot TAG_x^{[loc_1,loc_2]}$$

The above equations show how multiple fine-grained sensitivity tags can be reduced to a single coarse-grained sensitivity tag, and how temporal and spatial sensitivity tags can be reduced, respectively.

B. Tagging Workflow

In cloud computing, the dominate programming paradigm is MapReduce [3], in which data can be partitioned and processed in parallel. In the MapReduce programming model, different data partitions are independent from each other, which provides a natural platform to integrate our tagging mechanism into the execution of applications. The key idea is to partition the input data based on their tags, and then process sensitive and non-sensitive data accordingly on a hybrid cloud.

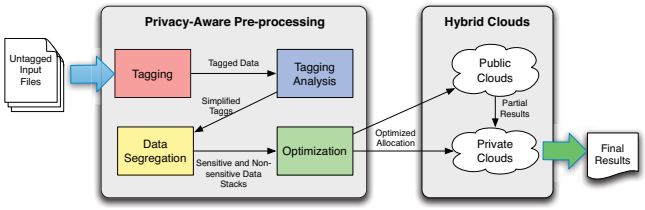

Fig. 1. Workflow of a MapReduce Application on Privacy-Aware Hybrid Cloud

Figure 1 shows the workflow of a MapReduce application executed on our privacy-aware framework with sensitivity tagging. A typical workflow consists of two phases, a privacy-aware data pre-processing, and an execution phase on both public and private clouds.

The pre-processing phase aims for segregating data based on their sensitivity, and then allocate the MapReduce tasks accordingly to public and private clouds. This phase consists of four procedures, *Tagging, Tagging Analysis, Data Segregation,* and *Optimization*. The Tagging procedure takes as input the untagged input files and the configuration of data sensitivity, [1] attaches different sensitive tags to files and lines based on the configuration, and sends the tagged data to the Tagging Analysis procedure. The Tagging Analysis procedure then analyzes the sensitivity tags and reduces them if possible, based on the sensitivity tag operations. After the analysis, the input data with simplified tags attached are sent to the next procedure, Data Segregation, which separates data according to their sensitivity, and forms two data stacks for processing on hybrid clouds: sensitive data stack, and non-sensitive data stack. Once the Data Segregation procedure is completed, we can guarantee privacy as long as all sensitive data stacks are retained on the private clouds. However, the workload for the non-sensitive data stacks can be further tuned among public and private clouds for better overall performance. The Optimization procedure is for this purpose. It takes as input the data stacks and optimize the performance by reallocating the non-sensitive data stacks among public and private clouds. A detailed description of the optimization can be found in Section III-C.

The data pre-processing phase forms a workload distribution between public and private clouds for the execution phase. During the execution phase, the public clouds and private clouds process the data allocated to them in parallel. At the end of the execution, the public clouds send its partial results to private clouds, which in turn combine results from both clouds and generate the final results for the application. Note that the execution of the two phases in our framework can be running in parallel, in order to process dynamically generated data. In that scenario, the input files are generated on-the-fly, and the output result files are also generated at runtime.

C. Performance Optimization

The Data Segregation procedure synthesizes sensitive data to form the sensitive data stack, and the non-sensitive data form another data stack. To guarantee privacy, the sensitive data stack must be processed on the private clouds, but the non-sensitive data can be allocated to either public or private clouds without violating the privacy standard. To this end, we develop a simple optimization model to reallocate the workload of processing non-sensitive data. Table II outlines the parameter used in our performance optimization model.

