A FRAMEWORK FOR PRIVACY-AWARE COMPUTING ON HYBRID CLOUDS WITH MIXED-SENSITIVITY DATA

By

XIANGQIANG XU

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

WASHINGTON STATE UNIVERSITY
School of Engineering and Computer Science, Vancouver

MAY 2015

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of XIANGQIANG XU find it satisfactory and recommend that it be accepted.

Xinghui Zhao, Ph.D., Chair

Scott Wallace, Ph.D.

Sarah Mocas, Ph.D.

ACKNOWLEDGMENTS

First, I would like to thank my dear advisor, Dr. Xinghui Zhao, who guides my graduate study in the past two years. I have greatly benefited from her genuine, passion and research ideas. When I was frustrated or stuck in my research, she is always patient and glad to give me helps. Time flies, two years' study is short, but her guidance will benefit my whole life.

Then, I would like to show my gratitude to my committee members, Dr. Sarah Mocas and Dr. Scott Wallace, who give me a lot of suggestions and feedback for my thesis. Dr. Mocas, especially, she is always glad to help me with courses and gives me suggestions for my research.

Also, I also would like to thank other faculty members, Dr. David Chiu and Dr. Jie Xu, who give me a lot of suggestions and helps. I am very fortunate that I have a chance to work on the collaborative project named Molecular Dynamic Simulations on Amazon EC2. This project enhances my understanding of cloud computing and also helps me understand how to formulate and execute a research plan.

In addition, the generous support from the Amazon AWS in Education Research Grant is gratefully acknowledged.

Finally, I would like to thank my family and friends. Besides, I want to thank some great couples sincerely: Emory and Carol, Dr. Chau and Lynda, David and Judy. I really appreciate their friendship and kindness.

A FRAMEWORK FOR PRIVACY-AWARE COMPUTING ON HYBRID CLOUDS WITH MIXED-SENSITIVITY DATA

Abstract

by Xiangqiang Xu, M.S.
Washington State University
May 2015

Chair: Xinghui Zhao

Cloud computing has significantly increased the computation/storage capacity for regular users, which leads to the popularity of transplanting large-scale computations, most likely big data applications, to clouds. In fact, a large fraction of big data applications contain sensitive data. However, high-level security cannot always be guaranteed by cloud service providers, thus sensitive information can be easily exposed during the process of data processing.

Hybrid clouds provide potentials for handling data separately based on their sensitivity, harnessing the heterogeneous architecture. In this thesis, we design and implement a privacy-aware framework to address data privacy challenges by supporting sensitive data segregation on hybrid clouds. On the one hand, to guarantee data privacy, sensitive data are tagged and retained on the private cloud, so that data sensitivity can be segregated from the public cloud. On the other hand, to ensure performance, we introduce a simple optimization model, which assigns certain amount of non-sensitive data to the public cloud for processing.

To achieve our goal of guaranteeing data privacy as well as improving performance, we

propose three tagging mechanisms to handle data with mixed-sensitivity, which includes a coarse-grained file level tagging, a fine-grained line level tagging, and a dynamic line level tagging for processing data generated on-the-fly. Specifically, static data with mixed-sensitivity can be handled by using either the file level tagging or the static line level tagging mechanism. Dynamically generated data with mixed-sensitivity can be handled by using the dynamic line level tagging mechanism, in which data are processed dynamically, and tags can be added or removed at run-time.

We demonstrate the effectiveness of these three mechanisms by evaluating them using a big data application. Our experimental results show that the privacy-aware framework successfully enables data sensitivity protection while providing good performance. Our framework shows good scalability when the data sensitivity level increases. In addition, our results demonstrate that our dynamic line level tagging has a stable and reliable performance for processing near-realtime data.

TABLE OF CONTENTS

List of Tables

List of Figures

Chapter 1

Introduction

Cloud computing has recently emerged as a new computing paradigm which fundamentally changes the way resources are shared in a large scale system. Consequently, other aspects of computing systems have also been evolving, including infrastructure architectures, information transmission, and model development for network security. Cost-effective cloud computing provides researchers and users an alternative option to carry out their computations without owning the required resources. 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.

1.1 Challenges in Cloud Computing

Cloud computing provides great potential for executing large computations on shared resources efficiently. 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. In addition, traditional security protection techniques are out of date to protect users' private and sensitive information in a cloud environment since they are no longer applicable to the new relationship between users and providers, namely cloud service user, cloud service provider and cloud provider.

The emergence of data storage and transmission have led to a number of severe security and privacy threats in the past several years [1], which seriously hampers resources sharing over the network in a secure way. For example, in early June 2010, due to vulnerabilities of the AT&T website, the registration information of 114,000 iPad users was leaked to the Internet, which included personal information of many companies' CEOs, Government officials and a number of well-known politicians [2]. In October 2010, some Facebook applications shared user account data with advertisers and Internet companies inappropriately, exposing users' names and even the names of the users' friends [2]. A Google error leaked owners' personal information for nearly 300,000 websites, which includes names, home addresses, emails and phone numbers used to register their websites, even though these users had chosen to keep their information private [3]. Furthermore, distributed data storage and services in clouds make these security and privacy issues even worse. In August 2014, hackers exploited Find My iPhone service, which potentially allowed them to brute-force users' Apple IDs and access their iCloud data [4]. Large amounts of private information that had been synced to iCloud storage were revealed.

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, provide potentials for security/privacy protection, it inevitably increases the complexity if data processing. As a result, most data-intensive frameworks, including MapReduce [5], do not support hybrid clouds.

1.2 Proposed Approach

In this thesis, 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 use MapReduce.NET on Aneka hybrid cloud [6] to build our framework. Three approaches, namely file level tagging, static line level tagging, and dynamic line level tagging, are applied to address the challenges of processing mixed sensitivity data on hybrid clouds. These mechanisms aim for providing privacy protection in different scenarios. Static data (data that are already generated) with mixed-sensitivity can be handled by either the file level tagging or the static line level tagging based on the granularity of their sensitivity. Dynamic data (data that are generated on-the-fly) with mixed-sensitivity can be handled by the dynamic line level tagging mechanism. In addition, we propose a simple performance optimization model for achieving better performance while the data privacy protection is guaranteed. We have evaluated our framework using a big data application, and the experimental results show that our framework can effectively protect the data privacy for both static and dynamic data. Our framework also shows good scalability in terms of sensitivity level.

1.3 Contributions

We design and implement a privacy-aware framework on hybrid clouds for processing data with mixed-sensitivity. Three approaches, including file level tagging, static line level tagging, and dynamic line level tagging, are proposed to handle data with mixed sensitivity in different scenarios.

1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 introduces cloud computing and related security and privacy protection challenges, and presents a detailed review on state-of-the-art approaches to address data privacy challenges on clouds. Chapter 3 describes the design and implementation of our privacy-aware framework on hybrid clouds. Chapter 4 evaluates our framework using a real world application. Finally, Chapter 5 concludes the thesis and presents future directions of this research.

Chapter 2

Related Work

Security and privacy are always the top concerns when organizations or researchers plan to move their private data to the public cloud. On the public cloud, data is located outside of users' network, not to mention that users could only access virtual machines on a shared infrastructure, instead of using physical machines exclusively. In that case, users can only rely on cloud service providers for ensuring data security and privacy, which is not always stable and reliable. Also, public cloud cannot always be thoroughly trusted by users due to potential malicious attacks. Privacy is a matter of organizations' accountability and transparency to data and personal information. Besides, public expectations also have a significant impact on the privacy.

Usually, data-intensive applications involve both public and private data (sensitive data). For example, in an social-network application, users usually do not mind making some data available to the public, such as places they visited, list of friends, and schools they attended. However, they do want to keep certain data private, such as social security number, credit card information, private email address or cell phone number, home address and other sensitive information. Therefore, it is challenging to carry out computations on the public cloud without revealing sensitive data to the public. Fortunately, secure hybrid cloud

provides a promising infrastructure to balance the needs of protecting data privacy as well as taking advantages of public clouds. In this chapter, we first overview cloud computing, and then we summarize existing approaches on addressing data privacy challenges. These approaches can be grouped in three categories, data encryption, trusted computing, and data segregation.

2.1 Cloud Computing

According to the definition from NIST [1], "Cloud computing is a compilation of existing techniques and technologies, packaged within a new infrastructure paradigm that offers improved scalability, elasticity, business agility, faster start-up time, reduced management costs, and just-in-time availability of resources". Cloud computing is a novel approach of resource sharing which enables dynamic resource allocation by accessing shared resource pools over the network. Those shared resources can be consumed on a pay-as-you-go basis. There are three popular infrastructures in cloud computing:

- Infrastructure-as-a-Service (IaaS), which is a raw infrastructure and associated middleware [7]. IaaS providers offer resources including computers (physical/virtual machines) and others on demand from their large resource pools [8]. Cloud users' applications can be configured by installing corresponding operation system and software on the cloud infrastructure. Cloud users use IaaS services on a pay-as-you-go basis. For example, IaaS enables cloud users to store data and run applications based on their demands by expanding resources over network [9].

- Platform-as-a-Service (PaaS), which provides platform service - operations system, required environment, database and web server - for applications development at runtime [8]. PaaS enables cloud users to develop and run applications without buying or

maintaining corresponding hardware and software. For example, PaaS can be used for business intelligence development including dashboards, reporting system, and data analysis [10].

- Software-as-a-Service (SaaS), which supports running software services remotely. SaaS provides users accessibility to software and databases on-demand by billing fees hourly or yearly [8]. SaaS has become a popular model for many business applications, such as DBMS, CAD, CRM, MIS, ERP, and HRM [11]. Unlike IaaS, cloud users do not have to configure or manage the hardware/software environment by themselves. Therefore, costs and time can be reduced by paying SaaS service a monthly or yearly fee. However, SaaS providers could access users data without authorization since users' data are stored on their cloud servers.

2.1.1 Categorization of Cloud Computing

Cloud computing has emerged as a new promising computing paradigm in the last decade. A great amount of resource pools can be shared in certain ways through private and public networks, which significantly facilitates global resource sharing. For instance, public clouds, such as Amazon EC2, enable cloud users to utilize the powerful computational resources by paying a usage fee. Using shared resources on clouds to carry out computations, and storing data has become an inevitable trend for users at present.

The development of cloud computing diversity tries to appeal to more people's demands. Cloud users could choose to deploy their applications, data computations or data storage on different kinds of clouds which include public cloud, private cloud, hybrid cloud and community cloud [12]. Making decisions on which cloud platform to use becomes a prerequisite step, which greatly affects the performance and cost. Table 2.1 [8] [12] [13] illustrates a simple introduction of the current four types of clouds.

Cloud Types	General Usage	Features and Benefits
Public Cloud	Provide services for the public. E.g. Amazon EC2, Apple iCloud, Microsoft Azure.	Scalability; Flexibility; Cost efficiency.
Private Cloud	Used by a single organization; Hosted from inner/outer side. E.g. on-premise private clouds, private clouds hosted from outer side.	High security/privacy; More control; Reliability; Cost efficiency.
Hybrid Cloud	Consist of two or more clouds; Independent, but connect with each other; Support multiple deployment models; Hosted from inner/outer side.	High security/privacy; Scalability; Flexibility; Cost efficiency.
Community Cloud	Shared by multiple organizations; Typically hosted from outer side. E.g. NYSE Technologies for financial trading community.	Collaboration for common goals; Cost sharing.

Table 2.1: Four Types of Clouds

2.1.2 Security and Privacy Issues on Clouds

Third Party Cloud Computing (TPCC), such as Amazon EC2, is an IaaS cloud computing service which provides large computing infrastructures and resources for users. The service is elastic computing which enables users to extend or shrink infrastructure by launching or terminating instances (virtual machines) [7]. Besides, third party clouds allow service providers to maximize the profit as well as maintaining users' confidentiality. However, new vulnerabilities and attacks are also brought into clouds accordingly. Just take virtual machine (VM) attack as an example, hackers can break virtual machine level to violate users' confidentiality on the physical machine level. Since a user's VM and the hacker's VM could be assigned to the same physical machine, as a result, the hackers can access or even modify the user's data by attacking the VM [14].

In fact, there are many benefits we could obtain from clouds, however, the universality is still not broad enough to attract everyone's interests. Acting as a big black box, clouds' internal infrastructures are invisible to clients. Besides, some potential threats come from the

internal side, since cloud providers and cloud service providers have access to these virtual machines and can even violate authentication, confidentiality and integrity. After all, due to the limitations of techniques, clouds are still subject to traditional confidentiality, integrity, availability, privacy issues, and malicious attacks.

Besides the infrastructure level security issues, the goal of data security of cloud computing is to ensure users' access at any time from any place [15]. Proposed data security management models such as self-managed and trusted third party mechanisms suggest that cloud providers and cloud service providers could cooperate with each other to solve security challenges on clouds.

2.1.3 MapReduce Overview

MapReduce is a programming model and associated implementation for processing large datasets with a parallel, distributed algorithm on a cluster [5]. MapReduce supports various data-intensive applications, which includes web search [5], document format conversion [16], and genome sequence analysis [17].

Currently, Hadoop is one of the most widely used open-source MapReduce implementations. It contains a Hadoop Distributed File System (HDFS), which not only provides high performance computing support for MapReduce applications, but also offers a run-time framework to manage MapReduce jobs [18].

MapReduce, as its name suggests, abstracts computation work through two operations: Map() processes filtering and sorting, while Reduce() processes summarizing operations. Both mappers and reducers can be executed independently. For a specific computation, assuming that a MapReduce computation job contains M independent mapper tasks and R independent reducer tasks. First the M independent mappers transform the inputs into intermediate results. Then the intermediate results are partitioned for R reducers [19].

