POSTER: SecureMR: Secure MapReduce Using Homomorphic Encryption and Program Partitioning

Yao Dong
Computer Science Department
Rensselaer Polytechnic Institute
dongy6@rpi.edu

Ana Milanova
Computer Science Department
Rensselaer Polytechnic Institute
milanova@cs.rpi.edu

Julian Dolby
IBM Thomas J. Watson Research Center
dolby@us.ibm.com

Abstract

In cloud computing customers upload data and computation to cloud providers. As they cede their data to the cloud provider, they may cede data confidentiality. We develop SecureMR, a system that analyzes and transforms MapReduce programs to operate over encrypted data. SecureMR makes use of partially homomorphic encryption and a trusted client. We evaluate SecureMR on a set of MapReduce benchmarks.

1 Introduction

Cloud service providers such as Google Cloud Platform and Amazon Web Service offer a wide range of data storage and computation products. When customers outsource data and computation to such cloud providers, they may cede data confidentiality. In this paper, we address the question: Can customers take advantage of inexpensive, efficient, and convenient cloud services, while preserving (to certain extent) the confidentiality of their data?

One approach is to use Partially Homomorphic Encryption (PHE) which allows computation on ciphertext. Customers upload PHE encrypted data, and programs that operate on ciphertext, thus preserving data confidentiality. PHE is limited in the sense that each PHE cryptosystem supports a single operation on ciphertext. The principal problem is data *conversion* — when data is computed using one encryption scheme but requires an operation that is not supported, data must be converted into a new scheme that supports the operation. One way to address the problem of conversion is to maintain a *trusted client* [1, 2, 4]. The client stores the cryptographic keys and performs conversion. It can also run segments of the program to alleviate conversion cost.

This paper presents SecureMR, a system that enables secure MapReduce computation on untrusted cloud servers using PHE and a trusted client. SecureMR makes use of five cryptosystems: randomized encryption (RND, supports no operations), additively homomorphic encryption (AH, supports addition/subtraction), multiplicatively homomorphic encryption (MH, supports multiplication/division), deterministic encryption (DET, supports equality checking) and order-preserving encryption (OPE, supports comparison).

2 Overview

The MapReduce paradigm supports large-scale parallel data analysis. A MapReduce program contains a *map* function and a *reduce* function, which are usually short (about 100 LOC), and exhibit predictable control flow. In addition, memory references tend to be easily resolved, and there are usually few calls to libraries that can be inlined. We observe that MapReduce applications are amenable to classical precise static analysis such as Reaching definitions (RD). Below is an overview of SecureMR.

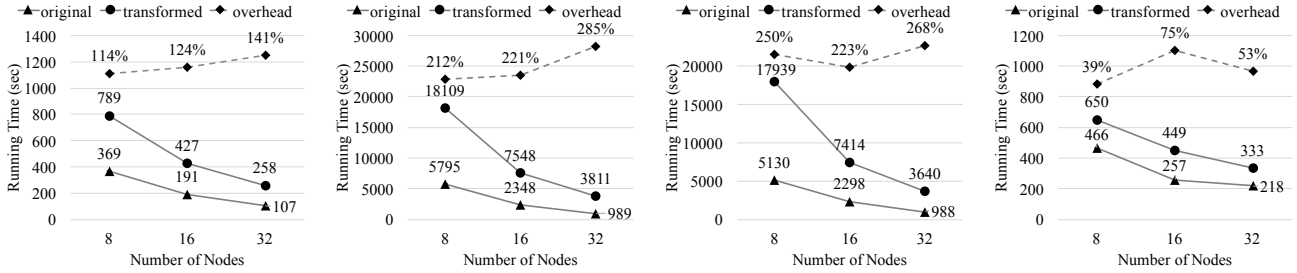
First, SecureMR infers the encryption schemes for input data, based on the involved operations. We start by annotating variable definitions that are read directly from input data. Reaching definitions analysis links the *definition* of a variable to the *uses* of that variable. There must be an encryption for the operation at each use.

When the same data is involved in different operations, the program needs *conversion* from one encryption scheme to another. Conversion entails *communication* with the trusted client during program execution. Specifically, a conversion consists of the following steps: (1) the server sends the ciphertext (e.g., AH) to the client, (2) the client decrypts it, and encrypts it into the new scheme (e.g., MH), (3) the client sends the new ciphertext back to the server. Clearly, communication is costly, and we must minimize its impact. One key insight of our work is that conversion placement matters. We frame the problem in terms of the classical Min-cut/Max-flow problem. Given a def-use pair (d, u) that requires conversion, we place conversions at the Min-cut edges on the graph from d to u. This achieves two purposes: (1) it covers all control-flow paths from d to u, and (2) it guarantees minimal number of executions of the required conversion.