TABLE II. PARAMETERS DESCRIPTION

Parameter	Description
D	Total data size
d	Public data allocated to private clouds
s	Ratio of sensitive data ($0 \leq s \leq 1$)
R_{pri}	Data processing rate on private clouds
R_{pub}	Data processing rate on public clouds
W_{pri}	Workload on private clouds (in data size)
W_{pub}	Workload on public clouds (in data size)
T_{comm}	Communicational time for returning result from public clouds to private clouds

The workload allocated to the private clouds includes processing all sensitive data and a fraction of non-sensitive data, specified by d. The rest of non-sensitive data is allocated on the public clouds. We assume that the performance on the clouds is relatively stable, which implies that R_{pub} and R_{pri}

[1] This configuration is provided by the user as sensitivity rules of their data.

do not change significantly. [2] In this case, the execution times on private and public clouds can be calculated as follows.

$$T_{pri} = \frac{W_{pri}}{R_{pri}} = \frac{s*D+d}{R_{pri}}$$

$$T_{pub} = \frac{W_{pub}}{R_{pub}} = \frac{D-d-s*D}{R_{pub}}$$

Taking into consideration the communication time between public and private clouds, the total execution time is

$$T = max(T_{pri}, T_{pub} + T_{comm})$$

Given the above equations, we can then formulate an optimization problem which aims for balancing the workload between public and private clouds, so that the idle time is minimized, under the constraint that the data privacy is protected. That is,

minimize

$$T_{idle} = |T_{pri} - (T_{pub} + T_{comm})|$$

subject to

$$\begin{cases} T_{pri} \geq 0, \\ T_{pub} \geq 0, \\ 0 \leq s \leq 1, \\ 0 \leq d \leq D * (1 - s). \end{cases} \quad (1)$$

Note that if the data to be processed are static files, the sensitivity ratio s is fixed, and the performance of both clouds are stable, the optimization procedure only needs to be executed once. Otherwise, it needs to be invoked periodically to tune the performance of hybrid clouds at run-time. In a static scenario, the performance of both clouds, R_{pub} and R_{pri}, can be obtained by processing a small sample data file. In the dynamic scenario, where data are generated on the fly, these parameters can be obtained using the performance during the previous time interval.

D. System Architecture and Configuration

Our framework is built on top of Aneka [8], a software which supports the deployment of hybrid clouds. The system architecture of our privacy-aware hybrid cloud is shown in Figure 2.

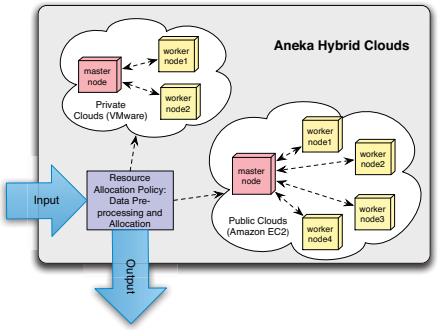

Fig. 2. Privacy-Aware Hybrid Cloud: System Architecture

[2]If the performance of the clouds is not stable, the optimization procedure needs to be invoked periodically to capture the dynamicity of the system.

As shown in Figure 2, we use the Aneka platform to integrate public clouds and private clouds. On both clouds, MapReduce applications are processed in a standard master-worker paradigm. For private clouds, we use a VMware cloud, and we use the same configuration for all nodes: dual-core processor (2.7 GHz Intel Core i5) with a 8 GB memory and 500GB disk space. For public clouds, we use the Amazon EC2 [10] with *r3.large* instances. The configuration for the each instance is Intel(R) Xeon CPU E5-2670 v2 @2.50GHz 2.49GHz with a15.25GB RAM.

We have developed a resource allocation policy for the hybrid clouds, in which we implement and integrate our sensitivity tagging mechanism and optimization model. This is the key component of the framework, and serves as the interface of the entire framework for interacting with users' MapReduce applications.

IV. Evaluation

Experiments have been carried out to evaluate the performance and scalability of our hybrid cloud framework using a MapReduce application with real data collected by the Center for Medicare & Medicaid Services.

A. Dataset and Application

The dataset we use in our experiments is downloaded from Center for Medicare & Medicaid Services (CMS), the official U.S. government site, which provides medicare supports, latest medicare enrollment, benefits and other information [2]. The dataset of CMS contains data records for the 2013 program year, which includes payment information and identification of physicians and hospitals [2]. Table III shows the fields that each data record contains. [3]

TABLE III. Description of Records

Field	Type	Key
General Transaction ID	integer	primary key
Company Name	string	unique
Payment Amount	decimal	-
Payment Name	string	-
Hospital Name	string	-
Physician Name	string	-
Physician Specialty	string	-
Name of Associated Drug	string	-
Contextual Information	string	-
Recipient Address	string	-

The application we use to evaluate our privacy-aware hybrid cloud framework is a MapReduce application which calculates the total payment amount for each *company* in the dataset. We design two sets of experiments to evaluate the performance and scalability of our framework under different scenarios: static tagging and dynamic tagging.