Data entries in MapReduce are presented as key-value pairs. The execution of MapReduce can be divided into two phases. In the map phase, the inputs are partitioned into a list of key-value pairs. In the reduce phase, it assigns the lists of key-value pairs to mappers which are running concurrently. The reducer will be triggered once for each key with associated values, then produces output values as final results [20].

Map takes a pair of data as input and produces a list of key-value pairs as output. Map() is called in parallel by each pair, thus a list of pairs are produced [19]. Then MapReduce collects and groups all pairs who has the same key, each group is combined by each key. That is: Map(key1, value1) $\rightarrow$ List(key2, value2).

Reduce takes the List(key2, value2) as input, Reduce() is called in parallel by each group [19]. Then it produces a collection of a value list List(key3, value3). That is: Reduce(key2, List(value2)) $\rightarrow$ List(key3, value3) [19]. Note that each type of output key and output value might be different from input key and input value.

2.2 Data Encryption Before Transmission to Clouds

There are several types of data security such as data-in-transit, confidentiality of none-secured protocol, and integrity of secured protocol. Cloud service providers cannot always guarantee high level data security/privacy. A common approach to address the data security/privacy challenge is to encrypt data before transmission to the public cloud.

However, traditional encryption techniques such as AES do not allow computation to be carried out on the encrypted data [21]. Users have to download the data from the cloud, decrypt it and then compute it, which is inefficient. Traditional cryptographic techniques could only perform limited operations on the encrypted data, based on this fact, cryptography approaches can be only applied to specific applications, such as secure and private sequence comparison [22] which is a protocol for sequence comparisons that no party could leak their

own private sequence information to others, and encrypted data searching [23], which is a cryptographic scheme for searching encrypted data in a cryptographic system.

Also, homomorphic encryption was found to be very expensive for large-scale computations [24]. Overall, the secret-sharing techniques enable intensive data transmissions among users on different clouds, however, enormous amount of data needs to be transmitted [25]. Consequently, it is difficult to ensure data scalability.

2.3 Trusted Computing and Secure Infrastructure

Another approach to address data privacy is trusted computing, which allows cloud users to check and verify data integrity and authentication. There are three types of approaches with different focuses: minimize lack of trusts, minimize loss of control and identity management on clouds.

2.3.1 Minimize Lack of Trusts

To minimize lack of trusts, P. Angin et al. proposes a user-centric approach for privacy and identity management in cloud computing [26]. The application logic remains on users' host PCs, which allows users to monitor their application and dataflow from all aspects. All outputs from underlying services are sent to the application logic, which serves as a single checkpoint to guarantee compatibility between services [26].

2.3.2 Minimize Loss of Control

There are many access control layers in cloud computing [27]. Based on the deployment model, some layers can be controlled by cloud service providers, and other layers can be controlled by cloud users. A pre-negotiated standard approach [26] is proposed that users

are given control over a decision-making process to manage access control by themselves, less trust in their cloud service providers are required.

Cloud providers are responsible for users' authentication and access control procedures. Some providers authorize authentication to users who allow users to manage it by themselves, but as a matter of fact, the access control management is still the providers' responsibility. Besides, it requires users to trust their service provider in terms of security/privacy protection and management of access control policies.

By using a pre-negotiated standard approach [26], decisions of resources and access between a cloud provider and users can be achieved. In addition, access decisions from users' side should be guaranteed. However, many secure data mechanisms require users to grant keys/certificates if the users query the database, as a result, the data owner need to be involved every time [27]. If the traffic is a major problem, frequent access scenarios are infeasible to use this method.

2.3.3 Identity Management

To increase security and efficiency while decrease cost [28], identity management (IDM) can be used to manage users' authentication, authorization [29], and priorities within or across system boundaries [30]. Trusted third party can be used to secure data identity on clouds [27] [31]. There are three advantages to using identity management. First, IDM enables the use of identity data on untrusted hosts by using self-integrity checking, integrity compromised-apostasies or evaporation [32]. By comparing data types, IDM determines whether identity data should be located on the host or not. Also, third party independence can prevent correlation attacks. Moreover, IDM constructs some trust mechanisms - including authentication data exchange, data transmission and negotiation - which allow users to know who holds their data. In addition, IDM supports selective disclosure [27] for

service providers, which only allows service providers to access the necessary information, other information are not allowed to access. Andreas Ess proposes an identity management mechanism using trusted third party [27] [31], which is shown in Figure 2.1, e-Bay shares the encrypted data with security/privacy standards. To minimize costs and the leakage of data privacy, multiple parties are used to shares partial of the secret key [32].

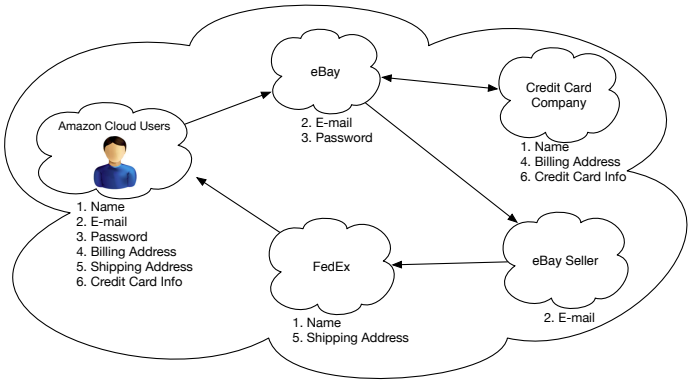

Figure 2.1: IDM Using Trusted Third Party

2.3.4 Secure Infrastructure

Besides trusted computing, infrastructure security is another important approach to address data privacy challenges. According to [14], the goals of infrastructure security are three-fold. First, it aims to ensure confidentiality and integrity of organization's data transition from public cloud providers. Second, it aims to ensure proper access control for resources sharing, such as what services do users want to obtain from clouds. Finally, it aims to replace tiers with domains and the established model of network zone.

In general, approaches to utilize trusted computing and secure infrastructure to protect data privacy by handling or even operating on sensitive data usually have high overhead and low scalability [21]. Therefore, secure infrastructure in trusted computing is infeasible for cost-efficient and large scale cloud computing.

2.4 Data Segregation Using Hybrid Cloud

Utilizing hybrid clouds to segregate data is also an approach to address data privacy issues. In this thesis, we propose to develop an integrated framework to support hybrid clouds, which includes public clouds, such as Amazon EC2, extensive capacity; private clouds, such as lab clusters, for higher security and ultimate control. Another approach to analogize the security idea to the concept of "do not put all your eggs in one basket" is that using hybrid clouds for different utilization can reduce some uncertainties and risks. Users are allowed to access different kind of services by harnessing the heterogeneous architecture.

Hybrid cloud provides an approach to split computations, by sending computations over non-sensitive data to the public cloud while retaining computations over sensitive data to the private cloud. Furthermore, MapReduce on hybrid clouds is designed for a single type of cloud without taking care of data with multiple security levels, but data is required to be partitioned manually depending on different resource allocation algorithms [21]. In addition, it needs to be supported by corresponding frameworks to handle data with mixed-sensitivity on hybrid clouds.

Obviously, there are many advantages to using hybrid clouds, in which cloud users can spread their risks, increase redundancy for each job to ensure high task completion probability for some important applications. However, potential issues including policy compatibility, data dependency and different data translations on hybrid clouds need to be considered as well. In general, it is a trade-off between redundancies and risks because if we spread our sensitive data across multiple clouds, consequently, this kind of redundancy increases exposure risks accordingly.

2.4.1 Airavat

Currently, for security protection in cloud computing, most research work focuses on virtual security, data storage security and data transmission security, while less work focus on security and privacy protection in a computation process. Airavat addresses data privacy challenges by setting a mathematical boundary for potential leakage and violations [33]. 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. As a result, cloud users still enable to do computations with mixed-sensitivity even if they do not have the data privacy awareness or experience. Based on differential privacy, Airavat ensures that the computation outputs should not violate the privacy of inputs [34]. Even though the influence - by adding random noise to the output - on computational accuracy is negligible, the risk of potential violations can be significantly reduced. Both trusted and untrusted mappers are supported by Airavat, however, only trusted reducers are supported, since the reducers have to comply with the enforcing privacy policies [33].

However, there are some limitations in Airavat. Airavat cannot confine all kinds of computations which are performed by untrusted code [33]. For instance, if untrusted mappers produce key pairs or value pairs in computations, data privacy cannot be surely guaranteed by Airavat. In this case, this mechanism leaves an opportunity for malicious mappers.

2.4.2 Sedic

Unlike Airavat, Sedic [35] 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 [35]. 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 so as to avoid leaking data sensitivity.

Sedic aims to assure high privacy which only allows non-sensitive data to be transmitted through public clouds. To maximize the utilization of public clouds, it allows the public cloud to take over the workload as much as possible while basing on developers' scheduling algorithms [35]. Besides, Sedic limits the inter-cloud transfer due to the high overhead. It provides a privacy-aware mechanism which extracts the combiner from the reducer such that the public cloud enables to process the data.

From users' perspective, they need to label sensitive data by utilizing Sedic data tagging tool [35]. This tool acts as a simple string scanner along with a string dictionary which contains sensitive keywords or patterns such as social security numbers, email addresses, home address, annually income, medical records, age, birthday, passwords, cell phone numbers, credit cards numbers and so on. When it pre-scans keywords or text patterns from a given dataset, once the keywords in the dataset are found, a sensitive label will be created and record the location and the path of the corresponding information [35]. Each file can be labeled individually for a dataset which contains multiple files. Then, all labeled records will be stored in the metadata file, and finally the metadata will be submitted along with the dataset to HDFS.

From Sedic's perspective, after analyzing reducers' source code, Sedic schedules the reduce tasks by minimizing internal cloud communications and by minimizing the workload on the private cloud [35]. Once the sensitive-tagging pre-processing completes, Sedic begins to partition and replicate the dataset depending on the sensitive labels. Then, it schedules mappers to the public or private cloud. Finally, it finalizes results from the public cloud and the private cloud and finishes reduction work on the private cloud. Sedic is designed for handling mixed-sensitivity data with sensitive labels, such that the sensitive data are required to be labeled before sending to the public cloud. Sedic enables MapReduce to be

more scalable on hybrid clouds as well as supporting traditional computations [35].

However, there are some shortcomings in Sedic. First, the flexibility limitations lead to not working well for complex data analytical tasks and practical applications [36]. Second, Sedic might still leak the locations and the length of the sensitive data [21]. Third, iterative MapReduce cannot be supported by Sedic [5]. Data restructure clusters the sensitive data from original blocks to new blocks. As a result, position movement of information for those blocks need to be recorded accordingly, more information about sensitive records needs to be processed. In this case, good performance cannot be achieved. Moreover, the code analysis and transformation techniques in Sedic only support this type of reducers [35].

2.4.3 Tagged-MapReduce

Tagged-MapReduce [21] addresses the data privacy challenges - to some extent - 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. Also, Tagged-MapReduce handles mixed-sensitivity data using hybrid clouds. As a result, the risk of leaking data privacy can be reduced for privacy-aware applications.

Tagged-MapReduce attaches a sensitivity tag on each key-value pair [21]. Scheduling policies are used to address data privacy as well as ensuring efficiency. In addition, Tagged-MapReduce allows applications to choose a default tagging policy to handle mixed-sensitivity data on hybrid clouds. Compared with Hadoop, Tagged-MapReduce provides better data privacy assurance on hybrid clouds with low overheads [21].

Tagged-MapReduce is a practical approach to address data sensitivity on hybrid clouds.

However, to some extent, one of their assumptions are not reasonable. They treat private cloud service which are actually located on the public cloud as their private cloud. But according to its definition, private clouds should be exclusively owned by users, therefore their data privacy concerns still exist.

Chapter 3

Privacy-Aware Hybrid Clouds

In this chapter, we present in great detail privacy-aware hybrid clouds, our framework for addressing the security and privacy challenges in clouds. Specifically, in Section 3.1, we introduce Aneka and MapReduce.NET, which are used in our framework to provide the infrastructure of hybrid clouds. We then describe our system architecture and configuration in Section 3.2. The design and implementation of the privacy-aware hybrid clouds framework are presented in Section 3.3. Finally, the optimization model is shown in Section 3.4.

