The *HybrEx* Model for Confidentiality and Privacy in Cloud Computing

Steven Y. Ko[†], Kyungho Jeon[†], Ramsés Morales[*]
[†]*University at Buffalo, The State University of New York and* [*]*Xerox Research Center Webster*

Abstract

This paper proposes a new execution model for confidentiality and privacy in cloud computing, called the *HybrEx* (*Hybr*id *Ex*ecution) model. The HybrEx model provides a seamless way for an organization to utilize their own infrastructure for sensitive, *private* data and computation, while integrating public clouds for non-sensitive, *public* data and computation. We outline how to realize this model in one specific execution environment, MapReduce over Bigtable.

1 Introduction

Cloud computing allows large organizations to tap into a virtually infinite pool of resources with the ability to control cost. Cloud providers give this power to their customers by offering a pay-as-you-go pricing model as well as elasticity and availability of computing and storage resources. Due to this benefit, cloud computing has quickly gained popularity in recent years.

Yet many organizations have not widely adapted the use of clouds due to the concerns of *confidentiality* and *privacy*. For example, a recent survey collected responses from more than 500 IT executives from around the world, and reports that IT executives prefer their existing internal infrastructure (i.e., their *private cloud*) over a third-party cloud (i.e., a *public cloud*) due to security threats and lack of control over their data and systems that handle it [12]. For industries such as finance and healthcare, explicit regulations regarding data protection — Payment Card Industry Data Security Standard (PCI DSS) [13] and Health Insurance Portability and Accountability Act (HIPAA) [9] — severely limit the potential use of public clouds.

These concerns are well-founded. Researchers have shown that an outside attacker can extract unauthorized information in Amazon EC2 [15]. Other researchers discovered a vulnerability that allows user impersonation in Google Apps [1]. Encryption can only provide a limited guarantee, since any computation on encrypted data either involves decrypting the data or has yet to be practical even with fully homomorphic encryption [7].

One line of research to resolve these issues is to make public clouds more secure [17, 20]. However, a public cloud is a shared platform managed by a third-party with potential security risks such as insider attacks and software vulnerabilities. These risks are difficult to eliminate as evidenced by the examples mentioned above as well as the history of security breaches and patches in general — after all, this is precisely the reason why organizations are hesitant to adapt the use of public clouds.

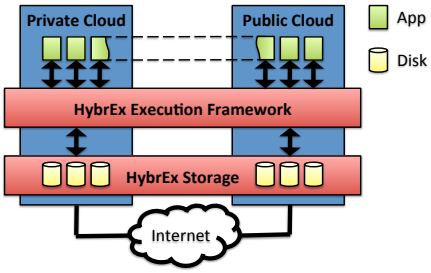
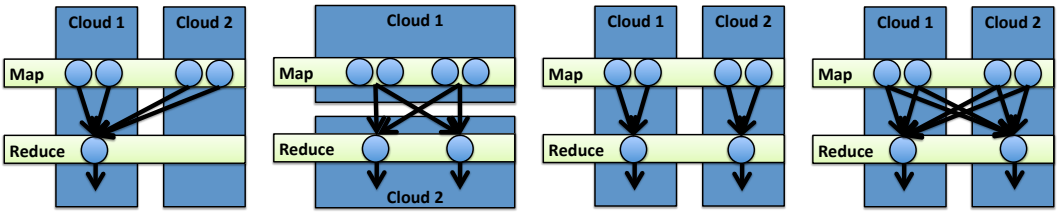


Fig. 1: The Architecture of the HybrEx Model

Recognizing this difficulty, we argue for an alternative that treats public clouds as an *inherently* insecure environment instead of trying to make them more secure; we propose an execution model that utilizes public clouds only for *safe* operations while *integrating* an organization's private cloud. We refer to this as the *HybrEx* (*Hybr*id *Ex*ecution) model.

More specifically, our HybrEx model utilizes public clouds only for an organization's non-sensitive data and computation classified as *public*, i.e., when the organization declares that there is no privacy and confidentiality risk in exporting the data and performing computation on it using public clouds. For the organization's sensitive, *private* data and computation, the HybrEx model utilizes their private cloud. Moreover, when an application requires access to both the private and public data, the application itself also gets partitioned and runs in both the private and public clouds. Figure 1 depicts the architecture with a HybrEx execution framework that partitions and runs applications, as well as a HybrEx storage that manages private and public data separately.

