Test driving Spark SQL with Intel SGX on encrypted data

Gidon Gershinsky Follow
Nov 8, 2017 · 6 min read

Protecting sensitive business and personal information is a central requirement when enterprises move to the cloud. Many aspects of this requirement are already handled at various levels. Data-at-rest can be secured in cloud stores by encrypting it before storage, while data-in-flight is transmitted on protected channels such as TLS and HTTPS. Data-in-use, processed in cloud compute nodes, is kept in isolated virtual machines or containers. And now, a new generation of secure hardware technology (Intel SGX, AMD SME/SEV) provides additional protection of data processing in public clouds, by making the process memory inaccessible to malicious privileged software or system administrators.

Opaque is a novel analytics framework that uses such hardware support for secure SQL queries on sensitive data in the cloud. This open source technology from Berkeley RISE Lab integrates the Apache Spark SQL engine with Intel Software Guard Extension (SGX) hardware, and offers data protection by running SQL transformations inside trusted enclaves. SGX works by encrypting data in the CPU SoC (system on a chip) as the data moves to the main memory. The Opaque research project has an additional security layer, called “oblivious computing mode”, which prevents leakage of data access patterns. However, this mode is not released to the open source repository.

Opaque uses the developer APIs of the Spark SQL Catalyst engine, and implements the SQL operators in C++, with a JNI interface between the Scala code and the native code running inside SGX enclaves. To run Opaque, you can use an out-of-the-box Spark cluster running on machines that are SGX-enabled. A client application should load the Opaque jar file (with Opaque classes and enclave binaries) to run the SQL statements securely on the Spark workers.

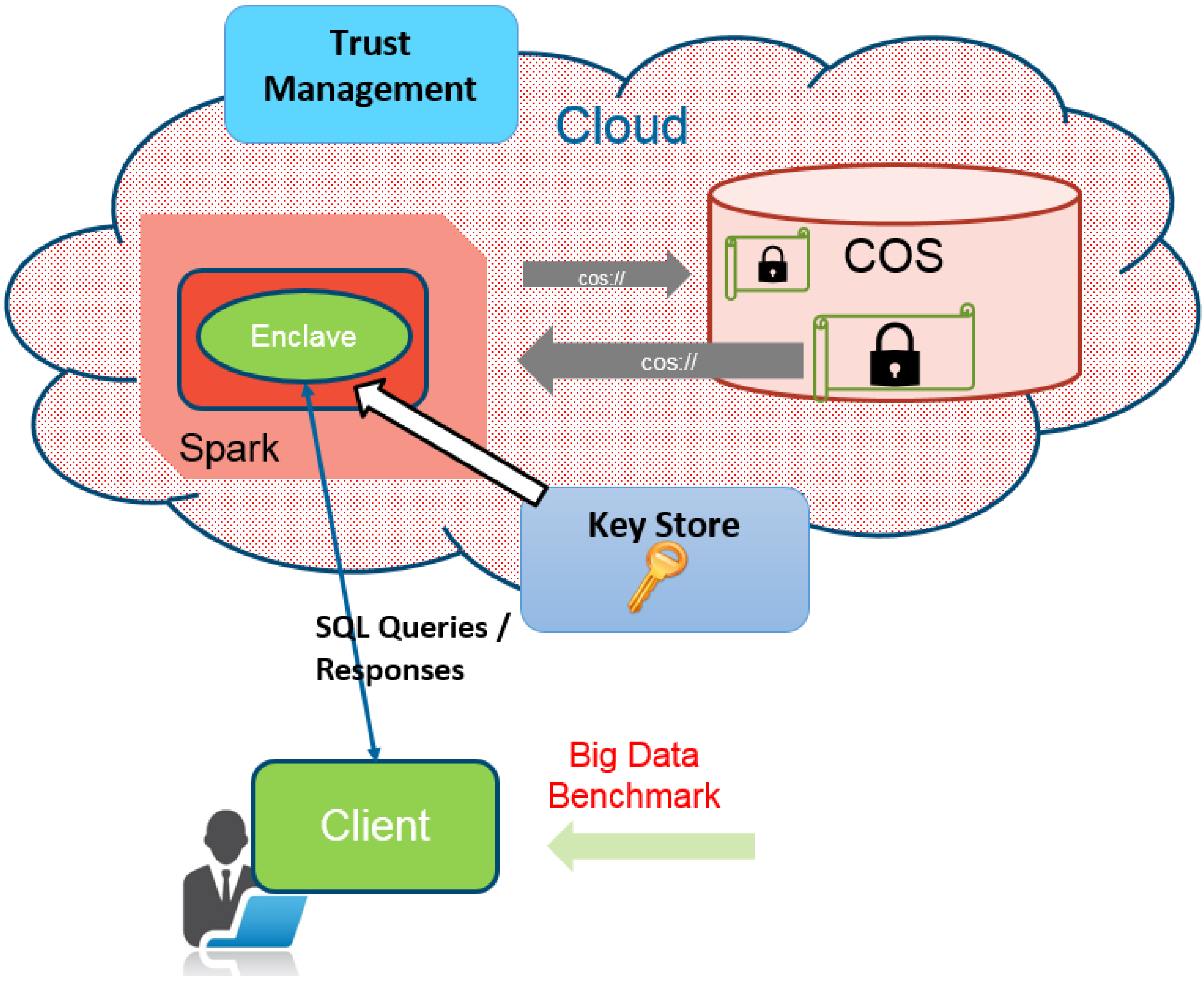
Currently Opaque can load encrypted data and perform several SQL functions on the data, including filtering, sorting, basic aggregation, and join operations. However, the framework is missing a few key mechanisms for secure data processing in SGX enclaves. The first is “remote attestation” , which verifies that the enclave binary code has not been tampered with, and runs on a genuine Intel SGX hardware. The published Opaque code has hooks for remote attestation, but the implementation is not complete. Moreover, the design assumes the Spark master is a trusted attestation broker, which prevents deploying the master in public clouds. The second missing mechanism is a system for delivery of data encryption keys to the software running in the Opaque enclaves. The current implementation assumes a fixed encryption key.

