Critique Paper 1: Executing SQL over Encrypted Data in the Database-Service-Provider Model

by Hacigümüş, Hakan, Bala Iyer, Chen Li, and Sharad Mehrotra
by Yafei Wang

Summary:

This paper mainly introduces a data privacy issue, that database service hosts are not trusted. In order to deal with this issue, authors provide a database framework which could execute as much queries as possible on the service side without decryption. In the first section, the importance of data privacy and security in ASP model are discussed. Also, previous works about encrypted data storage and query are shown as well. In the second part, the paper shows a new framework which could query encrypted data and do most computational work on the server side. Also, the paper shows the detail methods in the model: data encryption, encrypted data storage, relational operators in SQL and query spilition. In the last part, author evaluates the new model by utilizing TPC-H benchmark. Both simple query and joint query are tested in the experiment part. According to the results, the new protocol doesn’t increase executing cost significantly.

Comments:

The paper provides a good database-as-a-service framework for host is not trusted. In the framework, the authors create a new coarse index to store encrypted data. Also, the new index helps to decrypt data from data owner side. This protocol successfully deal with the problem which data hosts are not trusted by data owner. In addition, the new protocol doesn’t introduce significant cost overhead.

Query deposition in the paper reduces computational cost in the client side. Splitting query into host side and client side is a smart solution for query performance. The new method allows most query execution works in the host side. Only a small amount of computational work are in the client side. From the experiment results, the solution successfully reduces the query execution cost. It provides a good user experience especially in the database-as-a-service framework.

As paper discusses, coarse index id is generated by encrypted data. After each mapping, the system may return multiple data tuples with the same coarse index value. Thus, filtering the false hit is a major post-processing work after mapping. If the system encountered some complex queries, such as joint, post-processing workload in the client side may increase.

According to the experiment, authors show the performance of selection query and joint query with various number of buckets. In this case, the database in the service is not changed. If there
is a huge number of data insert into the database, what about the cost of insertion and query cost after insertion? Is there any automatic adjustment of number of bucket after insertion and deletion? For my prospective, authors may provide the experiments and discussion about data insertion and deletion as well.

**Question:**

As the experiment results, a larger number of buckets may reduce the client cost in query. What is the best bucket number?
Critique Paper 2: Human-powered Sorts and Joins

Adam Marcus, Eugene Wu, David Karger, Samuel Madden, Robert Miller
by Yafei Wang

Summary:
This paper introduces a system, Qurk, which integrates a crowd of workers into a declarative query processing workflow. The system may help crowdsourcing platform to optimize the parameters of crowd-based join and sort operators. In the first section, authors shows the importance of Qurk system on crowdsourcing platform. Also, the paper provides some previous work, like Berkeley and Stanford, about crowd-oriented database system. Authors give some query language overviews in the Qurk system. Generative sorts, joins, feature extraction and HIT generation are shown in this part. In the second part, paper introduces the detailed optimization techniques of join and sort operator. For join operator, paper implements feature filtering optimization method for reducing complexity. For sort operator, some different approaches are discussed as well. In the third part, some experiments are listed to compare the performance of different implementation methods.

Comments:

Compared with the Simple Join algorithm, two effective batching algorithms, Naive Batching and Smart Batching, are interesting. These two algorithms theoretically reduce the number of HITs in Qurk System. According to the experiment results, I find that two batching algorithm successfully reduce the latency time and cost. At the same time, batching algorithms doesn’t decrease significantly on accuracy. Thus, I think these two batching algorithms partially reach system goal.

For sorting, a harder problem, authors provide 3 successful alternatives in Qurk System with different emphasis. Theoretically, comparison-based sorting reaches the highest TAU with largest number of comparisons, while, rating-based sorting reaches the a reasonable TAU with a lower number of comparisons. The hybrid-based algorithm balance the TAU and number of HITs. It seems that the hybrid-based method is the best strategy. While, there are 3 algorithms in hybrid-based methods. Which one is the best one theoretically? How could I choose the hybrid algorithm? In addition, how could I optimize the parameters in each hybrid algorithm? If the paper could provide more information about the hybrid-based method, it would be better.

Even if the batching algorithms reduce number of HITs a lot, the cost of most batching algorithm is not acceptable if both two tables are large. As the paper shows, if the two tables being joined are R and S, the number of HITs for Naive Batching and Smart Batching are O(R*S/b) and O(R*S/rs), respectively. If the paper could provide one algorithm with lower number of HIT, it would be better.
During the experiments in the paper, all crowd workers are seen with the same ability in the system. While, it is unfair to treat the results with the same weight if correctness of one worker's tagging is higher than the other. The system should provide a weighted version in join and sort algorithms.

Questions:
The paper shows a good cost reducing system, Qurk, which runs on the top of MTurk. Does this system could run on the other crowdsourcing platform with small modification? If the authors could provide the adaptivity of Qurk, it would be better,