The main benefit of the HybrEx model is *integration with safety*, i.e., the ability to add more computing and storage resources from public clouds to a private cloud without the concerns for confidentiality and privacy. By partitioning data and computation, the HybrEx model side-steps the question of trustworthiness of pub-



(a) Map Hybrid (b) Horizontal Partitioning (c) Vertical Partitioning (d) Hybrid

Fig. 2: Execution Categories for HybrEx MapReduce

lic clouds and provides the same level of confidentiality and privacy guarantees as the traditional local computing provides. Businesses are already looking into the integration of private and public clouds mainly for capacity and performance, i.e., to elastically scale out from their private cloud to public clouds [14]; the HybrEx model can give an additional benefit of confidentiality and privacy to these businesses.

In order to concretely explore this general direction, we first focus our effort on how to realize the HybrEx model in one specific execution environment, *MapReduce over Bigtable*, using Hadoop MapReduce and HBase. We have chosen this execution environment for two reasons. First, MapReduce [4] is arguably the most popular execution environment in cloud computing. Second, MapReduce's massively parallel nature of execution, combined with the semi-structured data management of Bigtable [2], allows clean and well-defined partitioning between private and public clouds.

Using partitioning for secure computing is not a new idea [3, 21, 22]. However, realizing it in the HybrEx model and MapReduce brings its own set of challenges. First, we need to be able to partition data and computation. Second, utilizing both private and public clouds for a single (partitioned) MapReduce job means running the job over the wide-area Internet. Due to the all-to-all communication pattern in MapReduce as well as its master-slave architecture, providing reasonable performance over the wide-area can be challenging. Finally, while partitioning provides confidentiality and privacy for private data and computation, it does not ensure integrity of public data and computation. Section 3 outlines these challenges and our research directions.

The HybrEx model enables new kinds of applications utilizing both private and public clouds. We discuss this in the context of MapReduce in the next section.

2 Execution Categories

MapReduce presents a unique opportunity to realize the HybrEx model since a MapReduce job is sub-divided into tasks that run massively in parallel. We present four categories (Figure 2) showing how HybrEx MapReduce enables new kinds of applications that utilize both private and public clouds. These categories highlight both the *integration* and *safety* aspects of the HybrEx model.

Map Hybrid Many MapReduce applications analyze private and public data sets; a number of public data sets are available for domains such as forensic analysis, spam detection, and genome analysis, via well-known repositories, e.g., Amazon AWS Public Data Sets and CDC WONDER. Using these data sets, organizations can analyze both their own data sets and public data sets together for comparison or accuracy improvements. Since these applications process both private and public data sets, it is difficult to execute them in a public cloud without compromising on confidentiality and privacy.

HybrEx MapReduce enables this type of applications to safely utilize a public cloud by executing the Map phase in both private and public clouds while executing the Reduce phase in only one of the clouds. We refer to this category as *Map hybrid*, depicted in Figure 2(a).

We illustrate this with an example called CloudBurst [18], a bioinformatics MapReduce application. Simply put, it implements an algorithm that compares genome sequences, which serves as a basis for other biological analyses such as comparing a patient's genome to a reference human genome for medical purposes. CloudBurst's input consists of (potentially) private data called *target genomes* (e.g., patients' genomes) and public data called *reference genomes* (e.g., human reference genomes available from a public repository).

CloudBurst uses the same algorithm to process both the reference genomes and the target genomes in the Map phase. Thus, HybrEx MapReduce can safely utilize the public cloud for the reference genomes while utilizing the private cloud for the target genomes during the Map phase. On the other hand, the Reduce phase of CloudBurst produces (potentially) private outputs, e.g., a genome comparison result for a patient. Thus, we can utilize only the private cloud for the Reduce phase.

Horizontal Partitioning There are two popular usage cases for the current public clouds. The first case is long-term archiving of an organization's data, where the organization encrypts their private data before storing it on a public cloud. The second case is exporting and importing data for running periodic MapReduce jobs in a public cloud. This is in fact a common usage case in Amazon Elastic MapReduce [19]; an organization first exports its data to a public cloud, runs a MapReduce job

in the public cloud, and (optionally) imports the result back to its own private storage.

Since HybrEx MapReduce seamlessly integrates private and public clouds, it can automate these usage cases by utilizing different clouds for different phases. We refer to this category as *horizontal partitioning*, depicted in Figure 2(b). For example, in the long-term archiving case, HybrEx MapReduce can run Map tasks that encrypt private data in the private cloud, transfer encrypted data to the public cloud via the Shuffle phase, and run Reduce tasks that store the data in the public cloud.