3.1 Background: Aneka and MapReduce.NET

3.1.1 Aneka MapReduce.NET System Overview

MapReduce.NET is a master-worker based computational model, which supports MapReduce programming on the .NET framework. The architecture of MapReduce.NET contains 4 main components including a manager, a scheduler, an executor and a storage [20], as shown in Figure 3.1. Each client ($Client_0$, $Client_1$ ... $Client_n$) consists of three modules including a Client Process, a MapReduce Manager and an Aneka Message Handler. The manager component works as a supervisor for MapReduce.NET computations [20], which is

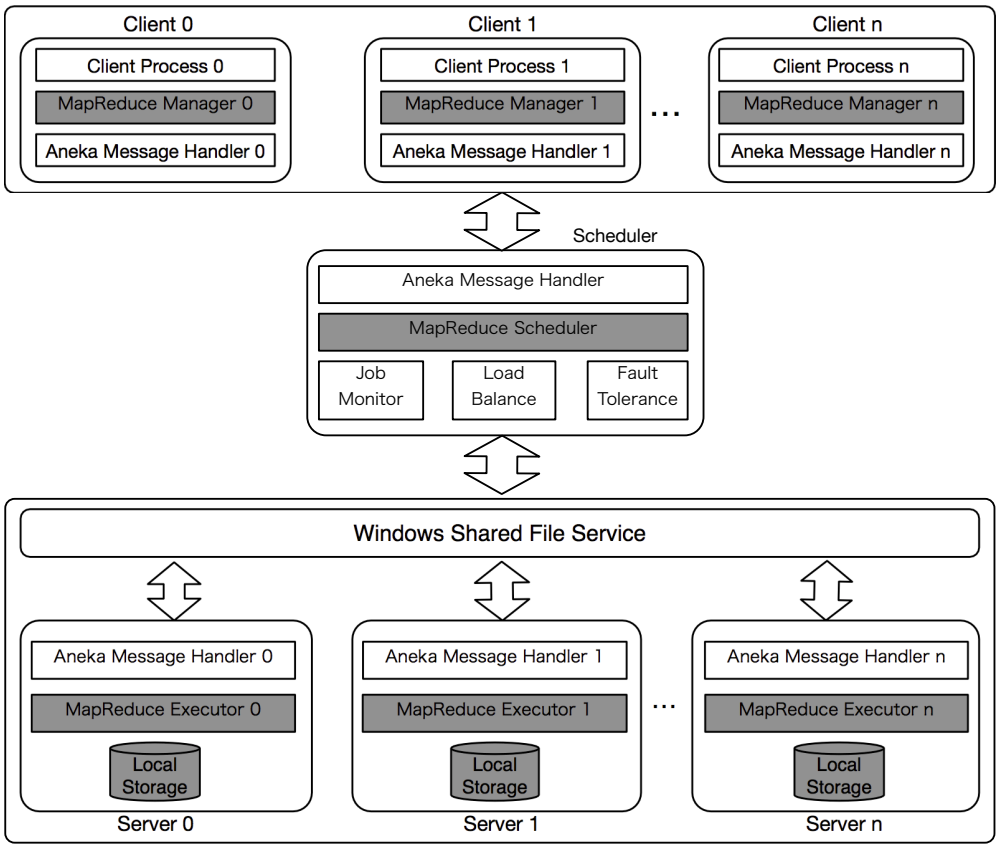

Figure 3.1: MapReduce.NET System View

responsible for submitting applications to the MapReduce Scheduler, and collecting the final results after the application completes.

The scheduler component contains a Aneka Message Handler, a MapReduce Scheduler, and three Monitors (Job Monitor, Load Balance Monitor and Fault Tolerance Monitor). When the Scheduler receives application tasks from the MapReduce Manager, it splits the tasks and assigns them to available server nodes ($Server_0$, $Server_1$ ... $Server_n$) [20]. The Scheduler monitors each task in this process. If some nodes cannot execute their tasks or show a much slower performance than others, the Scheduler will rebalance the workload through the task migration operations.

The executor component represents the execution of tasks. Every executor needs to wait

for being scheduled by the Scheduler in order to execute its task [20]. Usually, a Map task processes local input data. If the required input data is not local, the executor will fetch them from its neighbor nodes instead. A Reduce task cannot be scheduled until the executor obtains all input data from the mappers and merges them. In this case, the executor monitors the executing tasks, and reports their status to the scheduler at realtime. Each server node ($Server_0$, $Server_1$ ... $Server_n$) provides basic resources and services which are required for tasks execution. These includes: Aneka Message Handler Service, MapReduce Executor and Local Storage. Finally, all server nodes have access to a shared storage offered by Windows Shared File Service.

The storage component provides a distributed storage service for the .NET platforms, which contains two functions [20]. First, as a virtual storage pool, it manages disk storage on all available nodes. Second, it provides an object-oriented interface for users' applications.

3.1.2 MapReduce.NET Implementation

The implementation of MapReduce.NET is based on services provided by Aneka hybrid cloud platform. MapReduce.NET is designed for the .NET framework and Windows platforms. Similar to Google's MapReduce, it is an implementation of MapReduce programming model, which is widely used for processing large datasets with a simple parallel and distributed algorithm on clusters [5] [37].

MapReduce.NET provides object-oriented interfaces for Map() and Reduce() functions, which can be extended by developers for their applications [20]. A traditional MapReduce application works as follows. First, in the map phase, the input data is partitioned into a number of lists, each of which consists of key-value pairs. Then the lists of key-value pairs from phase 1 are assigned to a group mappers which are running concurrently. Finally, the reducers are triggered once for each key with associated values, and produce output values

as final results [20]. Figure 3.2 illustrates a big picture of MapReduce.NET from the users' perspective [38].

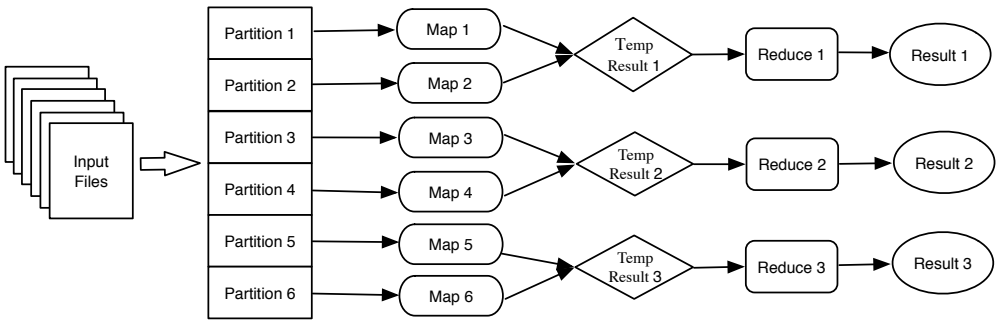

Figure 3.2: MapReduce.NET Computation Model

MapReduce.NET also provides corresponding APIs for programming data intensive applications. There are three basic components in a specific MapReduce.NET application: Mapper, Reducer, and MapReduceApplication [20]. Mapper is a base class in which users can specify their map function for filtering and sorting. Reducer is a base class in which users can specify their reduce function for summarizing. MapReduceApplication serves as a wrapper and entry point for the entire MapReduce.NET application. MapReduceApplication() is configured with Map and Reduce functions defined by users, and it is responsible for executing the map and reduce tasks, as well as collecting final results.

3.2 System Architecture and Configuration

Our framework is built on top of Aneka with MapReduce.NET. In this section, we describe in detail the architecture and configuration of our hybrid clouds. Specifically, we first illustrate our system architecture, then introduce the hardware and software settings on both the private cloud and the public cloud, as well as configurations on the Aneka platform.

3.2.1 Private Cloud Settings

CPU and Memory: The master container is responsible for scheduling jobs and keeping track of all workers in the network. We use a dual-core processor (2.7 GHz Intel Core i5) with a 8 GB memory and 500GB disk space for the master container node. Workers are responsible for executing jobs and a similarly capable machine would be ideal. In our system, we use the same configuration for worker container nodes.

Disk Space: In our system, we use disk space of 500GB for both the master container node and the worker container nodes.

Operating Systems: Our framework is built on Microsoft .Net Framework 2.0, and it can run on a variety of Windows Operating Systems, such as Windows 2000, XP, 2003, 2008, Vista, Windows 7, Windows 8 and Windows Server 2012 [6].

ECMA Runtime Environment: Aneka can be run on the Microsoft .Net Framework 2.0 or above, and the Mono Runtime 2.6 or above [6]. Also, it can be run on multiple Linux platforms based on Mono Runtime.

3.2.2 Public Cloud Settings

In this research, we use Amazon EC2 as our public cloud. The Amazon EC2 is an IaaS cloud computing service which provides large computing infrastructure and resources for users [7]. The service is elastic computing which enables users to extend or shrink infrastructure by launching or terminating instance.

Here, we choose *r3.large* instance type for windows based usage, region area is chosen to be *us-east-1*, and the network performance is moderate [39]. The r3.large here is Windows Server 2012 R2, 64-bit OS. Processor is Intel(R) Xeon CPU E5-2670 v2 @2.50GHz 2.49GHz.

RAM is 15.25GB. The resource characteristics of the r3.large for windows usage is described in Table 3.1 [39] [40].

API	vCPUs	Memory	ECUs	Storage	Cost
r3.large	2	15.25GB	6.5 units	32GB SSD	$\$0.1741/h$

Table 3.1: Amazon EC2 R3.Large Instance Description

3.2.3 Aneka Platform Setup

Aneka is a Cloud Application Development Platform for developing and running scientific computations and data-intensive applications [6]. Aneka supports 3 programming models that users can choose from: thread programming model, task programming model, and MapReduce programming model. Specifically, the task model in Aneka defines an application as a bag of tasks. Each task work independently and can be executed in any order by Aneka Scheduler [41]. The thread model in Aneka enables developers to virtualize multi-threaded applications in a transparent way [42]. The MapReduce model in Aneka abstracts computations by using a map function and a reduce function, such that any applications formulated in this way can be executed in parallel [20].

The Aneka platform also provides APIs for users to build new applications on the same cloud infrastructure. In this research, we use MapReduce APIs to build our own privacy-aware framework on the Aneka cloud infrastructure. Figure 3.3 illustrates the architecture a Aneka hybrid cloud [20] [38].

The Aneka platform can be installed in a few steps. To run it correctly with full features, it requires administrator privileges. Note that the corresponding inbound and outbound ports should be opened to allow two-way communication. After that, we can add resources to the lists with corresponding administrator credentials, and the repository can be shared through windows file sharing and FTP repository sharing.

The next step is to choose one of our machines in the list to be master container by

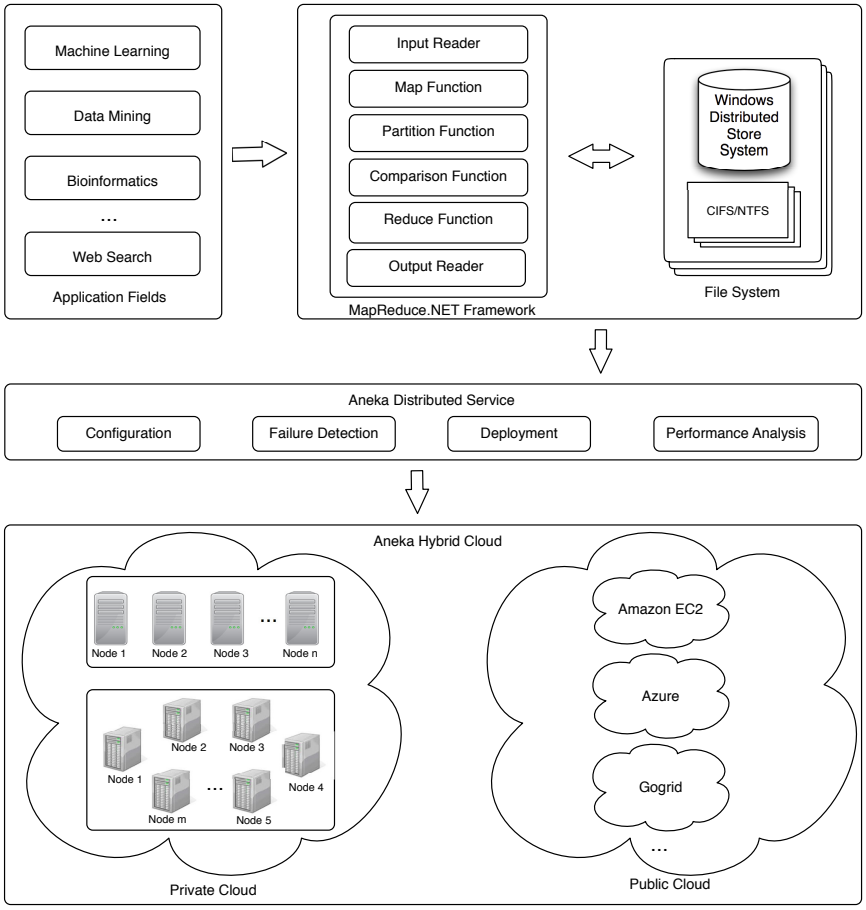

Figure 3.3: Aneka Hybrid Cloud

specifying its IP address and ports. The master container can be defined as the MapReduce scheduler. The remaining machines can be installed as worker containers by attaching the master container's IP address and ports. MapReduce executors can be also installed. Finally, our Anaka private cloud can be initialized.

To setup our Aneka public cloud, some prerequisite steps need to be taken. First we create a windows instance on Amazon EC2 which can be assigned to the default security group. More instances can be easily launched by cloning the image of the first one. Similar to the private cloud, more nodes can be added by specifying their IP addresses.

3.3 Privacy-Aware Computing

In this section, we describe how different levels of sensitivity are supported in our privacy-aware framework. These include a coarse-grained file level sensitivity, a fine-grained line level sensitivity and a dynamic run-time sensitivity.

3.3.1 File Level Sensitivity

In the first stage, we address the challenge of processing mixed sensitivity data on the file level. In this case, we add a boolean sensitivity tag to each file.

Files can be simply divided into two categories according to the tags when the system pre-processes them. Protection of sensitivity data has the highest priority, so sensitive files must be kept in the private cloud. For the non-sensitive files, we build a model to allocate the corresponding workload to either the private cloud or the public cloud. The workflow of this file level tagging mechanism is shown in Figure 3.4.

