Constructing and Testing Privacy-Aware Services in a Cloud Computing Environment – Challenges and Opportunities (Invited Paper)

Lin Gu
Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
lingu@cse.ust.hk

Shing-Chi Cheung
Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
scc@cse.ust.hk

ABSTRACT

After decades of engineering development and infrastructural investment, Internet connections have become a commodity product in many countries, and Internet-scale "cloud computing" has started to compete with traditional software business through its technological advantages and economy of scale. Cloud computing is a promising enabling technology of Internetware. One distinct characteristic of cloud computing is the global integration of data, logic, and users, but such integration magnifies a sharp concern about privacy, which is one of the most frequently cited reasons by enterprises for not migrating to cloud-based solutions. We argue that cloud-based systems should include privacy as a fundamental design goal, and that privacy in a cloud environment is bidirectional, covering both end users and application providers. End users need privacy-aware software services that prevent their private data from being exposed to other users or the cloud providers. Application providers need a privacy-protected testing methodology to prevent the companies' internal activities and product features from leaking to external users. Focusing on privacy protection, we discuss the research challenges in this unique design space, and explore potential solutions for enhancing privacy protection in several important components of the system.

Categories and Subject Descriptors

C.2.4 [**Computer-Communication Networks**]: Distributed Systems—*distributed applications*; D.2.9 [**Software Engineering**]: Management—*software quality assurance*; D.4.6 [**Operating Systems**]: Security and Protection—*access controls*

General Terms

Design, Human Factors

Keywords

Privacy protection, cloud computing, storage system, centralized processing, privacy-aware testing

1. INTRODUCTION

Cloud computing is postulated to be a paradigm that likely re-shapes the IT industry and software development process. By consolidating the computation carried out jointly by many datacenters, the economy of scale makes it possible to provide high-quality computing services at reduced cost. There are three major entities in a cloud-based application system – cloud providers, application providers, and end users, which correspond to the SaaS users, cloud users, and cloud providers, respectively, in the role assignment used by Armbrust et al. [1]. Organizing users and computing resources in the cloud computing model, as illustrated in Figure 1, provides significant benefits to all three entities because of the increased system efficiency and the economy of scale. Major IT solution providers and Internet firms, such as IBM, Microsoft, and Amazon, have realized the challenges and opportunities in this trend, and invested heavily in both the infrastructure and technological innovation in this area.

Some groups, however, question whether the concept and technological components of cloud computing are generally applicable to professional IT services. Among the concerns, privacy is one of the most frequently quoted reasons for not migrating to a cloud-based solution. Though there is similarity between cloud computing and earlier ideas such as grid computing, utility computing, or even network computers [1, 2], earlier systems did not have the ambition and potential of reforming the whole spectrum of information processing systems from personal software, handheld devices, to corporate business solutions. Hence, the lack of a sound privacy guarantee has become a sharp pain with broader and deeper implication than what earlier systems have encountered or had remedy for. Currently, the guarantee of users' online privacy is mainly supported by service providers' "good

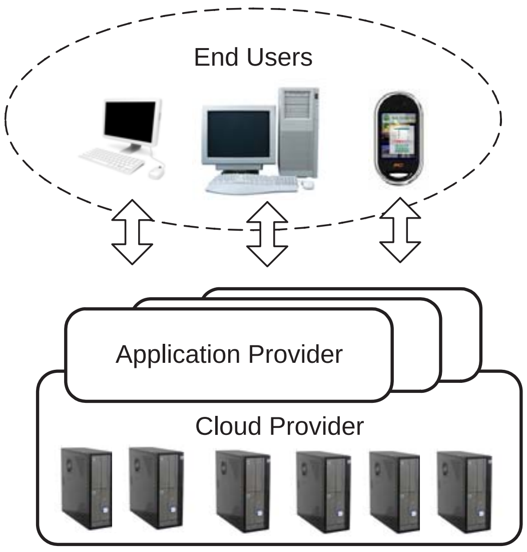

Figure 1: Participants of a cloud-based application system. Cloud providers are operators of a cloud infrastructure, including data centers, replication sites, and network facilities. Application providers develop application services on the cloud infrastructure. The end users are the customers who use the application service.

faith" practices, but privacy infringement are not unheard of [3]. The current practice cannot provide the strength of privacy protection needed for serious personal and business applications.

Privacy in cloud computing should be supported based on sound guarantee rather than "best effort" operational disciplines. Internal regulations and external legal enforcement are important to cloud service providers, but they would be far less effective if the technical foundation is weak. Hence, we need a set of methodologies to design, build, test, operate, and maintain software systems, Internetware components, and cloud-based computing services with explicit privacy guarantees.

However, privacy is traditionally not a primary design goal in software development and quality assurance practices. Many privacy solutions take the form of an "add-on" component inserted to a privacy-oblivious system – a system without explicit privacy guarantees. Similar to the case of computer security, add-on solutions implemented as an "afterthought" introduce unnecessary difficulties in software design, often leading to insufficient protection, inefficient operations, and incomplete adoption [4]. After years of development, many aspects of security requirements for computing services are understood, and numerous solutions are implemented. Privacy protection, however, is a different from computer security, and the cloud-based environment introduces new challenges to the system design. Traditional methods, such as encryption, would result in inefficient solutions in this new design space. Moreover, privacy protection in a cloud environment is bidirectional – not only need the end users' privacy be protected, but also the application providers need to ensure that their internal operations, including resource planning, product releases, bug fixing, research efforts, and code testing, are not revealed inadvertently or prematurely. The testing methodology is of particular importance to privacy protection for application providers because the quality assurance practice in cloud computing must involve a large number of external users. Traditionally, the testing procedures often emphasize the performance, coverage, and reliability. Much less attention is paid to protecting the privacy of the software components themselves. Even with black box testing, it is usually assumed legitimate that testers know the interface, functional specification, or properties-under-test of the software components. As we will analyze later, such assumptions and practice are no longer viable in the cloud-based paradigm.