Vertical Partitioning In 2007, the New York Times transformed its scanned images of the public domain articles to PDF files using MapReduce running in Amazon EC2. [1] Each original article comprised of many small TIFF images, and a MapReduce job "glued" these images together to produce one PDF file per article. However, they only transformed public domain articles; if an organization wants to process private and public documents at the same time, it becomes difficult to do so in a public cloud due to confidentiality and privacy.

HybrEx MapReduce enables this type of applications to safely utilize a public cloud by executing a MapReduce job in both private and public clouds while avoiding any inter-cloud shuffling of intermediate data. Although this type of partitioning is technically akin to running two separate jobs in private and public clouds, HybeEx MapReduce supports this naturally without the overhead of separate management of jobs and data. We refer to this category as *vertical partitioning*. Figure 2(c) depicts this category.

In vertical partitioning, HybrEx MapReduce workers in the public cloud execute Map and Reduce tasks using public data as the input, shuffle intermediate data among them, and store the result in the public cloud. Workers in the private cloud do the same with private data. In general, HybrEx MapReduce can run a MapReduce job this way when the job can process private and public data in isolation. Many applications belong to this category such as separate indexing of private and public web pages of an organization, pattern search (grep), etc.

Hybrid Many organizations are looking into ways to integrate public clouds for performance reasons or due to resource limitations in their private cloud [14]. These organizations mainly utilize their private cloud even for storing and processing public data, but occasionally want to scale out to a public cloud. HybrEx MapReduce can achieve this by utilizing both private and public clouds in all three phases of MapReduce as shown in Figure 2(d). We refer to this simply as *hybrid*.

[1] http://open.blogs.nytimes.com/2007/11/01/self-service-prorated-super-computing-fun/

Beyond Pure MapReduce For general applications, the pure MapReduce programming paradigm is limiting as there are only two primitives to play with. Thus, we are exploring ways to overcome this limitation. The first is an optional phase in between Map and Shuffle called the *Sanitize* phase. Using this phase, HybrEx MapReduce could allow programmers to explicitly sanitize intermediate data for more flexible utilization of private and public clouds.

We are also exploring the applicability of HybrEx in Pig [2] and Hive [3]. Both systems are based on MapReduce, but allow more flexbility and generality in programming. We believe that supporting these systems will likely result in wider applicability of HybrEx.

3 Research Challenges and Directions

We need to overcome three main challenges to implement the HybrEx model — data partitioning, system partitioning, and integrity — and achieve *integration with safety*. The following outlines these challenges and our research directions.

3.1 Data Partitioning

Since the HybrEx model proposes the use of partitioning for confidentiality and privacy, it raises an immediate question of how to partition data. We address this question by using two *labels*, the *private* label and the *public* label. HybrEx Bigtable and HybrEx MapReduce recognize these labels and determine data and computation placement accordingly. This use of labels is inspired by previous information flow control techniques used in programming languages such as Jif [11] and systems such as Asbestos [5]. While these approaches use labels to control and track how information flows among system components, we only need to use labels to determine the placement of data and computation.

We assume that organizations have policies to determine data sensitivity; we hypothesize that in many cases, organizations can perform labeling programmatically at the time of importing data into HybrEx Bigtable by running a MapReduce job. For example, if an organization has a file-naming convention that indicates the sensitivity of a document, it can run a MapReduce job that labels data according to file names.

3.2 System Partitioning

Since the HybrEx model utilizes both private and public clouds, any system that implements the HybrEx model has to partition its components, i.e., it needs to place some components in the public cloud and others in the private cloud. This naturally raises two sub-questions — i) how to keep public components from accessing

[2] http://pig.apache.org/

[3] http://hive.apache.org/

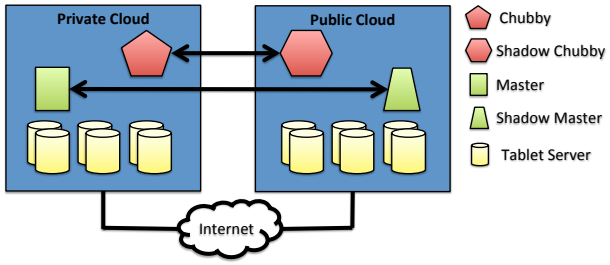


Fig. 3: The Architecture of HybrEx Bigtable

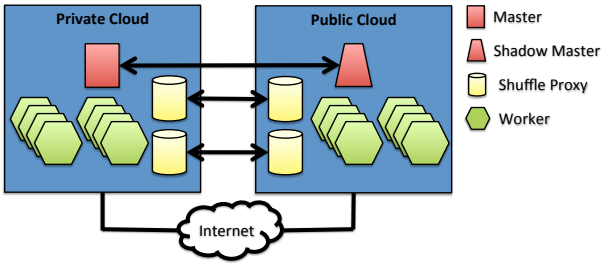


Fig. 4: The Architecture of HybrEx MapReduce

unauthorized information (e.g., private data), and ii) how to reduce the wide-area communication overhead if the communications between private and public components are necessary and become a bottleneck. We discuss how we are addressing these issues in HybrEx Bigtable and HybrEx MapReduce.