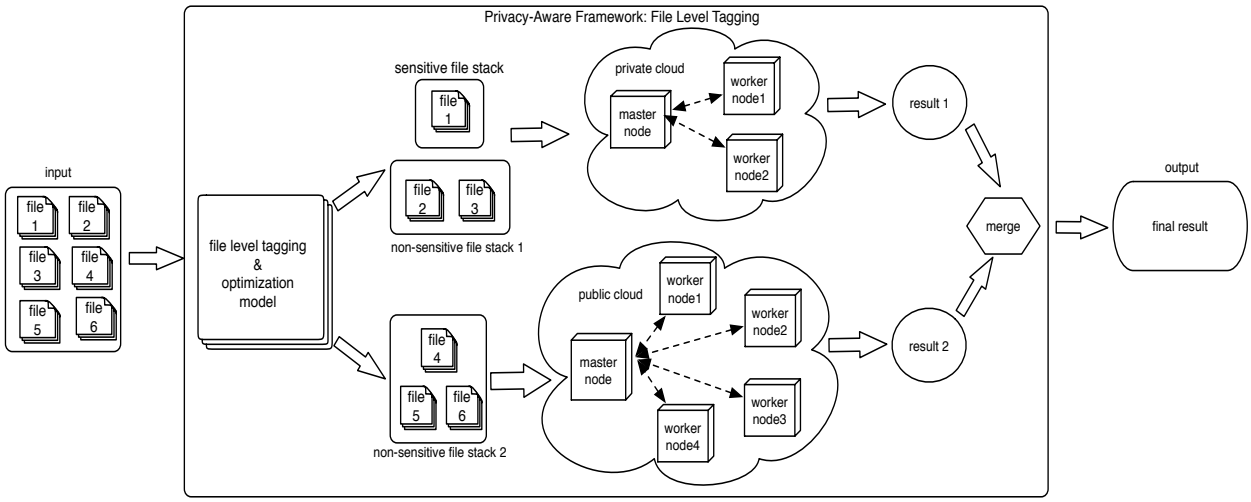

Figure 3.4: File Level Tagging Workflow

The execution of a MapReduce application can be separated into a map phase and a reduce phase. Each mapper obtains a (key1, value1) pair, and then produces a new pair

list(key2, value2). Each reducer obtains a key and a value list (key2, list[value2]) from the mappers. Then each mapper takes charge of summing up all of the values in the value list, that is list(key3, value3). The algorithm 1 shows how MapReduce computes the sum of the values associated with the same key.

Algorithm 1 Compute the sum of values associate with the same key

```
1: Mapper (string key1, integer value1)
2:     Emit (list(key2, value2))
3: Reducer (string key2, integer list[value2])
4:     sum ← 0
5:    For all value ∈ list[value2] do
6:        sum ← sum + value
7:    Emit (key3, sum)
```

The processing of non-sensitive data can be either on the public cloud or the private cloud, because it does not violate the expected security standard. However, it does affect the performance. Therefore, we have developed a performance optimization model to make such decisions about workload distribution. This model can be found in Section 3.4.

Once the workload is distributed, both public cloud an private cloud start to run MapReduce jobs. Each mapper is responsible for processing one single file. It groups all the pairs with the same key, and sum up the total amount for each distinct key. Each mapper generates a single immediate file. Once all mappers are done, the reducer will sort and merge all immediate result files to a single result file. The private result file will be put into the result folder, and the public result file will be sent back to the same result folder on the private cloud. Finally, the main process will be invoked to reduce the two result files to one final result file.

File Level Tagging Workflow on Private Cloud

The workflow on the private cloud can be divided into four steps, as shown in Figure 3.5. (1) Launch the private cloud service, read IP address from the master node on the public

cloud, then start the file-splitting thread to split files based on the configuration file and the optimization model. (2) Once the splitting completes, the private cloud starts to read contents from tag lists. If the file is tagged to be sensitive, it must be processed on the private cloud. Once the FileTransferPrivate function is triggered, the private cloud starts to run MapReduce jobs. (3) Public cloud monitors and receives input data from the private cloud. Once the files are received, the public cloud starts to run MapReduce jobs. Once it completes, it will return the result file back to the private cloud. (4) Finally, only after MapReduce jobs on both the private cloud and the public cloud complete, the reducer on the private cloud will reduce all the result files on both sides to generate the final result file.

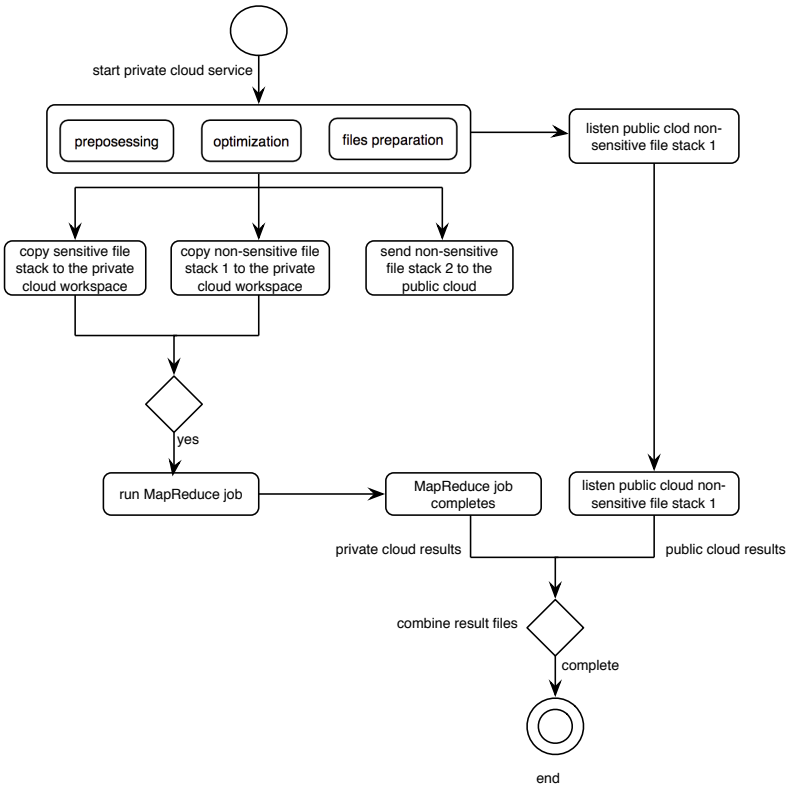

Figure 3.5: File Level Tagging Workflow on Private Cloud

File Level Tagging Workflow on Public Cloud

On the public cloud side, after launching the public cloud service, it starts to listen to connection requests from the public cloud. Once the master node on public cloud receives the correct files, it starts to run MapReduce tasks. It will return back results to the private cloud. The workflow on the public cloud is shown in Figure 3.6.

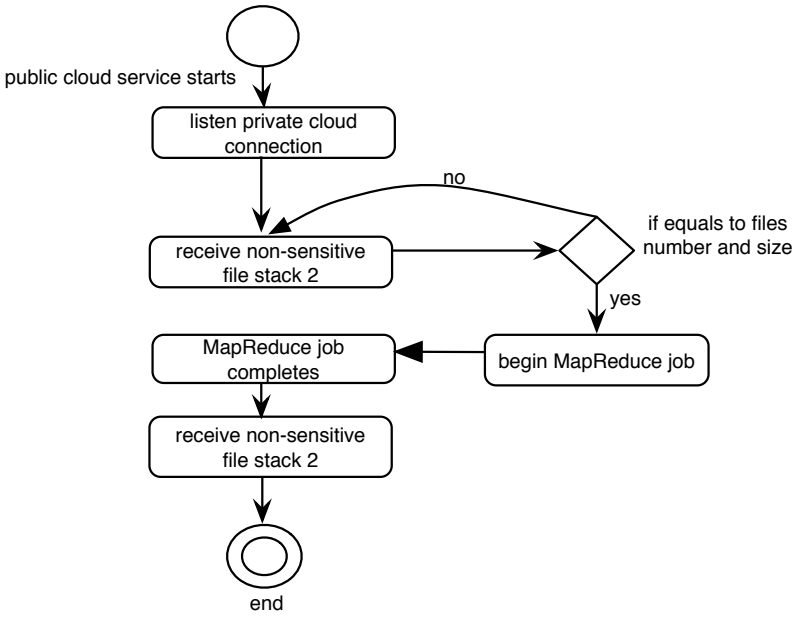

Figure 3.6: File Level Tagging Workflow on Public Cloud

3.3.2 Static Line Level Sensitivity

File level tagging is efficient and easy to implement. However in practice, files with mixed-sensitivity cannot be easily categorized to be sensitive or non-sensitive. Therefore, to address more practical problems, we propose a fine-grained tagging mechanism, tagging the mixed-sensitive data on the line level. In this case, we define that each single line in a file is either sensitive or non-sensitive.

The workflow on both private cloud and public cloud are similar to the file level tagging. However, static line level tagging enables protection of data sensitivity on a line basis. When

the private cloud completes file splitting, it will call ParseTagRomoteFile function, which keeps the sensitive data that is defined in the tag on the private cloud, and distributes the non-sensitive data to both public cloud and private cloud based on our optimization model in Section 3.4. The algorithm 2 presents the logic of tagging sensitive data and distributing non-sensitive data. Note that s is the data sensitivity rate, d is the public data size that will distribute on the private cloud, D is total data size, and $ratio = R_{pri}/R_{pub}$.

Algorithm 2 Logic of Tagging Sensitive Data and Distributing Non-sensitive Data

```
1: Tag (string key[ ], string tag[ ])
2:    Pre-scan all keys line by line
3:    For all key ∈ tag do
4:          split all lines and write to file PrivateFile
5:    For all key ∉ tag do
6:          split all lines and write to public files
7: Distribution (int s, int d, int ratio, int D)
8:    if  s ≤ ratio/(1+ratio)
9:       keep d sized public files to workspace on the private cloud
10:      send (D * (1 - s) - d) sized public files to workspace on the public cloud
11:   else
12:      send D * (1 - s) sized public files to workspace on the public cloud
```

In our approach, we tag the data on the static line level. The workflow of the static line level tagging is shown in Figure 3.7. Certainly, the highest priority is to keep the sensitive data on the private cloud. On the left side of the Figure 3.7, our system pre-scans data line by line, those lines which are marked as sensitive in their tags will be picked out and put into new private files, and the rest of non-sensitive data can be left in the original file. After tagging, based on our model, the non-sensitive data files can be partitioned into smaller files. The private cloud may computing a certain percentage of non-sensitive data while the public cloud will compute the rest non-sensitive data.

As Figure 3.7 illustrates, once the decision has been made, both public cloud and private cloud start to run their own MapReduce jobs. Similar to file level tagging, each mapper is responsible for processing one single file. It groups all the pairs with the same key, and sum

up the total amount for each distinct key. Each mapper will generate a single immediate result file.

Once all of the mappers are done, the reducer will sort and merge all immediate result files to a single result file. Once a MapReduce job on the private cloud is done, the results will be put into the result folder immediately. Also, once a MapReduce job on the public cloud is done, the results will be sent back to the private cloud in the corresponding result folder as well. Finally, the main process will be invoked to reduce the two result files to the final result file.

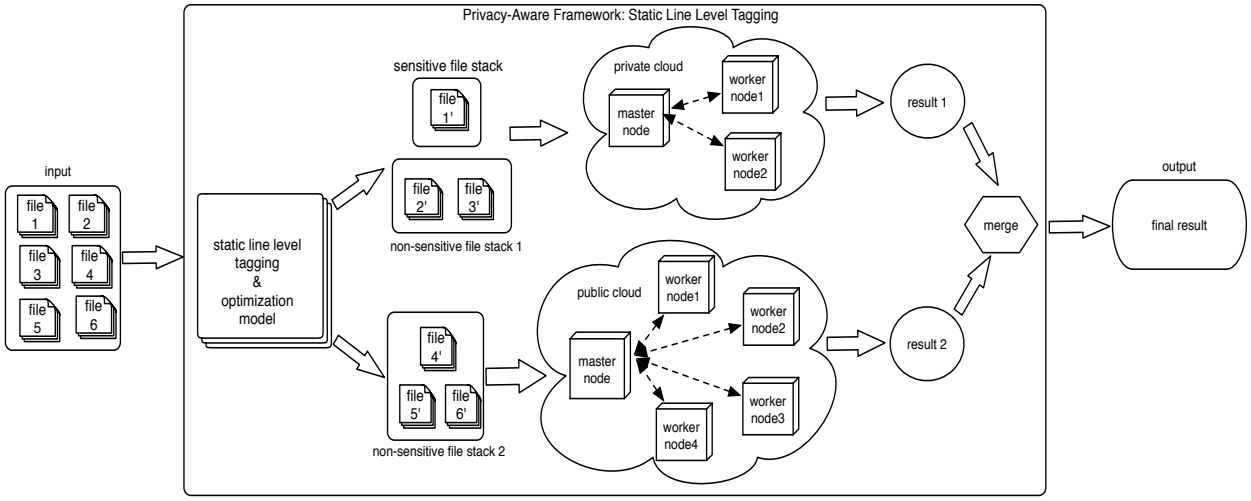

Figure 3.7: Static Line Level Tagging Workflow

3.3.3 Dynamic Line Level Sensitivity

Normally, the attributes of data such as the sensitivity can be changed with the time, since users might have a desire to protect their data at a certain moment, or vise versa. Therefore, we add the time dimension as a new constraint besides the privacy constraint and performance constraint, such that the "sensitivity over time" feature can be handled by privacy-aware framework. Initially each line in a file has a predefined sensitivity tag (either

sensitive or non-sensitive), and this tag can be changed over time. In the third stage, we tag mixed-sensitive data on a dynamic line level.

UDP socket is used here since the whole process is dynamic and large amount of data need to be transferred. Compared with TCP, UDP is not a directed connected protocol, which costs less resources and has higher transfer speed. Different from static line level tagging, new non-sensitive data can be added to the current tag policy and new sensitive data can be removed from the current tag policy as well. For instance, once the private cloud receives a request of adding new tagging lists, the new tagging list will be written to the tag immediately. Thus, the tag will base on the new tagging policies from the next round.