Our team at IBM Research is working on a number of SGX-related technologies. One of them is a **Trust management framework**, designed to ease the burden of SGX application development. The framework includes a service and toolkit that perform remote attestation, and assists in the delivery of secrets (such as encryption keys) to the SGX enclaves. The service simplifies integration of the applications running inside SGX enclaves with other applications and services in the cloud. The framework communicates with Intel Attestation Service (IAS) for the purpose of attestation, and provides a trusted key exchange mechanism between cloud services and the enclave applications. The toolkit allows SGX application developers to focus on their business logic, and get an “out of box” attestation with secret passing.

We integrated Opaque with our Trust service toolkit to enable remote attestation and data key delivery in Opaque enclaves. Internally, the code triggers interaction with the Trust service and IAS, which is transparent to Opaque code. We developed a simple key storage service that keeps the data encryption keys and uses the toolkit client APIs to deliver the key to Opaque enclaves after checking the enclave (and platform) validity.

Our team participates in a European Horizon2020 project, called RestAssured, where we plan to leverage Opaque with the Trust service for secure data processing in several application scenarios, defined by the project partners.

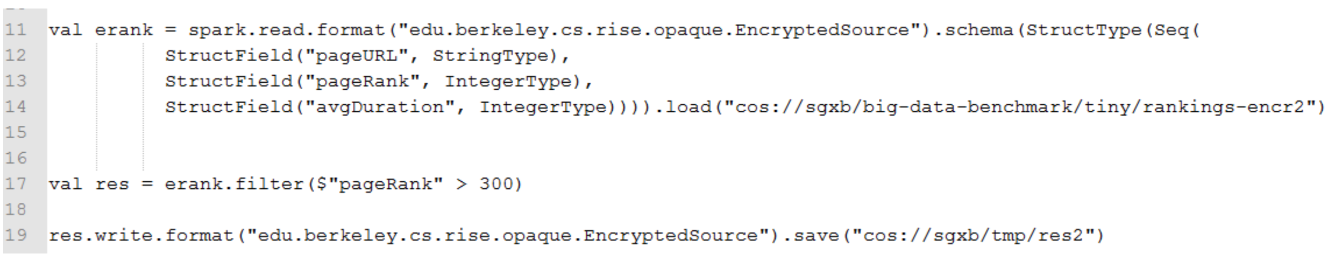
We demonstrated the modified Opaque in an end-to-end use-case using IBM Cloud Object Storage (COS), where the data — both the SQL input and output — is stored in an object store. We used an instance of Opaque that runs on a trusted host to encrypt and upload the data to the object store, as Opaque currently supports only its own (proprietary) encryption format. For a dataset, we started with a “rankings” table from the Big Data Benchmark (BDB).



Demonstration setup

The demonstration setup includes a Spark cluster running in an environment such as the IBM public cloud, a Trust service, a key storage service, the Stocator object connector, and the IBM object store. We run SQL queries via a Spark shell. The Opaque classes and binaries are packaged in a jar file, which is passed to the Spark shell as a command line argument.

```
import edu.berkeley.cs.rise.opaque.implicits._
import org.apache.spark.sql.types._

edu.berkeley.cs.rise.opaque.Utils.initSQLContext(spark.sqlContext)


sc.hadoopConfiguration.set("fs.stocator.cos.scheme", "cos")
sc.hadoopConfiguration.set("fs.cos.service.endpoint","s3-api.us-geo.objectstorage.softlayer.net")