In seeking solutions to the above-discussed problems, a clear understanding on the nature and challenges of cloud computing is useful. However, there has not been a consensus on the definition and unique features of cloud computing, though the term is widely used. Such confusion hinders us from identifying important problems and exploring effective methodologies for software development under this paradigm. For example, one straightforward explanation of cloud computing is to move the data and logic to the "cloud" (Internet or, more precisely, managed datacenters in the Internet), and combine the strengths of SaaS, utility computing, and grid computing. This view generally ignores the important role of users – when the user population and their participation in the system grow beyond a tipping point, the old computing paradigms are no longer efficient, and a new one needs to be created. Such incomplete views may lead to designs and solutions with serious limitations in a cloud environment.

In this work, we explore the requirements, research problems, and opportunities in constructing cloud-based, privacy-aware computing systems, focusing on the key privacy-aware components for such systems and the testing methodology. We first re-examine what is needed for the next-generation computing systems, and give a simple and informal definition of cloud computing as an ultimate "integration" of basic elements of computing systems. We then examine the privacy-related requirements of a cloud-based system, and propose solutions to organizing data and logic with privacy protection. As aforementioned, not only need the end users' personal information be protected, but also application service providers' internal information should be hidden from external entities. To address this problem, we compare the testing in a cloud environment with that in traditional in-house development environment, and discuss research issues toward testing methodologies that protect the privacy of the application providers'.

In our discussion, we use a very simple case, a "cloudy counter" service, to illustrate some technical issues. This minimalist case can be viewed as a "hello world" equivalent in the cloud-based environment. But a neat solution to this simple problem, at a scale appropriate in our design context, is non-trivial.

The rest of the paper is organized as follows. Section 2 introduces the characteristics of cloud computing, and emphasizes the privacy concerns in this new design space. Section 3 presents a “cloudy counter” example, highlights some technical challenges in constructing cloud-based services, explores research issues in privacy protection, and proposes potential solutions. Section 4 discusses research issues in testing cloud-based software, followed by a survey of related work in Section 5. Finally, Section 6 concludes the paper.

2. CHARACTERISTICS OF CLOUD-BASED SYSTEMS

To emphasize new challenges in designing and testing cloud-based systems, a comparison between cloud computing and earlier paradigms is in order. However, it is surprisingly difficult to derive a unanimously agreed definition for cloud computing [1, 5], and many researchers view cloud computing as an old idea recently revived as large-scale datacenters are constructed and made available for diversified use. We argue that cloud computing presents new challenges and research problems that earlier computing methodologies, such as SaaS (Software as a Service), grid computing, and utility computing, do not focus on or have solutions for. To illustrate this, we examine the properties of computing paradigms using a “DUL simplification”, and discuss their implications in system development and testing.

The DUL simplification views computing systems as implementations, combinations, and interactions of three basic elements – data, users, and logic. These three elements are “basic” because 1. they exist in almost all non-trivial computer applications; 2. they are primitive and, to a large extent, orthogonal – multiplying one element or adding two elements together does not produce the other element(s). Most of the characteristics and operations of a computing system can be related to characteristics and operations of these three elements.

Some other concepts, such as “program” and “computer”, are not considered basic elements because they are not primitive. For example, a program comprises data and logic. A “computer” is the equipment that facilitates the storage of data and the execution of logic. In some cases, a “computer” can be hypothesized to be a Universal Turing Machine[1], which is, consequently, also a program.

Following the DUL simplification, computing is, generally, to transform data with logic in a way that users find useful. The three basic elements take different forms, have different weights, and incur different constraints in various computing paradigms. When the characteristics, weights, and structure of the three elements change significantly, the design space of the computing systems changes accordingly, and new computing paradigms may emerge. For example, as the speed of logic transition became much faster than that of data feeds and user operations in the 1960’s, batch programming and time-sharing systems were made technically viable, and they grew to be the prevailing paradigms

[1]To describe a computer as a Turing Machine is an approximate but useful analogy. To name one difference, a Turning Machine, by definition, has infinite data storage, but real-world computers have finite storage.

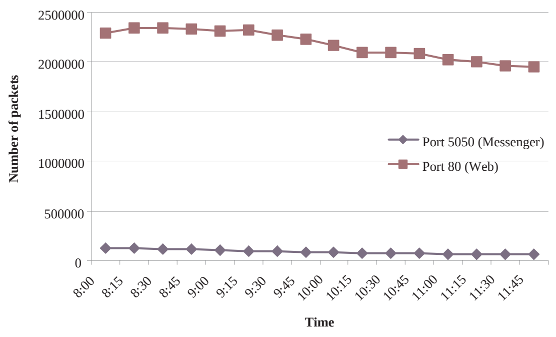

Figure 2: Number of traces observed per 15 minutes on a network exchange operated by Yahoo! (Courtesy data from Yahoo! [6]). Each trace containing one or more packets, the data reflect regional workload on two ports (Web and Messenger) on Apr. 29, 2008 (GMT). The local time (U.S. Pacific Time) is GMT-7.

for that time. Similar causal relationships can be observed in the motivations and development of personal computing, the client server model, and browser-centric system designs.