In practice, data are generated and are updated at run-time. It is not feasible to process data without considering the time dimension. We aim to handle dynamic computations from a dynamic perspective. Instead of applying static methods to process dynamic computations, we modify our privacy-aware framework to process large amount of data being generated at run-time.

Moreover, the sensitivity property can be redefined by users. For example, users might start with a dataset that they do not mind to make public, however, they may change their preference after more data are generated. In this case, using static data partition and computation will result in privacy violations, because the newly generated sensitive data could be sent to or scheduled to be processed by the public cloud. We add a mechanism to address this challenge by using new tag policies.

In our framework, the sensitivity tags are attached to data along timestamps. Assuming that at a certain moment, the user claims that certain lines related to some specific keys should be changed to be sensitive. Once the decision has made, these new tags will be added to our tag list, both the public and the private cloud will be notified immediately. The relevant data results will be sent back to the private cloud, and those input data will be deleted immediately. Thus in the next computing process, data tagging and splitting will be

based on the updated tagging list.

The whole architecture of privacy-aware framework using dynamic line level tagging is shown in Figure 3.8. On the left side of the Figure 3.8, data are produced dynamically over time, we tag data dynamically on line level accordingly. The highest priority is to keep the sensitive data on the private cloud. To this end, we pre-scan data line by line, the lines which are marked as sensitive will be extracted and put in new private files.

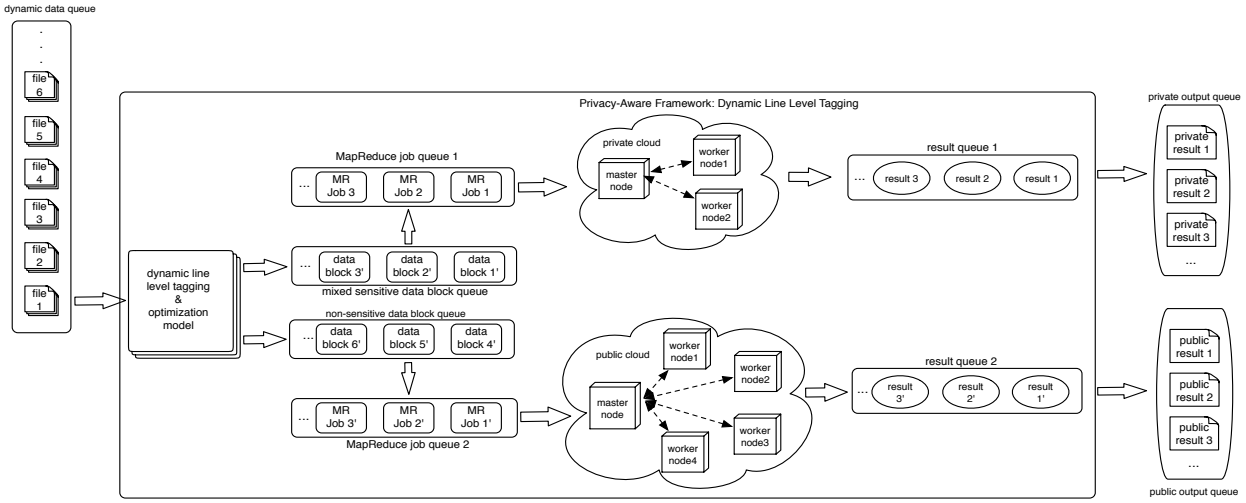

Figure 3.8: Dynamic Line Level Tagging Workflow

To simulate the dynamic data generation process, we use the dataset with 1GB as a base file, the system randomly pick one single line as a newly generated data unit at a time. When a new data file is generated, we check its tag. If it is sensitive, we must put it in the mixed-sensitive data files; if it is not sensitive, we can either put it in non-sensitive data files or mixed-sensitive data files. Once the file size reaches the block size, we split it to the data block ready queue. The data flows are illustrated in Figure 3.8.

Then, each private data block which contains mixed-sensitivity data is inserted in a MapReduce job queue on a FIFO basis. Similarly, each public data block with non-sensitivity data is inserted in another MapReduce job queue.

Note that better performance can be achieved if we could find an optimal data block size

for our framework. In an ideal case, the new data blocks should be available for processing when the current MapReduce tasks are completed. The evaluations of the optimal block data size in dynamic line level tagging can be found in Chapter 4.

On the right side of the Figure 3.8, once a MapReduce job on the private cloud is done, the results will be put into the results folder immediately. Also, once a MapReduce job on the public cloud is done, the results will be sent back to the corresponding result folder on the private cloud.

Dynamic line level tagging workflow on the private cloud

The workflow on the public cloud can be divided into six steps, which is shown in Figure 3.9. (1) Launch private cloud service, when a connection between private and public side build successfully, Communication and Data Generator service, MapReduce service will be initialized. Once the file reaches the data block size, it will be sent to the either data block ready queue 1 or queue 2 which based on our tag policy. (2) Start to run the first MapReduce job, and Data Generator service begins to randomly choose a single line from our base files at a time. (3) Once the MapReduce job finishes, it will release a semaphore named autoEvent, which enables private cloud side to go on running the new MapReduce job. The process will be repeated all the time until we stop the system service. (4) Simultaneously, the communication service monitors messages from the public cloud, once the connection is built, the private cloud will send the first data block in the ready queue 2 to the public cloud. (5) Transfer Private Manager on the private cloud monitors the result files from public cloud and store them in the corresponding directory. (6) The process will be repeated all the time until we stop either side of the system service. Once it stops, the Parse Result Manager will merge all the results files both from the private cloud and the public cloud to be one final result file.

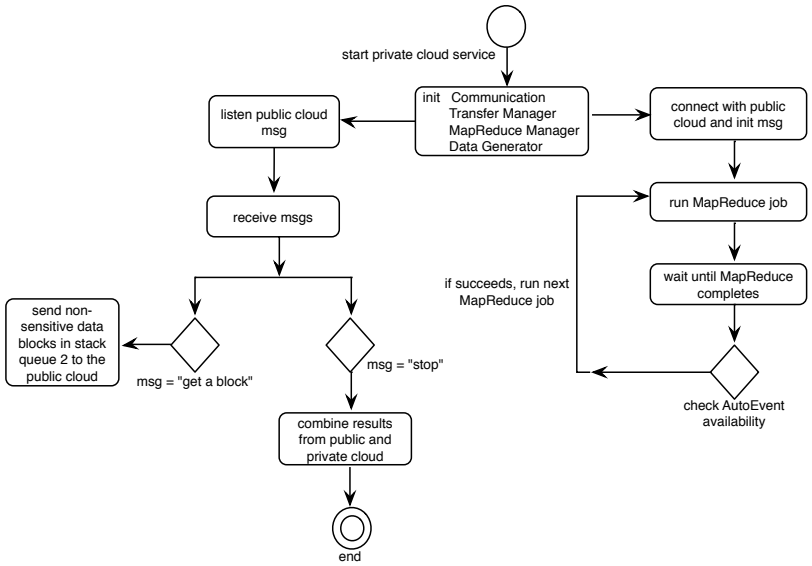

Figure 3.9: Dynamic Line Level Tagging Workflow on Private Cloud

Dynamic line level tagging workflow on the public cloud

The workflow on the public cloud can be divided into four steps, which is shown in Figure 3.10. (1) Launch public cloud service, when a connection between the private and the public side build successfully, Communication service and Data Generator service, MapReduce service will be initialized. (2) Transfer Manager service will monitor the messages from the private cloud. Once the first data block from the non-sensitive data block queue 2 has been received completely, the MapReduce service on public cloud will start to run the first MapReduce job. (3) Once the MapReduce job finishes, it will return back the result files to Transfer Private Manager on the private cloud. (4) The process will be repeated all the time until we stop either side of the system service.

3.4 Performance Optimization

Sensitive data is required to be processed on the private cloud, however, non-sensitive data can be processed either on the private cloud or on the public cloud. A better allocation of

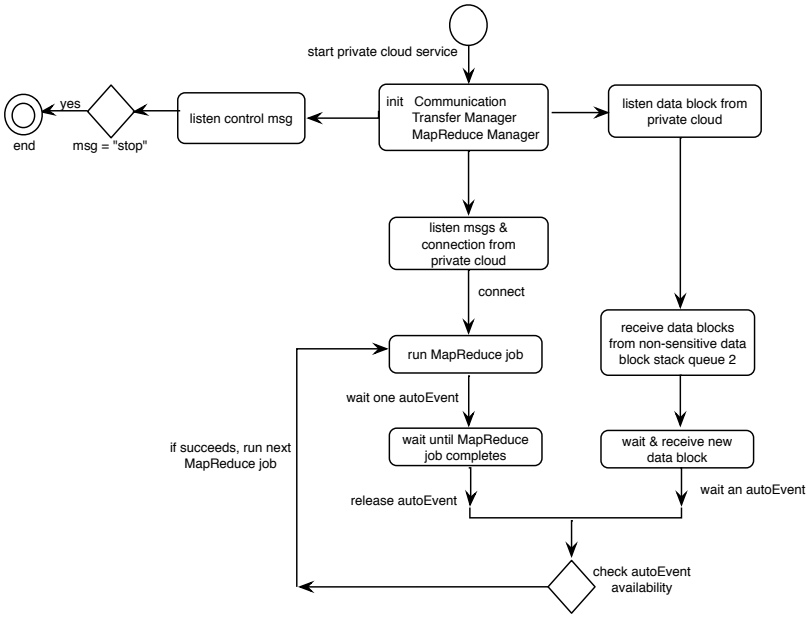

Figure 3.10: Dynamic Line Level Tagging Workflow on Public Cloud

non-sensitive data workload to the private cloud and to the public cloud can help to reduce system idle time so as to improve performance. Here, we try to optimize the performance by balancing workload between the public cloud and the private cloud. Table 3.2 outlines the parameter used in our performance optimization model:

Parameter	Description
D	total data size
d	public data size on private cloud
s	sensitivity rate $(0 \leq s \leq 1)$
R_{pri}	data processing rate on the private cloud
R_{pub}	data processing rate on the public cloud
$Wcomp_{pri}$	data size processed on the private cloud
$Wcomp_{pub}$	data size processed on the public cloud
T_{return}	communicational time for returning result from the public cloud to the private cloud

Table 3.2: Parameters Description

The total workload running on the private cloud is all sensitive data sizes and a fraction of non-sensitive data size. Thus the execution time for all workload running on the private

cloud should be:

$$Tcomp_{pri} = \frac{Wcomp_{pri}}{R_{pri}} = \frac{s * D + d}{R_{pri}} \tag{3.1}$$

The total workload running on the public cloud is part of non-sensitive data running on the public cloud, that is D - d - s * D. Thus the execution time for all workload running on the public cloud should be:

$$Tcomp_{pub} = \frac{Wcomp_{pub}}{R_{pub}} = \frac{D - d - s * D}{R_{pub}} \tag{3.2}$$

The total running time for the whole system should be counted as the one who completes its workload the latest. Thus the total running time should be:

$$Time = max(Tcomp_{pri}, (Tcomp_{pub} + T_{return})) \tag{3.3}$$

After given all of these above, the model for any computations run on our Aneka hybrid cloud can be treated as an optimization problem, which aims to satisfy privacy constraints the user gives and try to balance workload between private and public, as well as minimizing total time while protect data privacy. The formalization of the optimization problem is:
Minimize

$$T_{idle} = |Tcomp_{pri} - Tcomp_{pub}| \tag{3.4}$$

Subject to

$$\begin{cases} Time \geq 0, \\ Tcomp_{pri} \geq 0, \\ Tcomp_{pub} \geq 0, \\ 0 \leq s \leq 1, \\ 0 \leq d \leq D. \end{cases} \tag{3.5}$$

In practice, the feasible set indicates that d must be non-negative. The feasible set formed by the constraint set in Equation (3.4) is bounded. The constraint satisfaction aims to find at least one point in the feasible region. If there is no points to satisfy all given constraints, then the optimization problem is infeasible [43]. We aim to find the optimization solution such that $Tcomp_{pri} = Tcomp_{pub}$. Under all constraints described in (3.5), let $T_{idle} = 0$, then we have:

$$\frac{d}{D} = \frac{\frac{1-s}{R_{pub}} - \frac{s}{R_{pri}} + \frac{T_{return}}{D}}{\frac{1}{R_{pri}} + \frac{1}{R_{pub}}} \tag{3.6}$$

All of our experiments show that T_{return} is less than 0.3 second, thus T_{return}/D is negligible. Then the formula can be further simplified as:

$$\frac{d}{D} = \frac{\frac{1-s}{R_{pub}} - \frac{s}{R_{pri}}}{\frac{1}{R_{pri}} + \frac{1}{R_{pub}}} \tag{3.7}$$

Finally, let $ratio = R_{pri}/R_{pub}$, the final formula should be:

$$\frac{d}{D} = \frac{ratio}{1 + ratio} - s \tag{3.8}$$

Note that the Equation (3.8) works if $0 \leq d \leq D$. If $s \geq \frac{ratio}{1+ratio}$, we let $d = 0$.

Chapter 4

Evaluation

In this chapter, we first show our research objectives and our expectations on results, then we describe the dataset used in our experiments, and finally we present the experimental results for the three tagging mechanisms.

4.1 Evaluation Objectives

Our objective focuses on developing a framework for privacy-aware computing on hybrid clouds with mixed-sensitivity data, based on which:

- For a given computation with mixed-sensitivity data, process data using MapReduce.NET.
- For a given computation with mixed-sensitivity data, use public cloud Amazon EC2, to extent data processing capacity, while use private cloud to obtain higher security and ultimate control.
- For a given computation with mixed-sensitivity data generated at run-time, protect privacy of sensitive data by using dynamic-tagged mechanism.