B. Static Tagging

In the static tagging experiments, we assume that the data to be processed are ready before the execution of the application.

[3]Note that the dataset contains more fields than the ones being listed in Table III. We only included relevant data fields due to the limitation of the space.

The first set of experiments is carried out to evaluate the file-level tagging. In these experiments, the size of the dataset to be processed ranges from 100MB to 1GB. We first divide the dataset into smaller files, and then assume that certain percentage (sensitivity ratio s) of these files are sensitive. Our framework marks those sensitive files with the TAG_f tags, and processes all the files accordingly on the hybrid clouds. For comparison purposes, we also measure the performances of using public and private clouds only. The experimental results are shown in Figure 3. When the sensitivity ratio ranges from 0% to 50%, the performance of our framework is close to the performance of public cloud only executions, and the overhead of providing privacy guarantee is negligible. When the sensitivity ratio is higher than 50%, the overhead is higher, however, the performance is still much better than using private clouds only. Note that in reality, for cloud computing applications, the sensitivity ratio of users' data rarely goes higher than 50%.

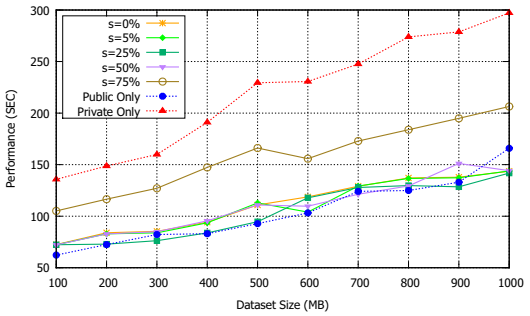

Fig. 3. File Level Tagging: Performance Comparison for Different Sensitivity Ratios

The second set of experiments is for evaluating the fine-grained line tagging. In these experiments, we assume that a portion of the companies consider their data sensitive, and we attach TAG_l tags to all the lines associated with those companies. We then run the application with different sensitivity levels on our framework and measure the performance. The experimental results are shown in Figure 4. Similar to the static file tagging, our framework does not introduce significant overhead comparing to public cloud only executions. Note that in these experiments, sometimes an increase of sensitivity level does not increase the execution time, e.g., the experiments with 50% of sensitivity run faster than that of 75% of sensitivity. This is because the line tagging is based on companies, instead of file sizes. In other words, in these experiments, the sensitivity ratio s only represents the percentage of companies which consider their data sensitive, and it is not as an accurate reflect in sensitive data size as in file-level tagging. However, in general, our framework shows good performance for different sensitivity ratios.

The results from these two sets of experiments illustrate that for applications which process static data, our privacy-aware hybrid cloud framework provides privacy protection at the cost of only a small overhead comparing to the public clouds, which do not consider data privacy issues. Notably, when the ratio of sensitive data is low ($< 50\%$), the performance of our framework comes close to the Amazon EC2 public clouds.

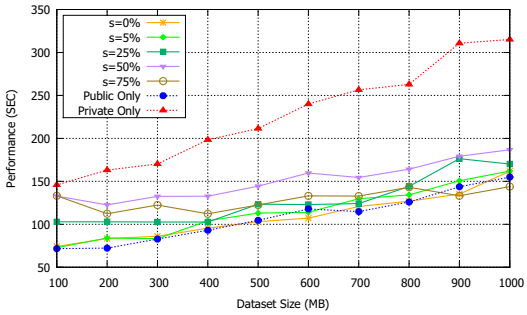

Fig. 4. Static Line Level Tagging: Performance Comparison for Different Sensitivity Ratios

C. Dynamic Tagging

We have also carried out two sets of experiments to evaluate the performance of our framework in dynamic environments, which include two scenarios: dynamic data generation, and dynamic sensitivity ratios.