Today, the nature of data, users, logic, and their interaction is very different from that of earlier systems. The data size is excessively large, logic is loosely coupled, yet a vast user population is closely “connected” on a shared platform. The new properties of data, users, and logic follow recent development in computing and communication hardware, usability of digital services, and information service commoditization, and they make cloud computing a viable and favored solution. Obviously, it is advantageous to organize the global computing resources to serve global users in an “integrated” way, and thus construct a much more efficient system that scales and saves. The cloud computing paradigm meets this requirement and holds the promise of implementing systems at such scale.

As an informal definition, cloud computing is the unification of data, users, and logic at vast scale. Though there exist very large computing systems that emphasize integration, the depth and breadth of the unification in cloud computing are orders of magnitude larger than those of earlier systems. A successful cloud computing system should be able to service millions of globally distributed end users, such “Internet-scale” computing makes many earlier solutions unsuitable in this new design space.

To illustrate the problem size of an Internet-scale system, Figure 2 shows the sampled network traffic on a Yahoo network exchange. The sampled packet trace represents a small portion of data traffic into and out of the exchange – only a small fraction (about 0.1%) of packets are recorded. This data shows that the application systems (Web and Messenger) handle millions to hundreds of millions of packets every minute. Note that the data is for the geographic region covered by this network exchange. The entire system, distributed and loosely coupled, comprises more than 100 datacenters and exchanges. Computation and communication at such a magnitude, with data, users, and logic distributed globally yet forming a holistic system, impose a grand challenge on system design and software development.

It is also this triadic integration that makes privacy a serious concern in system design. To accomplish high efficiency for large user populations, cloud computing systems need to break down the barriers between processes, machines, users, domains, access control groups, and other traditional protection zones. Data should flow smoothly from process to process, from machine to machine, and from domain to domain. The logic may also span multiple domains, e.g., orchestrating millions of processes across several time zones to smoothly handle a workload with diurnal patterns. Hence, how to protect the personal identities of end users and non-public information of corporations becomes a technical challenge and a major concern in this environment.

Wary of the risks in giving a definition, especially, one to a concept that has been used and discussed by many groups, we believe this extended analysis helps expose the nature of cloud computing and its unique design space so that we can describe the technical details and potential solutions with ease and precision. On such ground, we analyze the development and testing of privacy-aware services in the following sections.

3. CONSTRUCTING PRIVACY-AWARE COMPUTING SERVICES

Focusing on engineering practice and end users' privacy issues, we discuss a few challenges in the software development and system construction for cloud-based services. As research problems and solutions must be examined against the technical background to which they are applied, we highlight the difference between the privacy concerns in traditional systems and those in cloud computing as follows. We hope that they help illustrate the unique challenges and the specific solutions to be presented in the following sections.

- From the end users' point of view, the privacy concern involves two separate entities collaborating to provide a service – the application provider and the cloud provider. These two entities often have different perspectives on privacy protection, and the users may want to deal with them differently. For example, a user is likely to trust an application provider – otherwise, the user would not use the application service – but may not trust the cloud provider.

- Traditional privacy protection techniques, such as encryption and artificial noise injection, may not work efficiently in a cloud environment. For example, strong encryption is unlikely to be sufficient for protecting user data because the application service, provided by an external entity, needs to understand the data, and the user "state" (data) ultimately must be materialized in a memory structure maintained by the cloud providers infrastructure when it is being processed.

- Solutions must not diminish the benefits of cloud computing. Designed for "in house" processing, many traditional solutions scale poorly on a million-user platform, or introduce difficulty for the application provider and cloud provider to reduce overall system cost.

3.1 Data and operational privacy

Exemplified by Wirth's well-known equation "Algorithms + Data Structures = Programs" [7], data and logic are two major aspects of software products. Related to them are two types of privacy of particular importance – data privacy and operational privacy. Data privacy is to guarantee that the correlation between users and their data cannot be identified. Note that we may not require that users' data are invisible to other parties – personnel may see a user's data, but those unauthorized should not be able to identify the connection between the data and the user. This is an important difference from encryption. As aforementioned, encryption can be one means of protecting privacy, but it is an incomplete solution and performs poorly in a cloud environment. On the other hand, it is possible to protect privacy without exercising heavy encryption.

One example of data privacy concern is behind the growing popularity of online storage for photos, emails, office documents, and other digital contents. If not appropriately protected, such data files can potentially be read by technicians maintaining the service. Though infringement cases are believed to be infrequent because of the high ethical standard practiced by leading Internet application service providers, there is no technical guarantee that personal or proprietary information is safe. It is also difficult for the owner of the data to audit the accesses and detect violations of access policy. Many individuals and business owners are deeply concerned with this negative potential, and only send non-critical files to the online services. This has hurdled the migration of business applications to the cloud computing paradigm. Challenges in data privacy, hence, include the construction of technically sound privacy-aware data storage and data access methods, and the implementation of a mechanism to enable data owners (end users) to "participate" in the internal operations of a cloud provider for exercising owner-and-application-specific access control and auditing.

Operational privacy is to prevent operations issued by a user from being identified and related to the user. Once again, the purpose is to disconnect the linkage between the operations and the user. It may not be necessary to completely "encrypt" the user's operations.

An example of operational privacy is the query logs recorded by almost all major web sites. Most of the major web sites keep the log files generated by their web servers and other application services, and such logs accumulate to a gigantic database where an individual user can be identified through IPs, login IDs, telephone numbers, credit card information, and behavioral patterns, and associated to a historic sequence of operations performed by that identity.