4.2 Datasets and Application

The datasets we used in our experiments are downloaded from Center for Medicare & Medicaid Services, the official U.S. government site, which provides medicare supports, latest medicare enrollment, benefits and other information [44]. The dataset of CMS contains data records for the 2013 program year, which includes payment recipient information and identifying information from physicians or hospitals [44]. Our application aims to calculate the total payment amount for each company in the dataset.

(a). The data sensitivity features of records in the dataset are described by Table 4.1.

Dataset Name	**Sensitive Data**	**Non-sensitive Data**
CMS Medical Payment Records	Company Name $\in$ tag	Company Name $\notin$ tag

Table 4.1: Description of Dataset

(b). Related fields of each record in our dataset are listed in Table 4.2.

Field	**Type**	**Key**	**Tag Key**
General Transaction ID	integer	primary key	no
Company Name	string	unique	yes
Payment Amount	decimal	-	no
Payment Name	string	-	no
Hospital Name	string	-	no
Physician Name	string	-	no
Physician Specialty	string	-	no
Name of Associated Drug	string	-	no
Contextual Information	string	-	no
Recipient Address	string	-	no

Table 4.2: Description of Records

(c). All features of each job are described by Table 4.3.

Job Name	Operations	Descriptions
Startup	start(); connection().	Launch private cloud and public cloud; Build a connection between private and public cloud.

MapReduce	map(); reduce();
	sum(); count();
	sort(); merge().
		Sum up the total payment grouped by each company;
		Count the total transactions grouped by each company;
		Sort total payments for each companies in alphabet order.

File Tagging	tag(); scan(); keep(); send().	Tag every file to be either sensitive or non-sensitive; Scan all files names according to the tag items; Keep sensitive tagged files on private cloud; Keep certain amount of non-sensitive files on private cloud; Send the rest non-sensitive files to public cloud.

Static Tagging	tag(); scan();
	split(); keep();
	send(); receive().
		Tag every single line to be either sensitive or non-sensitive;
		Scan all lines in every files for a tag file;
		Intercept all sensitive tagged lines to private files;
		Split all other non-sensitive lines based on the model;
		Keep the private file on private cloud;
		Keep certain non-sensitive files on private cloud;
		Send the rest non-sensitive files to public cloud.

Dynamic Tagging	tag(); scan();
	trigger(); split();
	keep(); send();
	receive().
		Tag every single line to be either sensitive or non-sensitive;
		Scan all lines in each files for a given list in tag;
		Intercept all sensitive tagged lines to private files;
		When the non-sensitive data size reach the block size,
		keep the private file on private cloud;
		Accumulate every non-sensitive data block;
		Prepare non-sensitive data block for private cloud;
		Send next non-sensitive data block to the public cloud.

Table 4.3: Description of Jobs

4.3 Hybrid Cloud Setting

We build a hybrid cloud on Amazon EC2 and our lab machines. We treat our lab machines as our private cloud. The private cloud has 0 or 1 node at WSU Vancouver, running Windows 7, each provides 2.7 GHz Intel Core i5 Processor with a 8 GB RAM and 500GB disk space). The public cloud as 0, 1 or 2 r3.large instances on Amazon EC2 us-east-1, each runs Windows 2012 Server R2, and provides Intel(R) Xeon CPU E5-2670 v2 @2.50GHz 2.49GHz, 6.5 unites of ECUs, 15.25GB memory and 32GB SSD storage [39]. Network performance on r3.large is moderate, and the cost for each instance is $0.1741/h [40].

4.4 Experimental Results

4.4.1 File Level Tagging

To evaluate the performance of the file level tagging approach, we split the original dataset into some fixed sized sub-datasets, which the data size ranges from 100MB to 1000MB. According to algorithm 1 in Section 3.3, the key here is "Company Name", and value is the "Payment Amount", we use MapReduce in algorithm 1 to compute the sum of "Payment Amount" associated with the same "Company Name".

To begin with, the data processing rate on both public cloud and private cloud can be obtained by running a 1000MB test case without tagging (0% sensitivity). We find out that usually, the data processing rate ratio between the private cloud and public cloud is 0.96. In order to balance the workload on both public cloud and private cloud, resources can be fully utilized, so as to minimize the idle time for either side of the cloud. Here, based on our performance optimization model in Section 3.4, the computing workload which distributes to the public cloud should be 1.041 times of workload which distributes to the private cloud.

To simplify the computing workload allocation strategy, the system allocates files to the nearest whole number. Specifically, we split the non-sensitive file to 100 equivalent sub-files, the number of non-sensitive files need to be processed by the private cloud is 49 minus the number of sensitive files. And the number of non-sensitive files need to be processed by the public cloud is 51.

In Figure 4.1, all cases with different sensitivity rates show that the performance increases almost slowly as the data size increases. MapReduce is not efficient for small computation tasks, instead, it enables to have better performance when the computation size becomes larger. For example, the performance of data size with 100MB does not much smaller than other data sizes. Using our hybrid cloud is more efficient than private only performance. The performance of data size with 0%, 5%, 25%, 50%, 75% and 100% sensitivity rates are

much smaller than running on the private cloud.

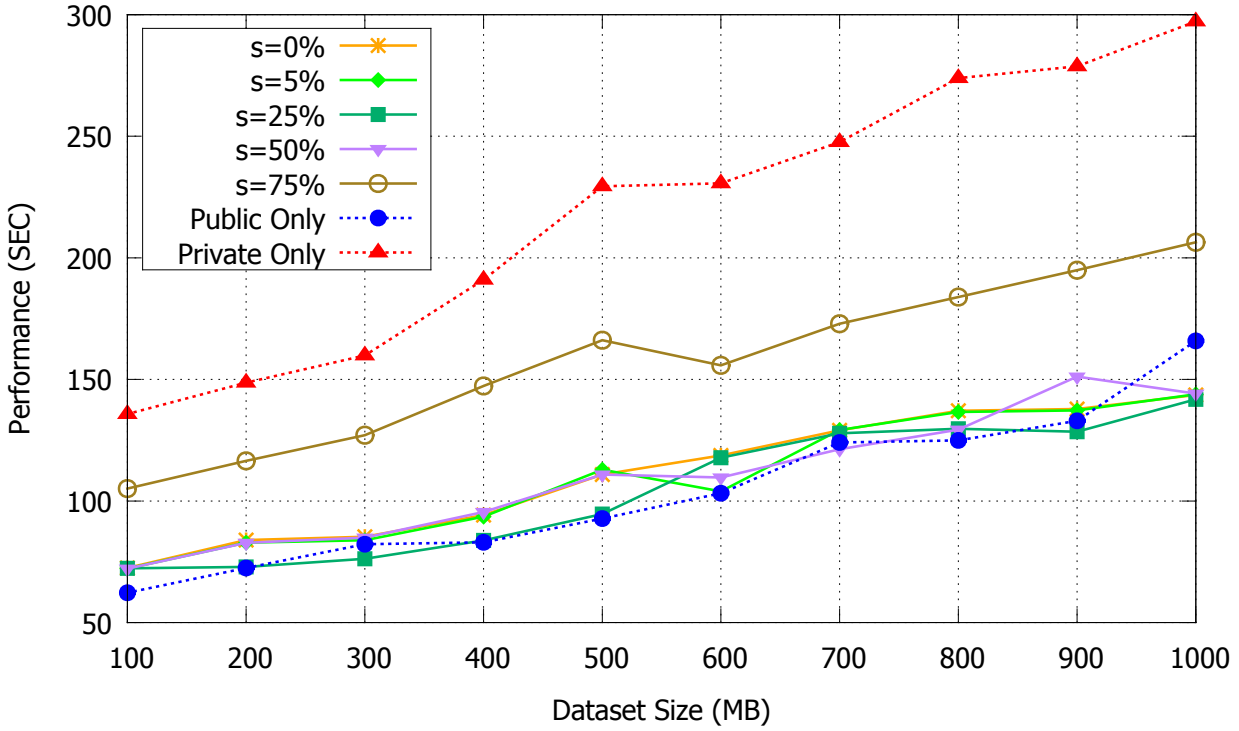

Figure 4.1: File Level Tagging: Performance Comparison for Different Sensitivity Levels

On the file level tagging experiments, the sensitivity rate is the percentage of sensitive data size divides total data size. We test performance with different sensitivity rates, all cases in our experiments indicate that the performance increases as the sensitivity rate increases. As Figure 4.1 illustrates, when the sensitivity rate belows 50%, the file-level tagging approach has little impact on the performance. However, when the sensitivity rate reaches 75%, the performance increase sharply. In comparison, however, it is still better than the case of running only on the private cloud.

Even though the performance of sensitivity rate varies from 0% to 75% is little worse than only using the public cloud, when the sensitivity rate is below 50%, the performance does not decrease that much. Moreover, running computations only on the public cloud cannot guarantee any data privacy, or it can be only limited in processing non-sensitivity

data. In addition, our hybrid clouds have much better performance than only running on the private cloud. Therefore, the results above elaborates that our file level tagging on the hybrid cloud can guarantee data privacy as well as ensuring relatively high performance.

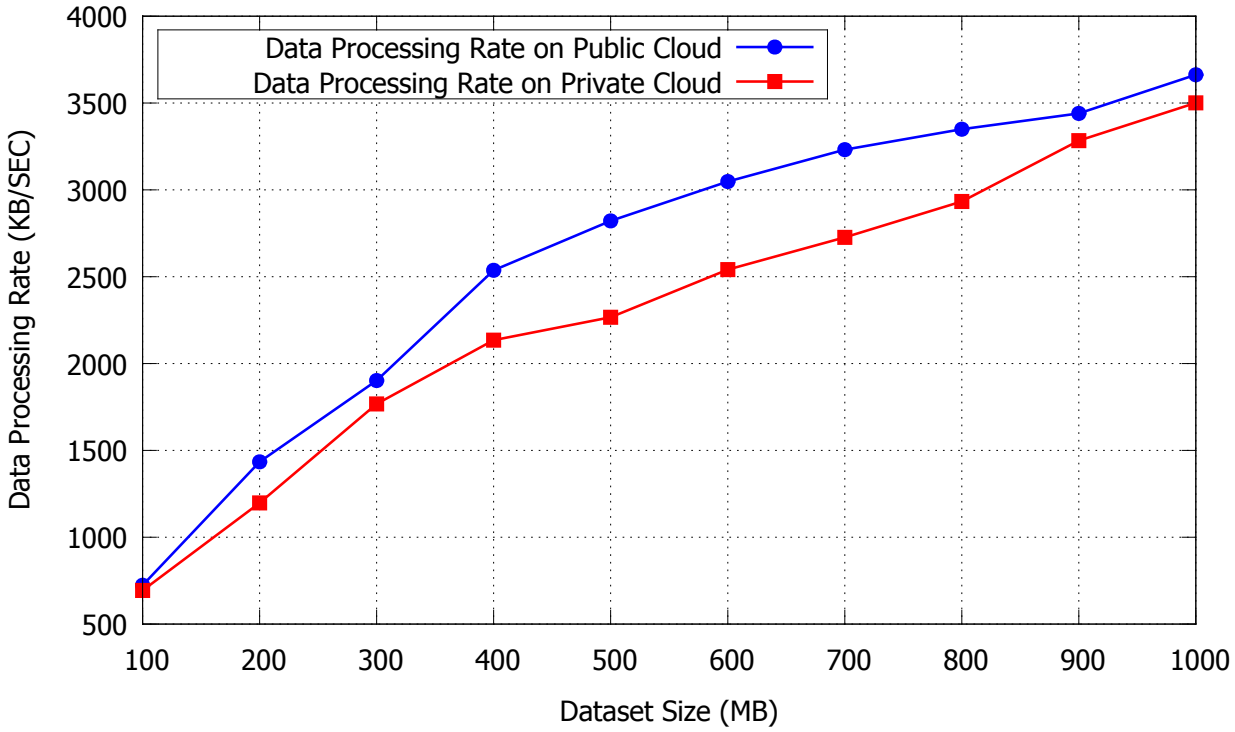

Figure 4.2: File Level Tagging: Data Processing Rate Comparison

As Figure 4.2 shows, the data processing rate of the public cloud is faster than the private cloud. Their data processing rate growth tendency is almost in parallel. The ratio of data processing rate between private cloud and public cloud is relatively stable.

In our performance optimization model, we use one small data size to test their data processing rate, then we can get the ratio of data processing rate between private cloud and public cloud. Figure 4.2 shows that our method of obtaining approximation ratio is acceptable. The ratio of data processing rate between private cloud and public cloud is 0.96. As Figure 4.3 shows, the scatter distribution of the ratio of data processing rate between private cloud and public cloud can be approximately represented by a horizontal line which

shows that ratio = 0.91 which is very close to our model.

