Towards Verifiable Metering for Database as a Service Providers

Min Du1; Ravi Ramamurthy2
1University of Utah, 2Microsoft Research

Abstract

**Metering** is an important component of cloud database services. However, there is no easy way to verify if the amount that a Database as a Service (DBaaS) provider is charging is in fact reasonable. We study potential problems in verifiability for existing DBaaS metering and initiate a proposal of how we can address this problem.

Database as a Service (DBaaS) Metering

Microsoft Azure SQL Database metering:

![Graph showing Azure SQL Database metering](image)

Unit: Database Transaction Units (DTUs)

-- maps to different ranges of CPU and IOPS utilization

**Question:** How does a client know that the bill is accurate for his workload and has not been tampered with?

**Metering Attacks**

**Example:** Volkswagen Emission Scandal

Late Discovery, (year 2008-2015, 11,000,000 cars)

The implicit assumption that the emission counters were reporting the correct values and this was never carefully verified.

**Question:** Could “similar” defeat devices be used to modify resource meters in a DBaaS service? Can such schemes be detected?

Potential Defeat Devices for DBaaS

**Direct attacks:** The catalog tables storing resource counters relevant to DTUs can be modified directly.

**Indirect attacks:** 1) Run additional workload concurrently; 2) Use inefficient query plan; …

Indirect attacks can significantly increase CPU/IOPS utilization, and hence DTU, and finally increase price level.

Trust, but Verify

If the client has a workload W of queries (e.g., the TPC-H query suite) which he is migrating to a DBaaS provider.

**Step 1:** Build a statistical profile (mean and standard deviation) of the appropriate metric (e.g., DTUs) by repeatedly running W locally.

**Step 2:** Use standard statistical hypothesis test to verify W on cloud.

Verifiable Metering

**Query Processing Units (QPUs) – #getNext()**

**Definition:** The work done by a query plan P is defined to be the total number of rows processed (i.e., sum of getNext() calls across all operators) in an execution plan.

**Remark:** Fix a query execution plan P for a query and a dataset, the QPUs of a query are invariant.

**Initial Results**

**Experiment Setup**

**Workload W:** the set of TPC-H queries, run 50 times to build a baseline

**Attacks:**


Attack 2 (A2): Limit number of CPU cores to a smaller number.

Attack 3 (A3): Pick non-optimal indexes.

Attack 4 (A4): Pick a non-optimal join order.

**DTUs usage**

While running W, record CPU/IOPS usage and upload the counters to an online DTU calculator: http://dtucalculator.azurewebsites.net

**QPUs usage**

Sum up the QPUs for each query, which is calculated by summing up the getNext() calls issued by all operators in the query plan.

**Summary**

We study the interesting problem of metering assuming an adversarial DBaaS provider and discuss a possible solution by exploring the notion of building baselines using verifiable metrics.

CONTACT

Min Du
University of Utah
mind@cs.utah.edu

Ravi Ramamurthy
Microsoft Research
ravirama@microsoft.com