In the first set of experiments, the data to be processed are generated on-the-fly. Here, data pre-processing (tagging, analysis, and reallocation) and computation execution are running in parallel. Similar to static line tagging, we assume certain companies' data are sensitive and tag them on a line bases when they are generated. We use a parameter *data block size* to represent the granularity of control on real-time processing. Specifically, the framework starts processing data as soon as a new data block is available. In other words, a data block is the smallest unit for data processing in the framework. The smaller the data block size is, the closer the framework is to real-time processing.

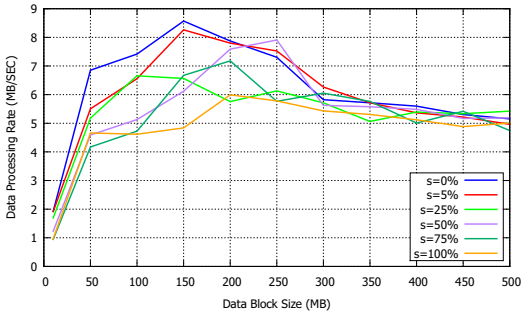

Fig. 5. Dynamic Line Level Tagging: Data Processing Rate vs. Data Block Size

Since in this dynamic scenario, the data processing is an ongoing process, we measure data processing rate as the performance metric for the evaluation. The experimental results are shown in Figure 5. As expected, the data processing rate in general decreases when the data sensitivity ratio increases. Interestingly, when the data block sizes range from 150MB to 250MB, the framework has a higher performance in general, for all sensitivity ratios. This is because in those cases, the system reaches a higher parallelism between pre-processing and execution. Starting from 300MB of data block size, the system performance starts to converge for all sensitivity ratios. This fact indicates that our framework does not introduce

more overhead when processing more sensitive data, when the data are dynamically generated and the data processing unit is large enough. In this case, the overhead caused by processing sensitive data is more likely to be offset by the slow availability rate of the data blocks.

The second set of experiments for dynamic tagging is to investigate the system performance when sensitivity tags are added to the system over time. This is particularly interesting when users' requirements about sensitivity changes during the execution of their applications. In these experiments, we set the data block size to be 200MB. We initiate 110 MapReduce jobs, $j_1, j_2, ..., j_{11}$, each of which processes one data block. In order to increase the data sensitivity ratio over time, we use temporal sensitivity tags $TAG_l^{[t_1,t_2]}$. Specifically, we divide the companies into 10 groups, $g_1, g_2, ..., g_{10}$, each with 10% companies in the dataset. For the data records (lines) which are associated with g_1, we attach temporal sensitivity tag $TAG_l^{[C(j_{10}),C(j_{110})]}$, where $C(j_{10})$ represents the completion time of job j_{10}, and $C(j_{110})$ is the time when all the jobs complete. Similarly, data records associated with g_2, g_3 and on are attached tags $TAG_l^{[C(j_{20}),C(j_{110})]}$, $TAG_l^{[C(j_{30}),C(j_{110})]}$ and so on. This tagging strategy guarantees that the data sensitivity ratio increases by 10% for every 10-job time interval.

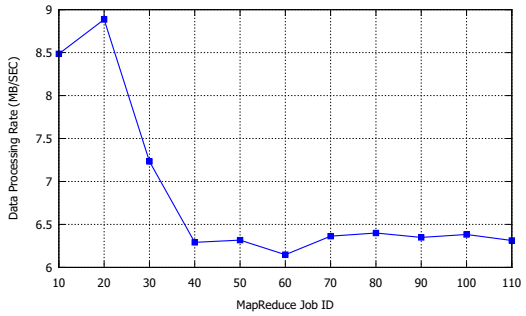

Fig. 6. Dynamic Line Level Tagging: Date Processing Rate vs. Scalability of Sensitivity Level

We measure the data processing rate at the end of every 10-job time interval, and the results are shown in Figure 6. The system performance decreases at the beginning when temporal sensitivity tags are added, but soon stabilizes when the data sensitivity ratio is more than 30% (after job j_{40}). These results illustrate that our privacy-aware hybrid cloud framework is scalable in terms of handling dynamic sensitivity ratios.

