Snowtrail: Testing with Production Queries on a Cloud Database

Jiaqi Yan[†], Qiuye Jin[†], Shrainik Jain[*], Stratis D. Viglas[‡], Allison Lee[†]

[†]Snowflake Computing, [*]University of Washington, [‡]Google

ABSTRACT

Database as a service provided on cloud computing platforms has been rapidly gaining popularity in recent years. The Snowflake Elastic Data Warehouse (henceforth referred to as Snowflake) is a cloud database service provided by Snowflake Computing. The cloud native capabilities of new database services such as Snowflake bring exciting new opportunities for database testing. First, Snowflake maintains extensive knowledge of historical customer queries, including both the query text and corresponding system configurations. Second, Snowflake is multi-tenant, which provides easy access to metadata and data that can be used to rerun customer queries from a privileged role. Furthermore, the elastic nature of Snowflake's data warehouse service allows testing with these queries using a separate set of resources without impacting the customer's production workload.

This paper presents Snowtrail, an infrastructure developed within Snowflake for testing using customer production queries with result obfuscation. Running tests with production queries provides us with direct insight into the impact of improvements and new features on customer workloads. It enables testing on queries of more shapes and complexity than can be manually constructed by developers. Snowtrail is also used to help ensure the stability of the online upgrade process of the system.

CCS CONCEPTS

• **Information systems** → **Database utilities and tools**; • **Software and its engineering** → *Software verification and validation*;

KEYWORDS

Database-as-a-service, Testing, Workload Selection, Automation, Snowtrail, Snowflake

ACM Reference Format:
Jiaqi Yan[†], Qiuye Jin[†], Shrainik Jain[*], Stratis D. Viglas[‡], Allison Lee[†]. 2018. Snowtrail: Testing with Production Queries on a Cloud Database. In *DBTest'18: Workshop on Testing Database Systems , June 15, 2018, Houston, TX, USA.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3209950.3209958

DBTest'18, June 15, 2018, Houston, TX, USA

ACM ISBN 978-1-4503-5826-2/18/06…$15.00
https://doi.org/10.1145/3209950.3209958

1 INTRODUCTION

1.1 The Snowflake Data Warehouse

Snowflake [1] is a multi-tenant cloud data warehouse offering a highly available, scalable and elastic database as a service. Snowflake has a multi-cluster, shared-data architecture and adopts a service-oriented design with three distinct layers where each layer is independently scalable: (*a*) the data storage layer, which utilizes the storage services provided by cloud computing platforms (e.g. AWS S3); (*b*) the virtual warehouses, which are abstractions of machine clusters that handle query execution; and (*c*) the cloud services layer, which consists of a collection of services for query compilation and optimization, transactions, security, virtual warehouses and other management tasks.

Snowflake supports ACID transactions via Snapshot Isolation on top of multi-version concurrency control (MVCC), and provides time travel functionality where queries can be executed as of a certain timestamp in the past as long as the relevant data is within the retention period.

1.2 Motivation

Snowflake's service-oriented design provides many new opportunities for testing. One of the key advantages to delivering a database as a service is the wealth of data available that can provide us with many insights into the system. This includes the history of all customer issued queries along with their configurations and related statistics. This historical data is stored in the Snowflake data warehouse, and we have made use of this data to build an array of tools that enable workload analysis tasks. The multi-tenant architecture of Snowflake also facilitates easy and secure access to metadata and data used in customer queries from a privileged role. The separation of compute and storage layers in Snowflake further enables running customer queries without impacting customer workloads.

We have created **Snowtrail** to take advantage of these capabilities to run customer queries to test Snowflake itself. To ensure security, the results of all queries run by Snowtrail are obfuscated. Whereas to avoid interference with customer workloads, Snowtrail reruns customer queries on isolated cloud services using separate, dedicated virtual warehouses. Testing with customer production queries gives us exact knowledge of the impact of a feature on the customer queries which allows us to gather feedback and make improvements before rolling it out to our customers.

There are various challenges to testing a fast-moving cloud database service. Snowflake runs tens of millions of customer queries per day and has an online upgrade process to ensure continuous availability. Therefore, Snowflake has adopted a weekly release cycle that enables fast feature delivery [1]. The quick turnaround time for each release means there is a relatively short time window available for testing on the release artifacts. For fast release cycles,