When we develop a computing service in the cloud environment, the data and operational privacy are important design goals. In the following sections, we describe challenges and opportunities concerning these two types of privacy. In our discussion, we use a "cloudy counter" example to facilitate the discussion and illustrate the solutions. A counter is a software component that keeps increasing upon every request, and can be used to implement an equivalent to sequence numbers in an online environment. As the name implies, the cloudy counter problem is to implement the counter logic in a cloud environment. In this section, we concentrate on the cloud-based solution and privacy issues.

The cloudy counter: The cloudy counter service takes a relaxed semantics as follows. Upon a request, the cloudy counter service returns a counter value, and successive counter values thus returned maintain monotonicity in each "channel". A channel can be implemented as a control structure affiliated with one TCP connection. The counter service does not guarantee the relative order of counter values in separate channels. However, if a request from one channel specifies a "last counter value" parameter, which is possibly obtained from a different channel, the counter service guarantees that the returned counter value is larger than the last counter value. We also explicitly relinquish the requirement that the "counting" covers all numbers. That is, some numbers may be skipped.

The cloudy counter thus defined can still support strictly coordinated operations in the cloud environment, and application designers, after understanding the mechanism of the counter, can adapt their programming activity to use the new counter semantics to construct complex application logic. The new semantics requires a slightly different interface. In the traditional counter interface, a new counter value may be requested by the following function implemented as a local function or a remote RPC:

```
int64_t Counter::next();
```

Here we use the C++ syntax to specify the function, and int64_t is a 64-bit integer type. The cloudy counter interface, however, requires an additional parameter:

```
int64_t Counter::next(int64_t last);
```

Here "last" specifies the "last counter value". With this change, the implementation of the cloudy counter becomes highly scalable, with a large number of counter servers handling allocated portions of the counter values and synchronizing only when the "last" parameter indicates an out-of-synchronization situation. Except for the burden of maintaining the last obtained counter value locally, programmers can easily adapt their code to use the new semantics and interface to implement various counter-related logic. Our observation is that, in many cases, it is possible to construct highly useful and efficient systems using cloud-friendly semantics. Many broadly used scalable solutions, such as GFS [8] and Bigtable [9], show a similar pattern of semantics relaxation and programming adaptation. Specifically, GFS relaxes the data consistency model and Bigtable limits transactional scope. With programming adaptation, such relaxation do not noticeably lower usability.

3.2 Tradeoffs in of privacy-aware designs

Ever since digital computers were first introduced to commercial organizations, most business organizations have been assuming they have full control over their computer systems. Even when some computing tasks are outsourced to external service providers, the data exchange and operational interfaces are clearly defined and, sometimes, legally bounded, by mutual agreements. Cloud computing is the first major computational paradigm that departs from a clear definition of ownership. The magnitude of the global integration behind the concept of cloud computing means a delineation of bound and responsibilities by bilateral agreements is no longer possible. For cloud computing to be successful, trust and protection must be established by technical means sound enough to relieve worries for most of the commercial purposes.

Traditional solutions, such as encryption, cannot provide an efficient mechanism for privacy protection in a cloud environment. Though cryptography has been a mature area with many standardized solutions, protecting privacy through encrypting user data has drawbacks, among which the prohibitive overhead is a major concern. While encryption can be conducted on users' computers to reduce server-side workload, encrypted files deprive the cloud providers of the opportunity of merging identical data to save storage space, and hinder application providers' capability to index and process the data. One challenge in designing a cloud-based solution is to provide protected, efficient, and sometimes transparent interaction among various users, application providers, and cloud providers. The designers must take into consideration not only the strength of protection provided to users, but also the efficiency of the overall system including all entities. This is particularly important because one major advantage of cloud computing is the economy of scale, and severe degradation in overall system efficiency diminishes this benefit.

3.3 Protecting data privacy

Traditionally, data privacy is protected by either restricting the access or anonymizing the data. Access restriction can be achieved by encryption, mandatory access control, or discretionary access control. Access control prevents unauthorized users from querying the protected data, and encryption essentially restricts data to be visible to only the parties with appropriate keys. Anonymizing data is a common technique that adds noise to either the data returned by a query or the raw data itself. All these privacy preserving techniques find applications in some design contexts. However, many existing solutions are application and system specific, and most of them are not designed for a cloud environment.

Would it be feasible to formulate a general mechanism for privacy-aware data storage? While it remains a grand challenge to create a general purpose mechanism to assure data privacy in a cloud environment, there are many opportunities in this domain. We enumerate the research questions and opportunities below, and describe design details currently explored in our research project.

First, what interface should cloud-based entities use to access data? Traditionally, users and applications organize data as a hierarchical file system or relational database. In a cloud environment, the data may be accessed by both end users and cloud application providers. Recent solutions, such as GFS, S3, and Bigtable, take a similar approach, but extend the capability to scale up to a large distributed installation. However, these scalable solutions do not provide an easy-to-use interface to end users, and do not provide strong privacy protection. For example, Crosby from Citrix notices that the Amazon EC2 service [10] may allow unauthorized users to probe proprietary information on virtual disks [11].