With the optimal conversion placement, the program can run on the server initiating communication with the client at each conversion edge. However, this approach may turn prohibitively expensive. It would be more efficient to extract a short segment with high conversion density, and run the

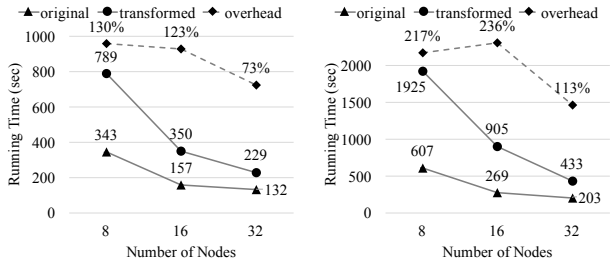
PPoPP '18, February 24–28, 2018, Vienna, Austria

ACM ISBN 978-1-4503-4982-6/18/02.
https://doi.org/10.1145/3178487.3178520



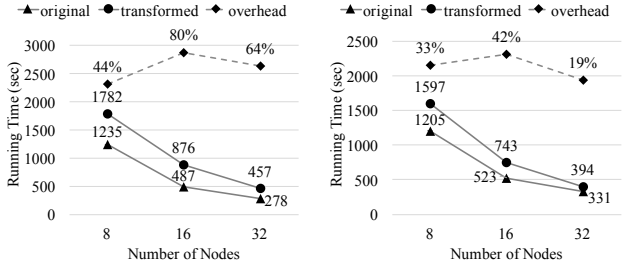
(a) Histogram Movies **(b)** K-Means **(c)** Classification **(d)** Join Task

Figure 1. Running time of benchmarks with conversion.



(a) L15 **(b)** Aggregate Task

Figure 2. Computation-intensive benchmarks.



(a) Word Count **(b)** Inverted Index

Figure 3. Search-intensive benchmarks.

entire segment on the client in plaintext form. SecureMR implements a simple heuristic: we extract the minimal segment that contains all conversions to run on the client.

3 Experiments and Results

We created 3 separate clusters on the Google Cloud Platform with 8, 16 and 32 nodes, as the *untrusted server*. The *trusted client* is a local Ubuntu machine which is used to encrypt the input files, store public/private keys, perform conversions and transform the programs into server and client partitions.

We use 3 MapReduce benchmark suites, PUMA[1], Brown[2] and PIGMIX2[3], a total of 36 programs. Based on our RD analysis, 4 benchmarks require conversion: 3 happen in map and 1 happens in reduce. We classify the benchmarks into two categories: *computation-intensive* (require AH and/or MH) and *search-intensive* (require DET and/or OPE only). Based on our RD analysis, there are 13 computation-intensive benchmarks, including the 4 programs that require conversion. MrCrypt [3] does not handle conversion, and does not evaluate computation-intensive benchmarks experimentally.

Figs. 1, 2 and 3 present the results of representative benchmarks. Each subfigure reports (1) the running time of the original program over plaintext, (2) the running time of the transformed program over ciphertext, and (3) the overhead percentage, on the 8, 16, and 32-node clusters.

In Fig. 1, the maximal overhead is 285% and the minimal overhead is 39%. In Figs. 1a, 1b, and 1c, conversion happens in map, while in 1d, it happens in reduce. Map in the first three benchmarks is called 540 times the reduce function in the fourth benchmark, and each function call is bound to one round trip communication with the client. Thus, we observe significantly larger overhead in Figs. 1a, 1b, and 1c than in Fig. 1d. Fig. 2 shows two computation-intensive benchmarks without conversion. The overhead is generally smaller compared to Fig. 1. They show benefits from increased parallelism, while Fig. 1 shows substantially less. As expected the overhead on search-intensive benchmarks (Fig. 3) is comparable to the one reported by MrCrypt [3].

[1]https://engineering.purdue.edu/~puma/pumabenchmarks.htm
[2]http://database.cs.brown.edu/projects/mapreduce-vs-dbms
[3]https://cwiki.apache.org/confluence/display/PIG/PigMix

4 Conclusions

We presented SecureMR, a system that provides data confidentiality for MapReduce applications running on untrusted clouds. SecureMR statically analyzes MapReduce programs and infers encryption schemes for input data. Furthermore, it provides optimal conversion placement, and a cost model that guides program partitioning. We evaluated SecureMR on 15 MapReduce benchmarks (13 computation-intensive and 2 search-intensive) running on the Google Cloud.

References

[1] Yao Dong, Ana Milanova, and Julian Dolby. 2016. JCrypt: Towards Computation over Encrypted Data. In *PPPJ'16*. 8:1–8:12.

[2] Meelap Shah et al. 2012. Language support for efficient computation over encrypted data. In *Off the Beaten Track Workshop: Underrepresented Problems for Program. Lang. Researchers.*

[3] Sai Deep Tetali et al. 2013. MrCrypt: Static Analysis for Secure Cloud Computations. In *OOPSLA'13*. 271–286.

[4] Stephen Tu et al. 2013. Processing analytical queries over encrypted data. In *Proc. 39th Int. Conf. Very Large Data Bases*. 289–300.