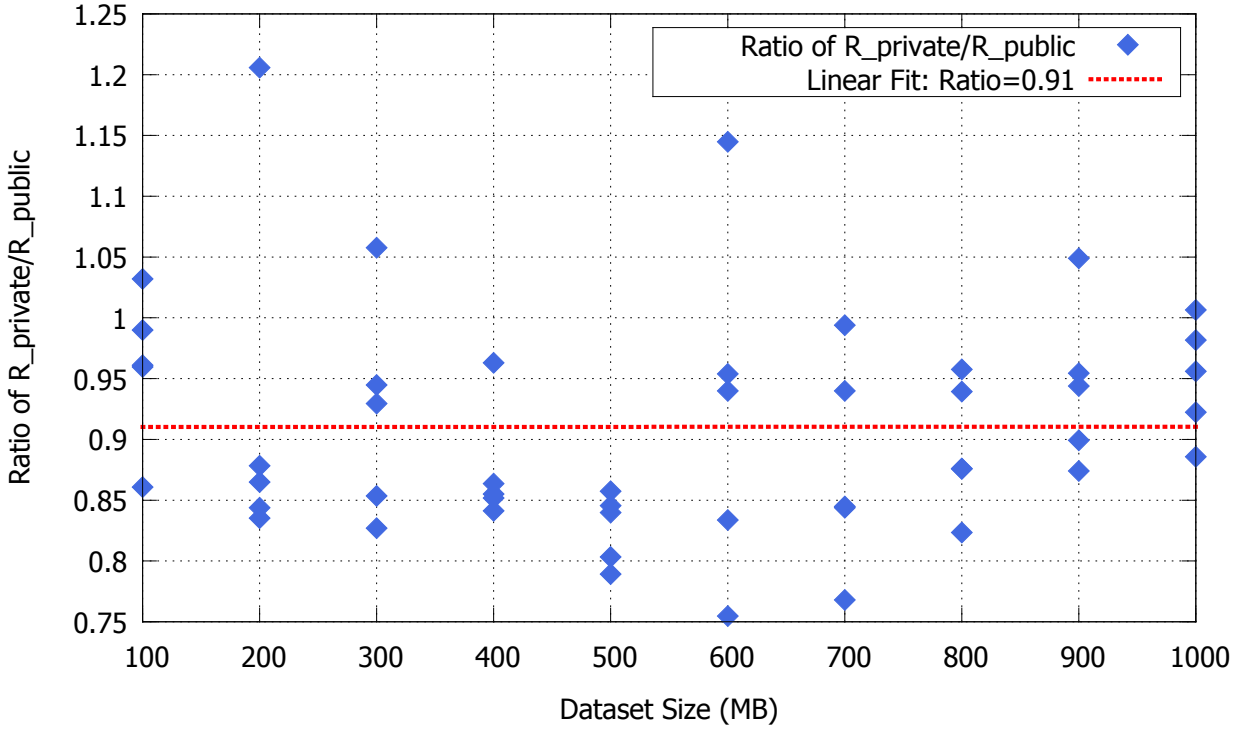

Figure 4.3: File Level Tagging: Scatter Plot of $R_{private}/R_{public}$

4.4.2 Static Line Level Tagging

To evaluate the performance of our static line level tagging approach, we tag the third row "Company Name" as the sensitive key. In our dataset, for every "Company Name" has many records. Thus the sensitivity rate here is on the "Company Name" basis, the sensitivity rate is defined as the percentage of the number of "Company Name" we tagged sensitive in the total number of distinct "Company Name".

We evaluate our static line level tagging approach in a similar way. As Figure 4.4 illustrates, our static line level tagging on hybrid clouds obtain much better performance than running on the private cloud only, which proves that our hybrid cloud enables to handle these computation well if they have a deadline to get it finished. In comparison, Figure 4.4 shows

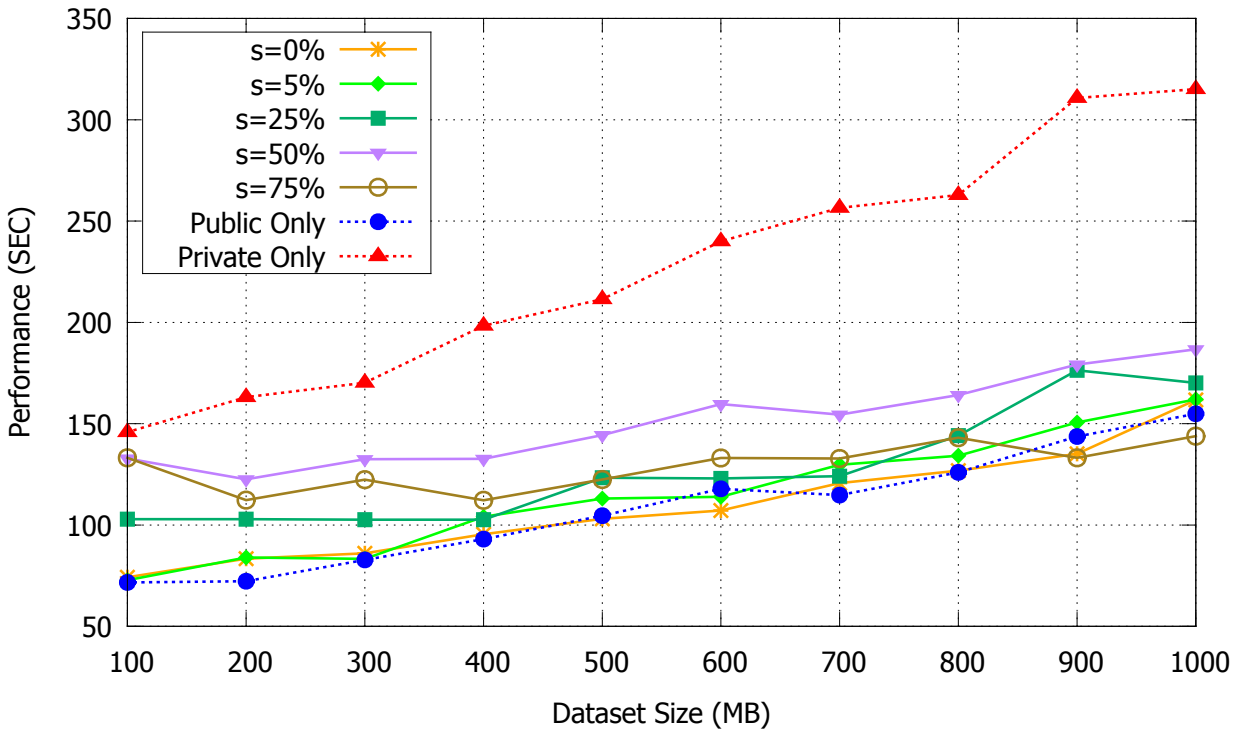

Figure 4.4: Static Line Level Tagging: Performance Comparison for Different Sensitivity Levels

that running these computations only one public could reach better performance. However, the same problem still exits such that data privacy cannot be guaranteed. Moreover, running only on the private cloud can guarantee data privacy, but it takes much longer time to finish some large computation tasks.

In addition, the execution time increases almost linearly as the data size increases. Also, the execution time always increases when the sensitivity rate increases from 0% to 50%. However, when the sensitivity rate is the 75%, it out performs the one with 50% sensitivity rate. This is the only exceptional case in our experiments. Since the sensitivity rate is based on the number of distinct "Company Name", the number of records for each "Company Name" is not uniform. The total number of lines for some "Company Name" is much larger than others. Thus, the sensitivity rate cannot directly represent the sensitivity data size.

Especially, this issue is more obviously for small size of datasets. However, when the dataset becomes bigger, the sensitivity rate trends to be pretty close the percentage of sensitive data size in the dataset.

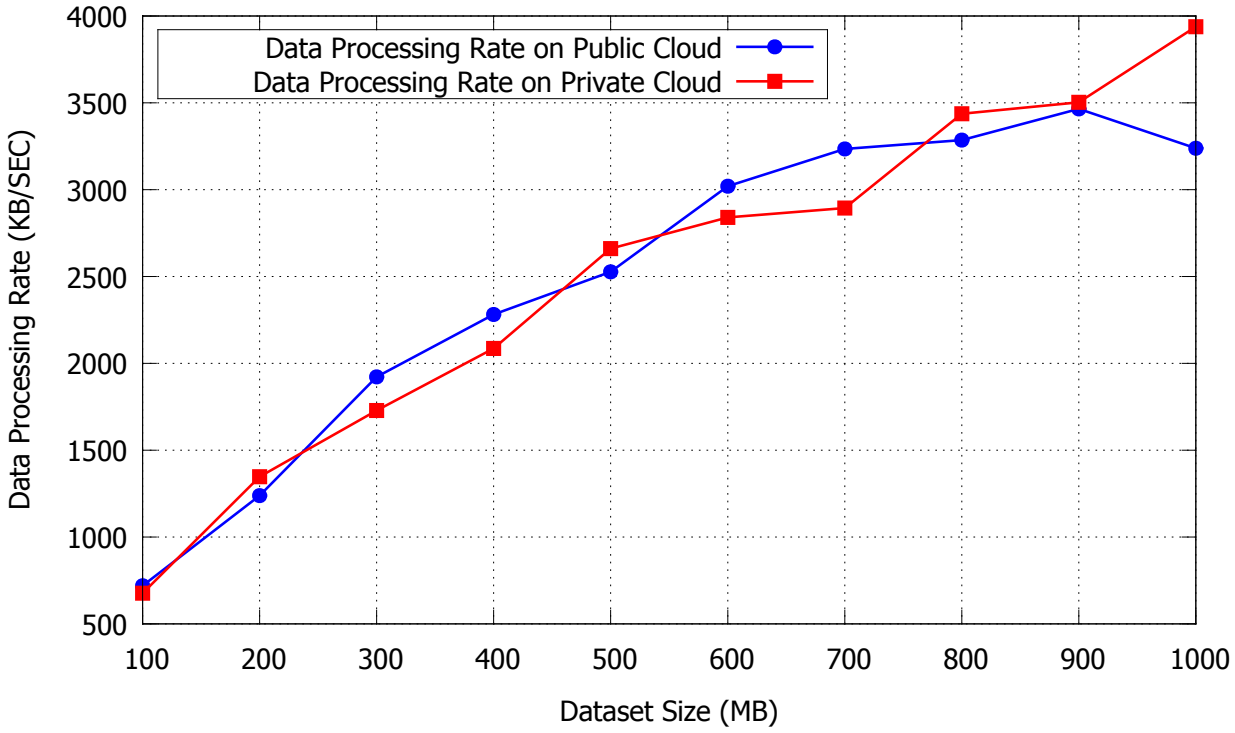

Figure 4.5: Static Line Level Tagging: Data Processing Rate Comparison

In our static line level tagging, we manipulate the ratio of data processing rate between private cloud and public cloud our performance optimization model. As Figure 4.5 shows, the data processing rate of the public cloud almost has the same increasing trend with the private cloud. Their data processing rate growth tendency can be treated as parallelism. According to Figure 4.5, the ratio of data processing rate is relatively stable. To some extent, the method of obtaining approximation ratio in our model is feasible in this approach.

4.4.3 Dynamic Line Level Tagging

To evaluate the performance of our dynamic line level tagging approach, we tag the third row "Company Name" as the sensitive key. Each data record is randomly generated based on the same 1.0GB dataset. In our experiments, we use 10MB, 50MB, 100MB, 150MB, 200MB, 250MB, 300MB, 350MB, 400MB, 450MB and 500MB as the data block size respectively. Since in our dynamic line level tagging in our system is a real-time process. To evaluate the relation between sensitivity rate and performance. In each test, we count the performance of running 10 continuous MapReduce jobs on the private cloud, and a certain number of MapReduce jobs that are completed on the public cloud during this time period. We count the performance in this time period except the system start-up time. Therefore, the data processing rate can be calculated as following:
(number of jobs completed on the public cloud + 10) * (data block size) / (finishing time - starting time)

In Figure 4.6, in general, the data processing rate has a positive correlation with the sensitivity rate. Furthermore, when the data block size is small, the data processing rate increases sharply in the beginning. For all sensitivity rate cases, Figure 4.7 shows that the the data processing rate reaches at a relatively high rate than others when the data block size locates in [150MB, 250MB]. The system has more parallelism when the data block size is in that range.

In addition, dynamically generated data with mixed-sensitivity can be handled by dynamic line level tagging mechanism, tags can be added or removed at run-time. To evaluate this, we carry out a set of experiments by fixing data block size as 200MB. Then we count the performance of running 10 continuous MapReduce jobs on the private cloud, and a certain number of MapReduce jobs that are completed on the public cloud during this time period. In each time period, we dynamically add new keys (keys that are not tagged as sensitive

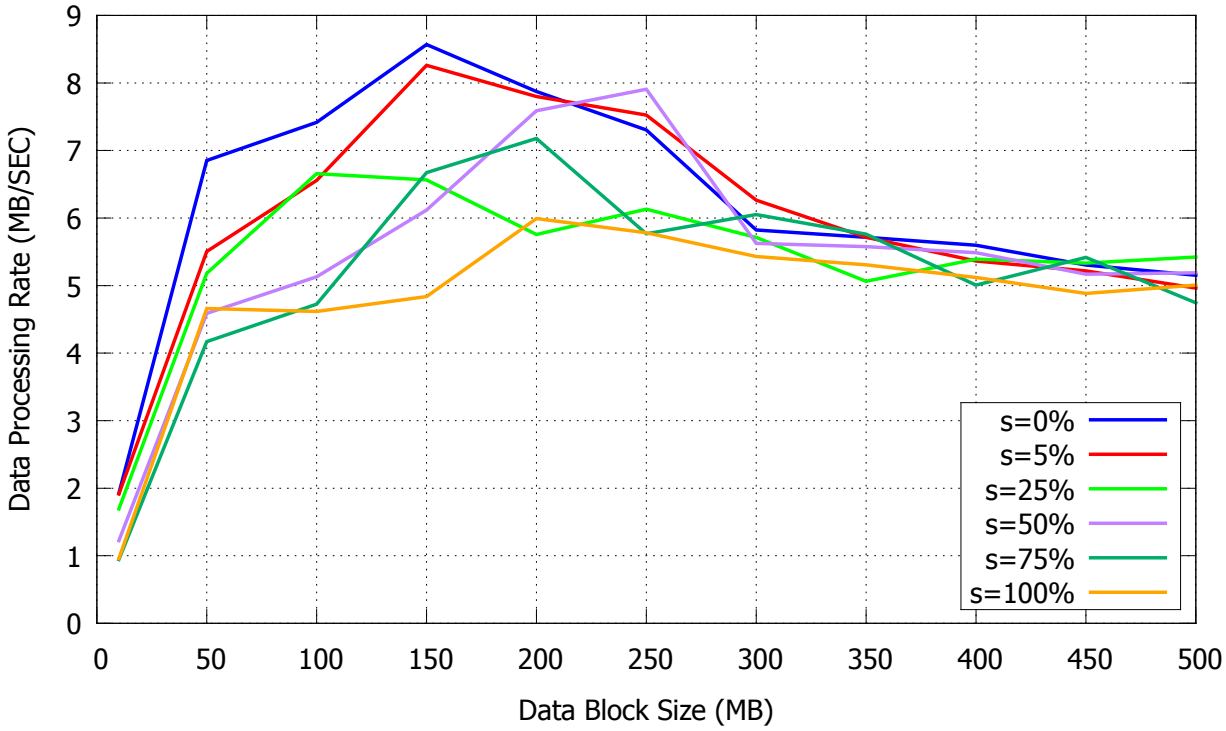

Figure 4.6: Dynamic Line Level Tagging: Data Processing Rate vs. Data Block Size

before) to the tag. The sensitivity level ranges from 0% to 100%. As the results show in Figure 4.7, the data processing rate decreases at the beginning, but it becomes relatively stable after the sensitivity rate is above 30%, which demonstrates that our framework has good scalability in terms of sensitivity level at run-time.