The lack of a crystal solution also indicates an opportunity for creating a cloud-based data access interface that is efficient, privacy-aware, and easy to use by both end users and application providers. We are designing a "virtual private data repository (VPDR)" service as a general mechanism for privacy-aware data access. The VPDR provides a file system interface that is familiar to most users and application programmers. However, data written to the VPDR are obfuscated so that they cannot be easily related back to a particular user, yet a user can efficiently access VPDR with a user-controlled access token. Before a cloud-based application processes user data, it authenticates itself to the user and obtains a derived token to unlock the data. The privacy protection in VPDR is provided by a software component called virtual private disk (VPD). The VPD is implemented on users' computers or included in a cloud-based application program, presenting an I/O device on which the VPDR file system is constructed. When data is written to the VPD device, the VPD driver obfuscates the information by slicing the data at bit level based on the , then sends data slices to servers maintained by the cloud provider. Without the derived token provided by the user, it is difficult to restore the data from obfuscated data slices, unless one collects a sufficient number of slices and perform a heavy matching test. To strengthen the obfuscation, multiple cloud providers may be employed. Such multi-provider systems provide additional benefit of enhancing the availability and performance of the application.

Note that the protection mechanism with the access token is different from encryption, because the bit-level slicing makes the bit sequence in a slice practically illegible, but still keeps some structural regularity inside data. Therefore, a cloud provider can still efficiently perform duplication removal, compression, and merging operations on the slices. On the other hand, the access token randomizes the locations where the slices are stored. Hence, the storage media holds a mix of slices from numerous files belonging to various users, and the dramatically increased entropy makes it very difficult, though not impossible if sufficient computing resources are available, for anyone without the derived token, including the cloud providers, from finding exactly the relevant slices and matching them to form the original file. The principle is to allow cloud providers to efficiently manipulate bits, not to manage contents.

Second, how can we prevent major cloud providers from accumulate sufficient data owned by specific users? As aforementioned, the VPD uses obfuscation to make it very costly to decode the data without the access token, but a dedicated party with sufficient resources (e.g., the cloud provider itself) may reduce the cost by associating bit slices to specific users, and restore the original files after collecting sufficient slices and launching a serious deciphering effort.

There are several opportunities for mitigating the "data accumulation" vulnerability. Using multiple cloud providers is an effective solution. Another approach is to employ a collaborative scheme we call "virtual network buffer (VNB)" to communicate with the cloud provider with controlled uncertainty. The key issue is to dissect the linkage between end users and bit slices so that gleaning a particular user's data without the access token or derived token involves considerably higher cost. Even though a dedicated party with sufficient computing resources can still decipher data, such heavy practice would most likely be detected by the auditing mechanism, which we believe is another necessary technical component for privacy protection.

Third, how could a cloud provider construct a storage system to handle large-scale and privacy-aware data? The VPDR and VNB operate on end users' computers and in the cloud-based application services. The cloud provider, which ultimately provides the hardware resources for storing data, also needs to provide a privacy-aware technical solution. The main purpose is to guarantee the privacy protection between end users, and minimize heavyweight formatting operations before allocating storage space to users.

We propose a "virtual cloud storage" (VCS) service for this purpose. The VCS software resides in the cloud, and is implemented as a genuine storage system on dedicated servers or as an overlay service riding on public cloud-based storage solutions, such as Amazon Simple Storage Service (S3). No matter whether it is implemented as a genuine or overlay system, VCS ensures that user data are sliced and stored in separately operated partitions so that one single operator cannot easily combine slices to assemble the original user data. The slicing and partitioning process ensures that the data in one storage area are obfuscated bits from different sources. On the other hand, the deletion and migration of user data is less straightforward, due to the uncertainty in the linkage among slices, files, and users. This is another challenge in cloud-based data storage. Slice link-count provides a solution, but we believe better mechanisms are possible.

Figure 3 shows the architecture of a privacy-aware data service using the VPDR, VNB, and VCS mechanisms. To illustrate the topics discussed in this section, we examine how the cloudy counter service can potentially use the privacy-aware designs. Though the counter is theoretically just a very wide integer, the set of counter values an end user has requested is a piece of "private" data owned by the user. This data is physically

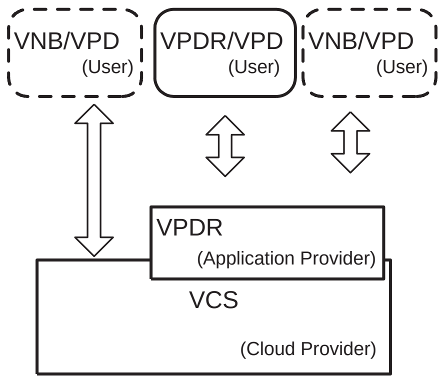

Figure 3: The architecture of privacy-aware data service

stored in the cloud provider's datacenters, but it has undertaken two passes of obfuscation process by VPDR and VCS, and physically resides in the storage system as a collection of "random" bits distributed in a number of storage media. When the user needs to access the counter values, the request is serviced by the application provider through a derived token obtained from the end user. Replication and availability are maintained by the VCS by the cloud provider, and the application designers determine special indexing and strategic positioning heuristics of the data to enhance performance. Current cloud services, such as Amazon AWS/EC2/S3, Google Apps Engine, and Microsoft Azure, are yet to provide application developers with such flexibility and strong privacy guarantee [10, 1].