HybrEx Bigtable In order to concretely discuss the two questions of system partitioning for Bigtable, we briefly introduce its three main components — the master, tablet servers, and Chubby. The master is responsible for the overall management of the system. Tablet servers manage stored data and handle read and write requests. Chubby provides meta data consistency and reliability.

With these components, we can rephrase the two questions of system partitioning as follows. First, when utilizing both private and public clouds, we need to place tablet servers in both clouds as they directly handle data. Thus, we must keep public tablet servers from handling private data. Second, we avoid placing the master and Chubby in the public cloud as we need to trust them for overall correctness. However, the master, tablet servers, and Chubby communicate with each other frequently for correct functioning of the system. Thus, we need to reduce the wide-area communication overhead between public tablet servers, the master, and Chubby.

In order to address these issues, we introduce two new components that we refer to as the *shadow master* and the *shadow Chubby* as shown in Figure 3. These components are public counterparts to the master and Chubby that run in the public cloud. Each shadow component is a restricted version of its private counterpart, and does not have any access to information regarding private data. Moreover, we only allow tablet servers in the public cloud to communicate with these shadow components. An immediate benefit of this architecture is the ability to avoid the master-slave wide-area communications, as it mostly localizes the communications among system components; the components in the private cloud do not communicate with the components in the public cloud for the most part, and vice versa.

HybrEx MapReduce Since both Bigtable and MapReduce have a similar master-slave architecture, we apply the same general approach to system partitioning; we partition the main components of the original MapReduce, the master and the workers. The master has its restricted public counterpart, the *shadow master*, and we place the workers in both private and public clouds as shown in Figure 4. As in HybrEx Bigtable, this architecture avoids the master-slave wide-area communications.

However, MapReduce has a critical difference from Bigtable in terms of wide-area communication overhead as some MapReduce jobs require shuffling of intermediate data over the wide-area as shown in Section 2. Thus, we introduce a new component called *shuffle proxies* that transfer intermediate data over the wide-area on behalf of the workers. This is different from the original MapReduce that allows the workers to directly transfer intermediate data among them.

Shuffle proxies give us the benefit of having a separate architectural component where we can apply optimization techniques to reduce the wide-area overhead. We are exploring techniques such as caching, aggregation, compression, and deduplication of intermediate data.

We recognize that using a public cloud in addition to a private cloud gives more computing power, which can lead to better performance despite the wide-area overhead. For example, in a small-scale experiment involving 5 local machines in Buffalo and 5 Emulab machines in Utah, the execution time of Hadoop sorting 20GB of input data turns out to be faster over the wide-area (702 sec with all 10 machines over the wide-area vs. 937 sec with 5 local machines). However, overhead reduction is still important to benefit a broad range of applications and system configurations.

3.3 Integrity

The last question is how to provide integrity for data and computation in the public cloud, as our basic assumption is that it is not safe to trust the public cloud. To address the question of data integrity, HybrEx Bigtable keeps the hashes of the public data in the private cloud. HybrEx Bigtable can verify the integrity either when there is a request for the public data or proactively by sampling.

For computation integrity, HybrEx MapReduce checks the integrity of the results from the public cloud in two modes that provide different levels of fidelity. The first mode is *full* integrity checking, where the private cloud re-executes every Map and Reduce task that

the public cloud has executed. HybrEx MapReduce provides this mainly as a means to enable auditing at a later time, e.g., when an organization wants to verify the correctness of past computations from the public cloud.

Obviously, the overhead of doing the full integrity checking can be costly. Thus, HybrEx MapReduce provides *quick* integrity checking, where the private cloud selectively checks the integrity of the results from the public cloud. HybrEx MapReduce provides this mainly to check the integrity at runtime for probabilistic detection of suspicious activities in the public cloud. For this purpose, we store data items that we call *inspection points* in the private cloud. Inspection points can be either new synthetic data items that we add to the public data or existing public data items selected randomly for the purpose of verification. For example, for a MapReduce job that counts words in a public document, we can either add new unique words to the document or select existing words at random from the document, and store them in the private cloud. We can verify that the result from the public cloud contains the accurate counts of these words by running the same job in the private cloud with the inspection points. The frequency, overhead, and effectiveness of inspection points are our current subjects of investigation.