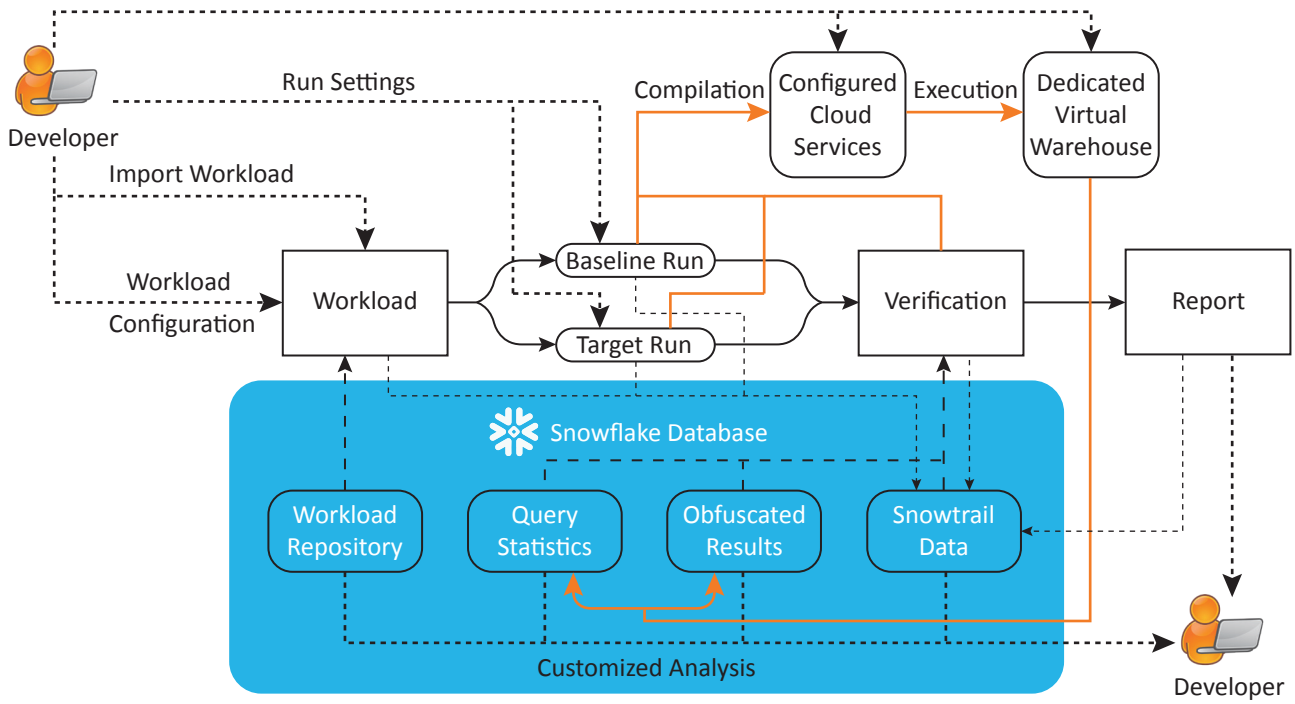

Figure 1: Feature testing workflow using Snowtrail

testing using customer queries on the new release before migrating actual customer workloads provides assurance of the stability and success of the release. Given the scale of customer queries submitted to Snowflake, however, it is prohibitively expensive to rerun all of them within a short time window. Therefore, an essential part of Snowtrail is workload sampling and selection to ensure that testing consumes a reasonable amount of time and resources.

1.3 Related Work

Oracle's SQL Performance Analyzer [8] is a feature from Oracle for testing impacts of planned changes such as upgrades, parameter changes, schema changes, etc. by executing queries under different settings. Snowtrail adopts a similar idea for running customer queries under different settings for and compare the runs. Oracle Database Replay [3] is another tool that can replay captured workloads to replicate the impact of the workload on the system. Snowtrail also provides a replay mode similar to Oracle Database Replay. A key difference of Snowtrail is that customers do not need to run these tests themselves; the tests are run by Snowflake internally without impacting customer production workloads. For release and feature testing, Snowtrail typically covers many customers within the same workload.

Microsoft's SCOPEPlayback [7] can replay the compilation and optimization phase of SCOPE jobs. Snowtrail's idea of of utilizing the job history for testing and validation is similar. Besides supporting an "explain" mode, Snowtrail focuses on executing customer queries, and is implemented as a generalized workload runner that supports many different ways to configure and replay workloads.

We also argue that workload selection is necessary to achieve efficient resource utilization for testing with production queries, and Snowtrail provides functionality to select and build workloads customized to the purpose of the test. Snowtrail provides verification and reports of finished runs, as well as a streaming mode that supports running queries that satisfy the testing criteria over an extended period of time.

The rest of the paper is organized as follows. Section 2 describes the design and implementation details of Snowtrail. Section 3 describes the usage of Snowtrail in production and presents some experimental results. In Section 4 we discuss some of the lessons learned from running Snowtrail in production and future work. We present some concluding remarks in Section 5.