3.4 Protecting operational privacy

Developing an algorithm for a cloud environment must consider scalability, exploit parallelism, and protect user privacy. One grand challenge in solving an "Internet-scale" problem is to automatically exploit parallelism within the algorithmic steps and the data items. Semi-automatic parallelization has been proved useful in some large-scale systems, such as web search engines [12]. Any simple search query results in a computation over the whole Internet's worth of data, yet the result should ideally be returned to the user with subsecond latency. Such performance is impossible to accomplish without parallelism. To address this challenge, parallelization models, such as MapReduce, have gained popularity [13]. A number of variants of MapReduce have been proposed, implemented, and installed at a number of institutes and organizations [14, 15]. However, most of such models are motivated by query-style computation, and do not address privacy issues.

Take MapReduce as an example. To illustrate its limitations, let's consider the MapReduce algorithm that one would construct for a simple function related to the cloudy counter: listing all the counter values owned by a user. The algorithm would, most likely, Map all counter values to pairs in the form $< owner, 1 >$, and sum up the pairs with key "owner" equal to the ID of the user. Clearly, by examining the Reduce task, the service provider, who runs the Reduce task, can easily infer the user who has initiated this computation, or even the intention of this MapReduce task.

Note that the end user is likely to trust the application provider – the provider of the cloudy counter service in this example – but may not have as much trust on the cloud provider, which is an obscure entity too remote from the end users for them to understand and trust. Hence, the ability for the cloud provider to identify users and relate user identities to operations is potential privacy threat, particularly to commercial users.

One research question is how to enhance existing parallelism frameworks, such as MapReduce and Hadoop, to provide privacy protection. The MapReduce model has not provided a mechanism to protect users' privacy. Intrinsically, privacy control is very difficult to be added to MapReduce because it was not one of the original design goals of this parallelization model. It is, hence, left to the "good faith" practice of the service provider to apply operational regulations to cover this technical weakness. Moreover, MapReduce is largely a simplified model for parallelization. The essence of MapReduce is to seek data-level parallelism, and coordinate the computation by a barrier operation in the shuffle stage between the Map and Reduce operations. With such a simple model, it remains a research question how to enhance MapReduce-style frameworks in privacy awareness.

How can we design a new parallelization framework that scales efficiently and protects privacy? We believe that one opportunity is to take advantage of the fact that the application provider can essentially delegate end users' tasks to the compute infrastructure. If the application provider is trusted, an indirection level implemented in the application service logic can, to a large extent, hide the identities of end users. Another research problem is to protect the privacy of the incoming user requests, if logging is enabled, the gateway interface (e.g., Apache's HTTP handler) can collect information on the user identity. This problem, again, can be conveniently solved if the application provider is trusted.

Overall, we believe that the operational privacy presents many open problems yet to be addressed. For instance, it is, in fact, non-trivial for a commercial entity to trust an application provider. High-fidelity auditing and verification techniques will help in this area, and a complete solution may involve elements beyond the technology domain.

4. TESTING CLOUD-BASED SERVICES

Software testing and quality assurance take a new form when end users become an integral part of a unified computing system. The traditional wisdom for in-house testing may not apply for many cloud-based services. For example, the managerial practice of allocating 50%–70% time for testing is no longer efficient or practical. Nor is it necessary.

One disruptive factor in Internet-based software development is the fast-paced software development and releases, and cloud computing certainly share this nature. This is driven by competition, and enabled by efficiency of online software distribution, often in the form of service delivery. Among the three basic elements of computing, logic can now be modified and improved at any time an application provider desires. Hence, Agile software development has gained popularity in leading online service providers, such as Yahoo, Google, and Amazon [16]. Another basic element, the user, has become an active participant in the process of software development in many cloud-based environments.

Traditionally, users of computing systems include developers, maintenance technicians, and end users, and the end users used not to directly determine the internal algorithms, organization, and implementation of a computing system. However, the end users can serve as an extension of the development team in the cloud computing paradigm, because the users have become an active part, continuously interacting with logic and data, in the holistic system following the triadic integration.

The active interaction of users is not limited to testing. Modern Internet-based algorithms "re-program" themselves based on user behavior. For example, the search algorithm adjusts the page ranking logic by analyzing the click rates and other user activities, in order to tune the algorithms to the best extent. One user's interaction, may affect the system behavior observed by millions of other users. Hence, the software behavior evolves continuously with user behavior. In this sense, users are closely involved in the internal design of cloud-based software by either mutating program state that affects the system behavior, or changing the program logic learned through the process of interacting with a larger number of users. Though traditional software also allows users to change configurations or options, the user participation in Internet-based services is deeper, broader, and more instantaneous.

In testing, end users' participation is even more active and direct – they have become a powerful and indispensably part of the "testing team" for cloud application providers. As cloud computing-based solutions expand to office and business application areas, the magnitude of integration continue to develop in this direction. As the other side of privacy protection, it is now necessary to assure that the user engagement at earlier stages of software production does not prematurely or unexpectedly reveal internal information of the software or organization involved in the testing.

4.1 The new paradigm of software testing

It is well known that the developer-to-tester ratio at some traditional software companies is around 1:1, but the ratio at some leading Internet application providers is n:1 where n is much larger than 1. This phenomenon has inspired a wide spectrum of views and discussions on the Internet. [17] offers a discussion on the developer-to-tester ratios at Microsoft and Google. The reason for this disparity is not the capability of programmers, but the quality assurance methodology. Online services often exhibit an agility to fix bugs quickly and release new versions of software within several weeks. How to assure software quality is a challenge in such a fast-paced release process.