```



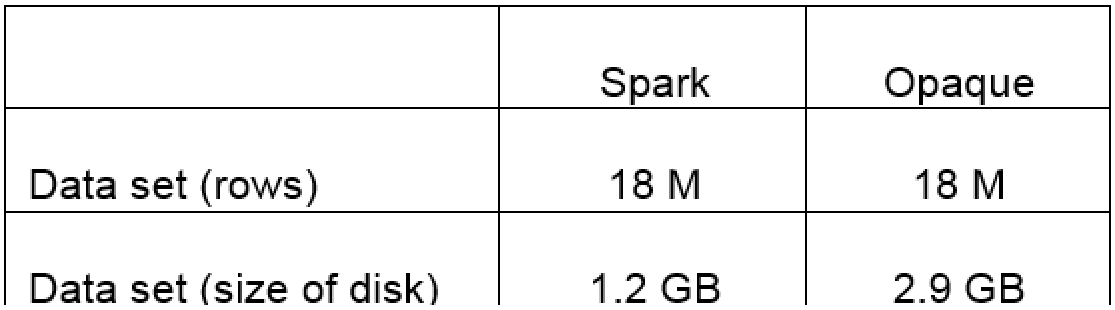
```
val erank = spark.read.format("edu.berkeley.cs.rise.opaque.EncryptedSource").schema(StructType(Seq(
        StructField("pageURL", StringType),
        StructField("pageRank", IntegerType),
        StructField("avgDuration", IntegerType)))).load("cos://sgxb/big-data-benchmark/tiny/rankings-encr2")


val res = erank.filter($"pageRank" > 300)

res.write.format("edu.berkeley.cs.rise.opaque.EncryptedSource").save("cos://sgxb/tmp/res2")
```

Scala code snippet performing a secure SQL query on the encrypted data

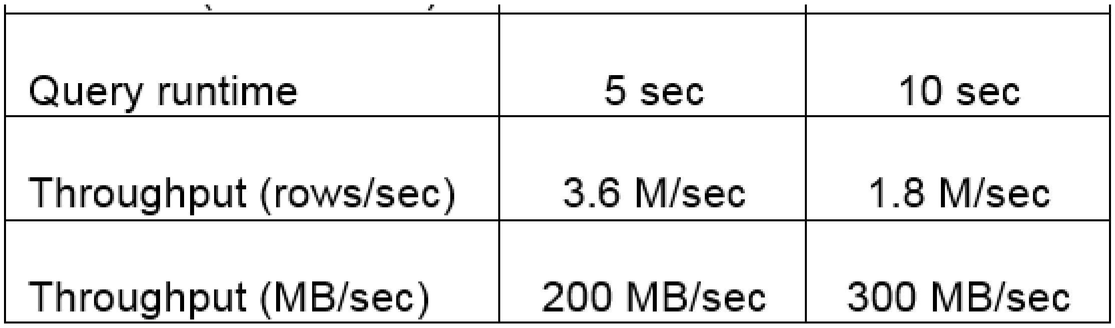
How does Opaque performance compare to that of the standard Spark SQL? Not only does Opaque run in SGX enclaves, it also works with encrypted data. This means it has to spend some cycles on AES decryption. A proper answer to that question requires running a comprehensive benchmark, such as BDB. We started with a quick performance test, to get a basic "feel" for Opaque processing throughput. The results were somewhat surprising.

We compared Opaque versus native SparkSQL running the Spark worker on the same hardware and storing the data locally on disk to eliminate access time to data and focus on the processing aspects. We worked with the 1.2GB "rankings" dataset from BDB. The native Spark SQL ran the filter query (*df.filter($"pageRank" > 3000).count*) in 5 seconds, indicating a processing throughput of ~ 200 MB/sec. Notice — BDB data is stored in CSV files, which is not the most efficient format among those supported in Spark. CSV ingest speed could affect the measured throughput.

We encrypted the 1.2GB "rankings" dataset using Opaque. This resulted in a 2.9GB dataset, almost 3 times larger than the original. Apparently, this is the result of the Opaque serialization mechanism, since encryption does not impact size dramatically. Running the same filter query (*df.filter($"pageRank" > 3000).count*) in Opaque with SGX enclaves using the encrypted data, takes around 10 seconds, corresponding to a higher processing throughput of ~ 300 MB/sec — but still 2 times longer than the standard Spark SQL.



	Spark	Opaque
Data set (rows)	18 M	18 M
Data set (size of disk)	1.2 GB	2.9 GB



Query runtime	5 sec	10 sec
Throughput (rows/sec)	3.6 M/sec	1.8 M/sec
Throughput (MB/sec)	200 MB/sec	300 MB/sec

In a separate test, we measured raw AES decryption using SGX enclaves. It runs at a rate of about 4 GB/sec, an order of magnitude faster than Spark SQL throughput with either standard Spark/CSV or with Opaque. This implies that SGX decryption may not be the bottleneck here.

What do the results above tell us about Opaque performance? Basically, they indicate that Opaque is already quite fast, and can perhaps be made as fast as the off-the-shelf Spark/CSV if the encryption format issued is resolved.

To summarize, Opaque is a promising approach to secure data processing in public clouds. Currently, it is a research prototype, with a few issues that need to be taken care of before the technology can be deployed for production use. Our integration with Trust services closed the key security gaps, resulting in a functioning proof-of-concept code that can run SQL queries with end-to-end protection for the data. We found several additional gaps and plan to perform a deeper analysis of the technology to identify other missing features required for cloud deployment.

Encryption Computer Security Apache Spark Cloud Computing IBM