2 DESIGN AND IMPLEMENTATION

Snowtrail is designed as a versatile workload runner with a focus on providing a tool for running production queries and detecting regressions. Snowtrail organizes queries into workloads that can be run under different settings and later be compared to analyze the differences in these runs. Snowtrail has three main functionality: workload selection, workload running, and result analysis. The workload selector creates the appropriate workload from the repository of historical customer queries for the purpose of the current test. The workload runner runs the selected workload with specified settings. Finally, the result analyzer takes in the results of multiple runs and compares them to report on differences or regressions. All useful data generated by Snowtrail is stored back in the Snowflake database.

2.1 Workflow

Figure 1 depicts a typical workflow of using Snowtrail for feature testing. The developer starts with building a workload of queries to run by either specifying the workload selection configuration or directly importing a workload to run. The workload selection configuration contains criteria for selecting queries best suited for testing the target feature. Snowtrail will then automatically build a workload from the given configuration.

After the workload has been built, the developer needs to specify the settings to use for running the workload. At least two sets of settings are needed, one for a baseline run and one for a target run. For feature testing, the two settings would be the same except for the feature flag. Snowtrail runs are decoupled from workloads so the same workload can be run any number of times with different settings. The developer then kicks off the baseline and target runs on dedicated virtual warehouses to ensure resource isolation. Once the runs are finished, Snowtrail performs verification of the results to eliminate false positives before generating a report for regressions. False positives mostly occur in performance comparisons, and can be caused by variations in the cloud environment.

All relevant data including query history, query statistics, obfuscated results, and metadata of the runs are stored in the Snowflake database and available to be queried using SQL. Developers can either drill down to investigate specific queries from the report, or choose to perform deeper analysis using customized SQL queries depending on the goal of the test.

2.2 Snowtrail Workloads

Snowtrail provides the functionality to automatically select a workload based on a set of criteria provided by the developer given the number of queries (sample size) in the workload. Snowtrail supports a wide variety of selection criteria including accounts and users that issued the queries, database and schema of the queries, the time window the original queries were run, matching query text fragments, matching execution configurations, and many others.

After deduplicating the queries that satisfy the developer-supplied criteria, Snowtrail applies a set of heuristics that further prunes the search space. Typically, a sizable percentage of queries in the system are very simple queries or ones that do not present much testing value. For example, "select 1" queries are constantly issued by some BI tools to verify they have a valid connection to the database. Also, queries selected for testing should only consume a reasonable amount of resources. Therefore, queries that are either too simple or too expensive are not considered.

For the remaining candidate queries, Snowtrail tries to select the subset that can achieve the maximum coverage according to a diversity measurement. Most production schemas tend to have fixed usage patterns, and it should be enough to pick a small number of representative queries for each query pattern.

We use Query2Vec [4] based workload summarization for selecting a diverse workload sample. Query2Vec is an algorithm that maps SQL queries to a high dimensional vector space. Thus, queries can be embedded into a high-dimensional vector space in a manner that preserves its semantics, such that cosine distance between the vectors performs as well as application-specific distance functions over applications-specific feature vectors. Next, to summarize a workload, we embed all queries in the workload using Query2Vec and then use K-means to find K query clusters and then pick the closest query to the centroid in each cluster as a representative query for that cluster in the summary. To determine the optimal K we use the Elbow method [5] which runs the K-means algorithm in a loop with increasing K until the rate of change of the sum of squared distances from centroids 'elbows' or plateaus out. This method ensures that the subsample thus created contains at least one query from all query clusters, where each query cluster represents a specific type of customer query.

Selected workloads are persisted in Snowflake along with some associated contexts, such as the compilation environments and the start time of the original query which will be useful for rerunning the queries. For previously selected workloads, Snowtrail also enables refreshing the workload to pick up the latest set of customer queries that satisfy the selection criteria.

Snowtrail Workloads can also be imported directly from query texts or other structured formats like CSV or JSON. This is typically used when the workload is fixed and the developer has an exact knowledge of which queries to run. Therefore, Snowtrail workloads could consist of not only production queries, but any query the developer wishes to test on a production environment.