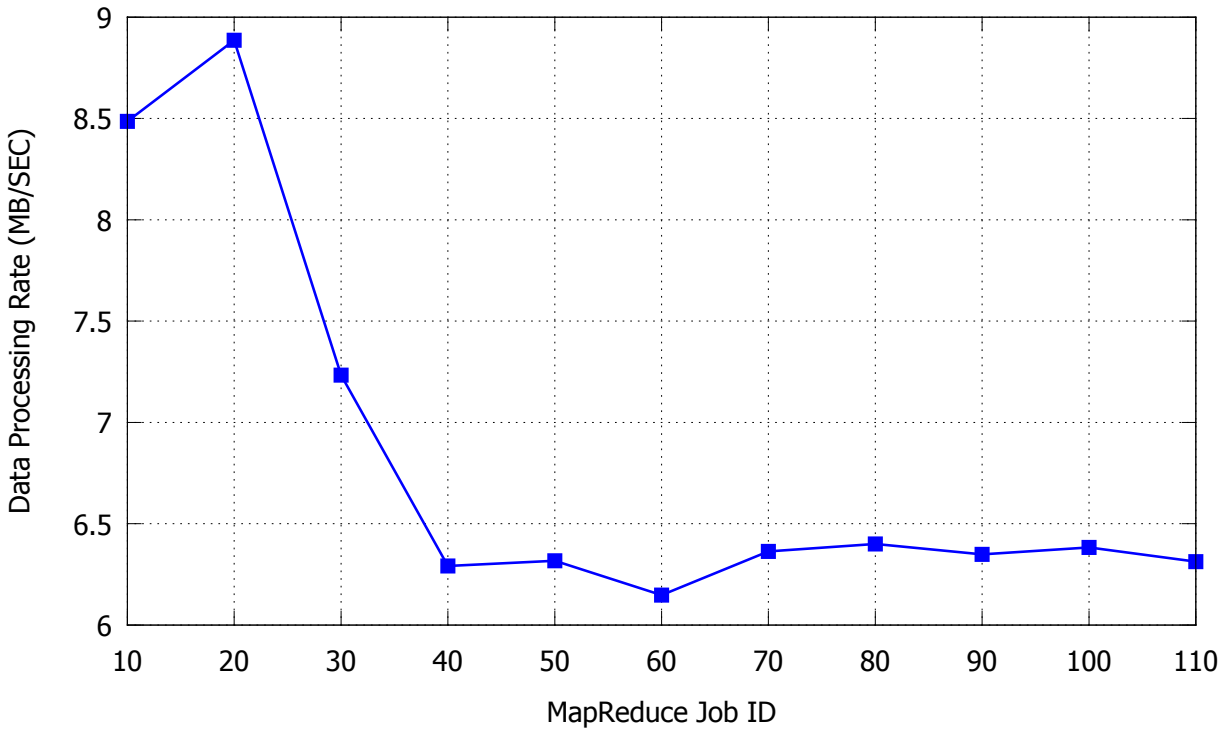

Figure 4.7: Dynamic Line Level Tagging: Date Processing Rate vs. Scalability of Sensitivity Levels

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have proposed a framework for privacy-aware computing on Aneka hybrid clouds with mixed-sensitivity data. The implementation is based on MapReduce.NET programming model. Our performance optimization model can be used to allocate computation workloads on our hybrid clouds efficiently. It also can be used as a baseline to help users make decisions about the cost, time, and resource configuration.

The design and implementation are divided into three stages, which includes file level tagging, static line level tagging, and dynamic line level tagging. File level tagging aims to process data with a high rate of sensitivity, which is simple and straightforward. Static line level tagging aims to process data on a line level basis, which is efficient to handle data with a relatively light or medium sensitivity. Dynamic line level tagging enables to dynamically process data with time, and also addresses challenges of redefining data sensitivity. Our framework is scalable in terms of sensitivity level. Moreover, our experiments using dynamic line level tagging shows that the system can reach a relatively higher data processing rate when the data block size locates in [150MB, 250MB].

In conclusion, our privacy-aware framework improves the performance of data computation by using the public cloud, as well as protecting data privacy by segregating sensitive data from disclosure to the public. The challenges of handling data with mixed-sensitivity can be addressed.

5.2 Future Work

In the future, our current framework can be generalized to accommodate other types of computations such as social network applications with mixed-sensitivity data. Beside of MapReduce programming model, we plan to extend our privacy-aware framework to support task programming model and thread programming model so that more diverse data computations can be applied. In addition, more resource allocation policies can be developed basing on user's requirements.

Moreover, adding some resource provisioning mechanisms into our current framework will be more promising. Provisioning resource pool can be dynamically shrinked or extended by using more flexible resource allocation strategies. Our private cloud is still a static resource pool, however, public clouds such as Amazon EC2 will be on a dynamic resource basis. On the one side, more EC2 instances can be launched and added to our public cloud resource containers provisionally. Once the current hybrid clouds still unable to handle the overloaded computations, or there is a deadline for completing computations on some big datasets. On the other side, current EC2 instances on the public cloud can temporarily shut down to stop the service, then those resources will be temporarily removed from our public cloud resource pool. In this case, performance can be ensured for some deadline-driven computation tasks. The cost can be saved for small computation task who do not have a high desire for performance. Therefore, more flexibilities can be achieved by using our framework.

Bibliography

[1] L. Ran, S. Helal, and S. Moore, "Drishti: An integrated indoor/outdoor blind navigation system and service," in *Proceedings of the Second IEEE International Conference on Pervasive Computing and Communications (PerCom'04)*, ser. PERCOM '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 23–. [Online]. Available: http://dl.acm.org/citation.cfm?id=977406.978696

[2] C. I. E. C. Center. (2011) Inventory of global top ten leaks in 2010. [Online]. Available: http://www.100ec.cn/detail–5594360.html

[3] J. Kirk. (2015) Google error leaks owner personal info for nearly 300,000 websites. [Online]. Available: http://www.pcworld.com/article/2896472/google-error-leaks-website-owners-personal-information.html

[4] 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

[5] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008. [Online]. Available: http://doi.acm.org/10.1145/1327452.1327492

[6] M. P. Ltd. (2013) Aneka installation guide aneka 3.0. [Online]. Available: http://www.manjrasoft.com/download/3.0/AnekaInstallationGuide.pdf

[7] A. Iosup, S. Ostermann, N. Yigitbasi, R. Prodan, T. Fahringer, and D. Epema, "Performance analysis of cloud computing services for many-tasks scientific computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 6, pp. 931–945, Jun. 2011. [Online]. Available: http://dx.doi.org/10.1109/TPDS.2011.66

[8] Wikipedia. (2015) Cloud computing. [Online]. Available: http://en.wikipedia.org/wiki/Cloud_computing

[9] Interoute. (2015) What is iaas? [Online]. Available: http://www.interoute.com/what-iaas

[10] C. Taxonomy. (2010) Platform as a service. [Online]. Available: http://cloudtaxonomy.opencrowd.com/taxonomy/platform-as-a-service

[11] CT. (2011) Software as a service. [Online]. Available: http://cloudtaxonomy.opencrowd.com/taxonomy/software-as-a-service

[12] J. Ames. (2013) Types of cloud computing: Private, public and hybrid clouds. [Online]. Available: http://blog.appcore.com/blog/bid/167543/Types-of-Cloud-Computing-Private-Public-and-Hybrid-Clouds

[13] I. C. Ltd. (2014) Cloud articles. [Online]. Available: http://www.interoute.com/cloud-article/what-private-cloud

[14] P. B. y Rita, M. E. Tyler, and K. A. Kaczmarek, "Seeing with the brain," *International Journal of Human-Computer Interaction*, vol. 15, no. 2, pp. 285–295, 2003. [Online]. Available: http://dx.doi.org/10.1207/S15327590IJHC1502_6

[15] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Gener. Comput. Syst.*, vol. 28, no. 3, pp. 583–592, Mar. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.future.2010.12.006

[16] M. C. Schatz, "Cloudburst: highly sensitive read mapping with mapreduce," *Bioinformatics*, vol. 25, no. 11, pp. 1363–1369, 2009. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/btp236

[17] B. Langmead, M. Schatz, J. Lin, M. Pop, and S. Salzberg, "Searching for snps with cloud computing," *Genome Biology*, vol. 10, no. 11, p. R134, 2009. [Online]. Available: http://genomebiology.com/2009/10/11/R134

[18] T. A. S. Foundation. (2015) Welcome to apache hadoop! [Online]. Available: http://hadoop.apache.org/index.pdf

[19] Wikipedia. (2015) Mapreduce. [Online]. Available: http://en.wikipedia.org/wiki/MapReduce#Map_function

[20] Manjrasoft. (2012) Developing mapreduce.net applications aneka 3.0. [Online]. Available: http://www.manjrasoft.com/download/3.0/MapReduceModel.pdf

[21] 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. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6846438

[22] 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. [Online]. Available: http://doi.acm.org/10.1145/1005140.1005147

[23] 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–. [Online]. Available: http://dl.acm.org/citation.cfm?id=882494.884426

[24] 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. 14–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1929820.1929839

[25] 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. [Online]. Available: http://doi.acm.org/10.1145/1536414.1536440

[26] P. Angin, B. Bhargava, R. Ranchal, N. Singh, M. Linderman, L. Othmane, and L. Lilien, "An entity-centric approach for privacy and identity management in cloud computing," in *Reliable Distributed Systems, 2010 29th IEEE Symposium on*, Oct 2010, pp. 177–183. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5623390

[27] M. Zhou, R. Zhang, W. Xie, W. Qian, and A. Zhou, "Security and privacy in cloud computing: A survey," in *Proceedings of the 2010 Sixth International Conference on Semantics, Knowledge and Grids*, ser. SKG '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 105–112. [Online]. Available: http://dx.doi.org/10.1109/SKG.2010.19

[28] M. Rouse. (2013) identity management. [Online]. Available: http://searchsecurity.techtarget.com/definition/identity-management-ID-management

[29] I. B. Justus. Identity management series role- and rule-basing part 1: Introduction. [Online]. Available: http://securitycatalyst.com/role-and-rule-basing-part-1-introduction/

[30] J. K. Waters. The abcs of identity management. [Online]. Available: http://www.csoonline.com/article/2120384/identity-management/the-abcs-of-identity-management.html

[31] A. Ess, B. Leibe, K. Schindler, and L. Van Gool, "Moving obstacle detection in highly dynamic scenes," in *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, May 2009, pp. 56–63. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5152884

[32] Wikipedia. (2015) Identity management. [Online]. Available: http://en.wikipedia.org/wiki/Identity_management

[33] 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. 20–20. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855711.1855731

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

[35] 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. [Online]. Available: http://doi.acm.org/10.1145/2046707.2046767

[36] K. Ren, Y. Kwon, M. Balazinska, and B. Howe, "Hadoop's adolescence: An analysis of hadoop usage in scientific workloads," *Proc. VLDB Endow.*, vol. 6, no. 10, pp. 853–864, Aug. 2013. [Online]. Available: http://dx.doi.org/10.14778/2536206.2536213

[37] S. Shankland. (2008) Google spotlights data center inner workings. [Online]. Available: http://www.cnet.com/news/google-spotlights-data-center-inner-workings

[38] C. Jin, C. Vecchiola, and R. Buyya, "Mrpga: An extension of mapreduce for parallelizing genetic algorithms," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, Dec 2008, pp. 214–221. [Online]. Available: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4736760

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

[40] AWS. (2015) Amazon ec2 pricing. [Online]. Available: http://aws.amazon.com/ec2/pricing

[41] M. P. Ltd. (2012) Developing task model applications. [Online]. Available: http://www.manjrasoft.com/download/3.0/TaskModel.pdf

[42] Manjrasoft. (2012) Developing thread model applications. [Online]. Available: http://www.manjrasoft.com/download/3.0/ThreadModel.pdf

[43] Wikipedia. (2014) Feasible region. [Online]. Available: http://en.wikipedia.org/wiki/Feasible_region

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