To solve this problem, online application providers routinely take an incremental-release approach. A new release is first distributed to a small fraction of users, and proceeds with larger user base tests only if smaller-scale tests are successful. Such a small-change, small-scale quality assurance process seldom introduces inconveniences to the initial testers – the annoyance of network outage has a much higher occurrence rate than the software failure due to testing activities. In practice, the incremental testing and releasing are highly unlikely to negatively affect a large number of users if the process is carefully administered. However, such incremental releasing essentially includes end users as part of the development team as testers, and enables the development of complex, reliable, and high-quality software yet keeps the core engineering team surprisingly small and well-communicating. This was not possible in pre-cloud systems.

4.2 A privacy-aware approach for testing

The involvement of users in the very early stage and the whole process of designing, implementing, and tuning complex software in a clouding environment calls for a bidirectional protection for privacy. One direction is to provide privacy guarantee for users, which is described in Section 3. The other direction is to protect cloud service providers' internal information from leaking to the public. The latter is often neglected because it has much less publicity than the former issue. However, protecting the service provider is no less important than protecting the end users' privacy, and a weak guarantee in this direction would hinder the development of the cloud-based methodology.

While incremental release and testing have been a standard practice in online service development, it remains a research question to formalize this process and explore privacy-aware procedures.

It is worth clarifying that the privacy concern in this context is broader than testing privacy guarantees provided by the component-under-test *per se*. There are two types of privacy concerns in the testing process. The first type is the privacy protection of user data retrieved and handled by the software component. The main focus is to protect the users, and the test should assure that the component does not leak sensitive user information. The second type is the privacy protection of the identity and property of the software component itself. The main focus is to protect the cloud application provider, and the test should ensure that the testing procedure itself does not leak proprietary or premature information about the application provider and the component-under-test. There are research questions concerning the testing practice for both types of privacy concerns, while the first type has been partially covered in some early research works involving security and privacy specifications and the test cases covering the specifications [4].

Specifically, testing a software component for its privacy guarantee involves specifying a set of predefined privacy policies, and designing test cases to provide an adequate coverage. As an incomplete list, the important research questions to be solved are the following:

1. How can privacy policies be modeled so that they are subject to specification based testing?

2. Can we exhaustively test a piece of software against its privacy policies? What is the criterion of "exhaustive coverage"? If "exhaustive coverage" is too expensive, could we relax it to "adequate coverage"? What is then the criterion of "adequacy"? How can we evaluate the quality of such adequacy?

3. How can we generate the test inputs and data to achieve such coverage effectively? How can we generate the test oracles which can judge whether a test outcome is correct or not? What are the challenges in formulating test oracles for privacy?

4. How can we identify the problem sources from test outcomes? Do we need to instrument the software accordingly? How does the instrumentation for privacy differ from traditional ones?

5. How can we generalize the issues to cloud computing?

To protect the cloud application providers, the testing procedure should prevent attentive observers from acquiring internal or premature information about a software component by participating in the testing process as an external user. Specifically, the following research questions need to be answered.

1. What is an appropriate abstraction of a "user" in a testing procedure? What are the properties of the user, and how can we map these properties to the testing process?

2. What information about the software component should be given to the external test users, and what should be hidden from them? How can we provide sufficient information to facilitate testing, yet ensure the hidden information is protected?

3. How do we classify external users to be a number of groups, with each group associated with a trust level? How to measure and monitor users' trustworthiness?

4. What obfuscation is needed to protect the code to be tested? How to implement such obfuscation efficiently?

5. Can we obtain high-fidelity test results with test cases divided among users and test code obfuscated for privacy protection?

6. How to modularize the user-related software components to facilitate efficient and safe testing and ensure confidentiality and user satisfaction?

5. RELATED WORK

It is widely perceived that cloud computing calls for new methodologies for developing and testing software services. Related to the concept of Internetware, modules and components are investigated in various cloud-based architectures [18, 19]. Researchers also point out that virtualization, outsourcing, and a view of "global" software are important elements in the new paradigm [20]. Pearson discussed privacy aware software development guidelines in a cloud environment [21]. However, there has not been a consensus on what the key challenges are and what the solutions may be.

The software development and testing of online services have shown significant difference from the traditional in-house project development. Many online application providers advocate the "agile" methodology [16, 22]. However, most of the agile practices focus on enhancing the productivity and predictability of software development, not privacy protection in the system architecture and software testing process. The mystic disparity between the developer-to-tester ratios has been widely discussed, but only a few articles attribute this to the integration of users in the holistic system. For example, Whittaker discussed this disparity in his blog but only one reply post mentioned that the true reason is that "Google's online-centric model means that bug fixes can be rolled out rapidly" [17]. We believe that this disparity is, in fact, the tip of an iceberg of a significant paradigm shift in software engineering practice.

Program state (data) management is a key part of cloud computing. A number of solutions have been proposed and implemented for data storage, data access interface, or even specific "storage cloud" [10, 9, 8, 20]. Overlay storage services have been practiced by a number of storage service providers. For example, it is well known that some storage software implement a large online storage using numerous gmail accounts, each account providing a few GBs of storage [23]. However, how to protect privacy in such systems is an open problem.

Designed by Google, MapReduce is by far the most successful parallelization framework used in Internet-scale computing [13]. Hadoop is an open-source implementation of MapReduce [15]. In the early design stage of MapReduce, the design goals emphasized performance and scalability, but largely ignored privacy issues. It might be legitimately argued that privacy was a non-issue when MapReduce was designed and employed for the searching pipeline including crawling, indexing, and ranking. The situation is, however, completely different when we migrate to the cloud computing paradigm where more sophisticated logic and dependence must be in the system design specifications.