V. Conclusion

In this paper, we present a privacy-aware hybrid cloud framework which supports MapReduce applications on hybrid clouds. The key component of this approach is a tagging mechanism. We represent data sensitivity at different granularities using different sensitivity tags, including a coarse-grained file-level sensitivity tag, a fine-grained line-level sensitivity tag, as well as temporal and spatial sensitivity tags. These sensitivity tags can be combined and logically reduced. We have integrated this tagging mechanism into Aneka and MapReduce.NET to support hybrid cloud execution of MapReduce applications. We use a big data application with real data to evaluate our framework in both static and dynamic scenarios. The experimental results show that in both scenarios, the overhead caused by the privacy-aware framework is minimum. Notably, for the cases where the data sensitivity rate is low, the performance of our framework comes close to that of Amazon EC2 public clouds, while still providing privacy protection. In addition, our framework demonstrates good scalability under increasing data sensitivity ratios.

Work is ongoing in a number of directions. First, we are generalizing this framework to accommodate more computations besides MapReduce computations. Second, we are adding dynamic resource provisioning mechanisms to our framework to enhance the flexibility. Third, we are applying this approach to process a variety of data types, e.g., social network applications with mixed-sensitivity data.

Acknowledgment

The authors would like to thank the generous support of an Amazon Web Services Research Grant and a WSU Seed Grant.

References

[1] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and Private Sequence Comparisons," in *Proceedings of the 2003 ACM Workshop on Privacy in the Electronic Society*, ser. WPES '03. New York, NY, USA: ACM, 2003, pp. 39–44.

[2] CMS.gov. (2014) Dataset Downloads. [Online]. Available: http://www.cms.gov/OpenPayments/Explore-the-Data/

[3] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[4] Y. Duan, J. Canny, and J. Zhan, "P4P: Practical Large-scale Privacy-preserving Distributed Computation Robust Against Malicious Users," in *Proceedings of the 19th USENIX Conference on Security*, ser. USENIX Security'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 14–28.

[5] C. Dwork. (2006) Differential privacy. [Online]. Available: http://research.microsoft.com/pubs/64346/dwork.pdf

[6] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, ser. STOC '09. New York, NY, USA: ACM, 2009, pp. 169–178.

[7] M. P. Ltd. (2013) Aneka Installation Guide Aneka 3.0. [Online]. Available: http://www.manjrasoft.com/

[8] Manjrasoft. (2012) Developing MapReduce.NET Applications Aneka 3.0. [Online]. Available: http://www.manjrasoft.com/

[9] I. Roy, S. T. V. Setty, A. Kilzer, V. Shmatikov, and E. Witchel, "Airavat: Security and Privacy for MapReduce," in *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation*, ser. NSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 20–36.

[10] A. W. Service. (2015) Amazon EC2 Instances. [Online]. Available: http://aws.amazon.com/ec2/instance-types

[11] D. X. Song, D. Wagner, and A. Perrig, "Practical Techniques for Searches on Encrypted Data," in *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, ser. SP '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 44–55.

[12] T. Verge. (2014) Reported iCloud Hack Leaks Hundreds of Nude Celebrity Photos. [Online]. Available: http://www.theverge.com/2014/9/1/6092089/nude-celebrity-hack

[13] C. Zhang, E.-C. Chang, and R. Yap, "Tagged-MapReduce: A General Framework for Secure Computing with Mixed-Sensitivity Data on Hybrid Clouds," in *Cluster, Cloud and Grid Computing (CCGrid), 2014 14th IEEE/ACM International Symposium on*, May 2014, pp. 31–40.

[14] K. Zhang, X. Zhou, Y. Chen, X. Wang, and Y. Ruan, "Sedic: Privacy-aware Data Intensive Computing on Hybrid Clouds," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 515–526.