4 Related Work

In addition to the previous work on cloud security, partitioning, and information flow control discussed in Section 1 and 3, the recent line of work on hybrid clouds, cloud accountability, security and privacy in MapReduce, and untrusted public clouds is closely related. Notably, CloudNet [20] proposes a VPN-like network to provide secure and seamless resource integration between private and public clouds. The work on accountable virtual machines (AVM) [8] proposes an accountability scheme for virtualized environments. Since the AVM approach is basically VM logging-and-replaying, it is effectively the same as our full integrity checking, potentially with more overhead. Airavat [16] provides security and privacy guarantees in MapReduce by mandatory access control and differential privacy. Recent works such as Depot [10] and SPORC [6] also assume public clouds as an untrusted environment.

5 Conclusions and Future Work

In this paper, we start from the position that it is fundamentally difficult to secure public clouds, and then outline an execution model called the HybrEx model that uses partitioning of data and computation as a way to provide confidentiality and privacy. We discuss how we can realize this model in one specific execution environment, MapReduce over Bigtable.

We are currently implementing HybrEx MapReduce and HybrEx Bigtable. Moving forward, we believe that the HybrEx model could be useful for other execution environments such as VM-based hybrid clouds and smartphones interacting with clouds. We plan to explore the feasibility of applying the HybrEx model in these environments.

References

[1] A. Armando et al. Formal Analysis of SAML 2.0 Web Browser Single Sign-On: Breaking the SAML-based Single Sign-On for Google Apps. In *ACM FMSE*, 2008.

[2] F. Chang et al. Bigtable: A Distributed Storage System for Structured Data. In *USENIX OSDI*, 2006.

[3] S. Chong et al. Secure Web Applications via Automatic Partitioning. In *ACM SOSP*, 2007.

[4] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *USENIX OSDI*, 2004.

[5] P. Efstathopoulos et al. Labels and Event Processes in the Asbestos Operating System. In *ACM SOSP*, 2005.

[6] A. J. Feldman et al. SPORC: Group Collaboration using Untrusted Cloud Resources. In *USENIX OSDI*, 2010.

[7] C. Gentry. Fully Homomorphic Encryption Using Ideal Lattices. In *ACM STOC*, 2009.

[8] A. Haeberlen et al. Accountable Virtual Machines. In *USENIX OSDI*, 2010.

[9] Health Insurance Portability and Accountability Act of 1996 (HIPAA), Public Law 104-191.

[10] P. Mahajan et al. Depot: Cloud Storage with Minimal Trust. In *USENIX OSDI*, 2010.

[11] A. C. Myers and B. Liskov. A Decentralized Model for Information Flow Control. In *ACM SOSP*, 1997.

[12] Survey: Cloud Computing 'No Hype', But Fear of Security and Control Slowing Adoption. `http://www.circleid.com/posts/20090226_cloud_computing_hype_security`.

[13] PCI DSS v2.0. `https://www.pcisecuritystandards.org/documents/pci_dss_v2.pdf`, 2010.

[14] Forecast for 2010: The Rise of Hybrid Clouds. `http://gigaom.com/2010/01/01/on-the-rise-of-hybrid-clouds`, 2010.

[15] T. Ristenpart et al. Hey, You, Get Off of My Cloud: Exploring Information Leakage in Third-Party Compute Clouds. In *ACM CCS*, 2009.

[16] I. Roy et al. Airavat: Security and Privacy for MapReduce. In *USENIX NSDI*, 2010.

[17] N. Santos et al. Towards Trusted Cloud Computing. In *USENIX HotCloud*, 2009.

[18] M. C. Schatz. CloudBurst: Highly Sensitive Read Mapping with MapReduce. *Bioinformatics*, 25(11):1363–1369, 2009.

[19] P. Sirota. Keynote: Making Hadoop Enterprise Ready with Amazon Elastic MapReduce. Hadoop Summit, 2010.

[20] T. Wood et al. The Case for Enterprise-Ready Virtual Private Clouds. In *USENIX HotCloud*, 2009.

[21] S. Zdancewic et al. Untrusted Hosts and Confidentiality: Secure Program Partitioning. In *ACM SOSP*, 2001.

[22] L. Zheng et al. Using Replication and Partitioning to Build Secure Distributed Systems. In *IEEE Oakland*, 2003.