As users are integrated as an active part of a unified computing system, the modeling and understanding become more important in the cloud computing paradigm. It is known that end users and their "programming" ability are difficult to model and integrate [24, 25]. Innovations in this direction would dramatically enhance our ability to conduct testing, estimate test coverage, and reason about service quality.

6. CONCLUSIONS

It is our hope that this treatise has provided a new insight into the design space of privacy-aware system development and testing in a cloud environment. Not intending to solicit consensus, we hope our analysis can help expose technical challenges and research problems in the next-generation computing, which we believe will contain cloud computing as a central element. Though many research problems can trace back to earlier works in related areas, few existing techniques can provide complete and efficient solutions suitable for this new design space. Given its importance, we believe that novel designs and techniques for cloud computing will develop in the years to come.

7. ACKNOWLEDGMENTS

This work is supported by UGC, Hong Kong. We would like to also express special thanks to Yahoo for the Webscope traffic data they provide.

8. REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. Above the clouds: A berkeley view of cloud computing. *UC Berkeley Technical Report UCB/EECS-2009-28*, February 2009.

[2] Brian Hayes. Cloud computing. *Commun. ACM*, 51(7):9–11, 2008.

[3] Mark Zuckerberg. An open letter from Mark Zuckerberg. *http://blog.facebook.com/blog.php?post=2208562130*.

[4] Premkumar T. Devanbu and Stuart Stubblebine. Software engineering for security: a roadmap. In *Proc. of the Conf. on the Future of Software Engineering*, pages 227–239, 2000.

[5] Ken Birman, Gregory Chockler, and Robbert van Renesse. Toward a cloud computing research agenda. *SIGACT News*, 40(2):68–80, 2009.

[6] Yahoo! Academic Relations. G4: Yahoo! network flows data 1.0. *http://research.yahoo.com/Academic_Relations*.

[7] Niklaus Wirth. *Algorithms + Data Structures = Programs*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1978.

[8] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proc. of the 9th ACM Symposium on Operating Systems Principles (SOSP'03)*, pages 29–43, 2003.

[9] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Trans. Comput. Syst.*, 26(2):1–26, 2008.

[10] Amazon Elastic Compute Cloud – EC2. *http://aws.amazon.com/ec2/*.

[11] Mache Creeger. CTO virtualization roundtable: Part II. *Commun. ACM*, 51(12):43–49, 2008.

[12] Luiz André Barroso, Jeffrey Dean, and Urs Hölzle. Web search for a planet: The Google cluster architecture. *IEEE Micro*, 23(2):22–28, 2003.

[13] Jeffrey Dean and Sanjay Ghemawat. MapReduce: simplified data processing on large clusters. In *Proc. of the 6th Symp. on Opearting Systems Design & Implementation (OSDI'04)*, 2004.

[14] Roy Campbell, Indranil Gupta, Michael Heath, Steven Y. Ko, Michael Kozuch, Marcel Kunze, Thomas Kwan, Kevin Lai, Hing Yan Lee, Martha Lyons, Dejan Milojicic, David OąŕHallaron, , and Yeng Chai Soh. Open Cirrus cloud computing testbed: Federated data centers for open source systems and services research. In *Proc. of the Workshop on Hot Topics in Cloud Computing (HotCloud '09)*, June.

[15] Apache Hadoop Core. *http://hadoop.apache.org/core/*.

[16] A. Cockburn. *Agile Software Development*. Addison-Wesley, Boston, MA, 2002.

[17] James Whittaker. Google v. microsoft, and the dev:test ratio debate. *http://blogs.msdn.com/james_whittaker/archive/2008/12/09/google-v-microsoft-and-the-dev-test-ratio-debate.aspx*.

[18] Alexander Lenk, Markus Klems, Jens Nimis, Stefan Tai, and Thomas Sandholm. What's inside the cloud? an architectural map of the cloud landscape. In *Proc. of the ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 23–31, 2009.

[19] Jan S. Rellermeyer, Michael Duller, and Gustavo Alonso. Engineering the cloud from software modules. In *Proc. of the ICSE Workshop on Software Engineering Challenges of Cloud Computing*, pages 32–37, 2009.

[20] Pavan Yara, Ramaseshan Ramachandran, Gayathri Balasubramanian, Karthik Muthuswamy, and Divya Chandrasekar. Global software development with cloud platforms. In *Proc. of the 3rd Intl. Conf. on Software Engineering Approaches for Offshore and Outsourced Development (SEAFOOD'09)*, 2009.

[21] Siani Pearson. Taking account of privacy when designing cloud computing services. In *Proc. of the ICSE Workshop on Software Engineering Challenges of Cloud Computing (CLOUD'09)*, pages 44–52, 2009.

[22] Sridhar Nerur, RadhaKanta Mahapatra, and George Mangalaraj. Challenges of migrating to agile methodologies. *Commun. ACM*, 48(5):72–78, 2005.

[23] Gspace site. *http://www.getgspace.com/*.

[24] Margaret Burnett, Christopher Bogart, Jill Cao, Valentina Grigoreanu, Todd Kulesza, and Joseph Lawrance. End-user software engineering and distributed cognition. In *Proc. of the 2009 ICSE Workshop on Software Engineering Foundations for End User Programming (SEEUP '09)*, pages 1–7, 2009.

[25] Jonathan Lung, Jorge Aranda, Steve M. Easterbrook, and Gregory V. Wilson. On the difficulty of replicating human subjects studies in software engineering. In *Proc. of the 30th ACM/IEEE Intl. Conf. on Software Engineering (ICSE'08)*, 2008.