2.3 Snowtrail Runs

Snowtrail employs several mechanisms to enable running customer queries in a secure, reliable and configurable way. These mechanisms include but are not limited to:

- **Result Obfuscation**. Data security is of the utmost importance for Snowflake, so we take great measures to make sure that rerunning customer queries does not reveal customer data. During Snowtrail's compilation phase, hash aggregation operators are added to the top of the query plan. This reduces the results of obfuscated queries to one single hash value, and guarantees that developers using Snowtrail will not be able to see the results returned by the customer query. Snowtrail uses a proprietary hash function to compute stable hash aggregations, which ensures the results are comparable across different runs of the same query.
- **Query Redirection**. It is a common requirement, especially for release testing, that Snowtrail needs to compare two versions of cloud services and virtual warehouses. Therefore, a query redirection mechanism was implemented to redirect queries to run on a specified cloud services instances or clusters with the targeted test version.
- **Configurable query compilation environment**. For workloads imported as query text, a compilation environment (e.g. account, database and schema) needs to be specified to successfully run the workload. Other environment settings such as feature flags can also be configured.
- **Time Travel**. Customers may be running DMLs concurrently with a Snowtrail run, therefore the data for the underlying tables could change between runs. Since Snowflake provides time travel functionality, Snowtrail runs can be configured with a time-travel timestamp where all queries in the workload will be run as of the specified timestamp.

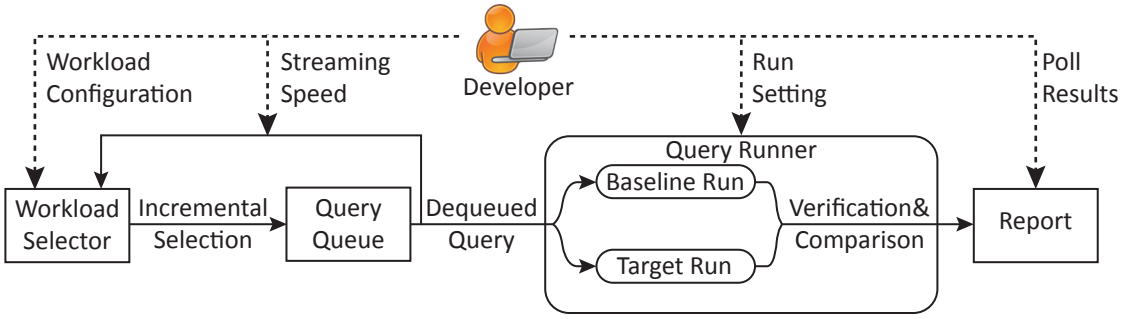

Figure 2: Workflow for Snowtrail Streaming Runs

Query results of two runs are only comparable when they have the same time-travel version.

Snowtrail runs aim to test customer queries without impacting the customer's production workload. Therefore, runs use a separate set of cloud services instances and dedicated virtual warehouses, which can be configured by the developer.

Depending on the resources available (e.g. the size of the virtual warehouse used), developers can specify the concurrency to use to run the workload. High concurrency runs can also be used as a good stress testing mechanism for the virtual warehouse. Snowtrail also has a timeout limit option that puts a cap on the maximum time a query can run before it is canceled to avoid rogue queries from taking up too many resources.

Since Snowtrail workloads maintain the original start times of captured queries, it also provides a "replay" mode to run workloads. The replay mode runs queries with the same time intervals apart as their original start times. This is particularly useful for simulating the exact impact of a captured customer workload on the system. Finally, Snowtrail also provides an "explain" mode, which only compiles queries in the workload for detecting plan changes and compilation errors.

2.4 Verification and Analysis

After a workload generates multiple runs under different settings, Snowtrail performs verification and analysis on the results. We have found that additional verification steps are necessary before performing the analysis, especially for performance comparisons, in order to reduce false positive rates.

When comparing the baseline and target runs, Snowtrail verifies performance regressions in the target run by rerunning the queries reported as slower than the same queries in the baseline run multiple times using the target run settings. This has been very effective in weeding out the false positives in performance comparisons due to variations in the cloud environment or different cache states for either the data in virtual warehouses or the metadata in the cloud services layer. Only after a query consistently shows a significant percentage of performance regression after several verification runs does Snowtrail report it as an actual performance issue. The verification process also looks for queries containing objects with different versions of schema definitions and remove them from the comparison.

Besides performance regressions, Snowtrail also detects wrong results as well as new user errors or internal errors, and automatically generates a report. The report takes extra care to ensure the comparisons are always valid. For example, non-deterministic queries are excluded from performance and result comparisons; queries that reached the time-out limit set by the runs are not compared for performance differences; and queries that fail with the same error code in both runs are not reported as regressions.

Overall, the Snowtrail report tries to be concise and easy to digest. Since all metadata of the workloads and runs are stored in Snowflake, including detailed statistics for each query, sophisticated developers can easily write their own analytical queries to perform more detailed analysis suited to their own testing requirements. We have found it is much more flexible and useful to expose the data in relational schemas through a SQL interface as opposed to baking more functionality into the report itself.

2.5 Streaming Mode

Snowtrail's default model is batch-oriented, where a workload is selected up front, then run under different settings for comparison. This works well enough for small to medium sized workloads, but for larger workloads, this poses the following problems:

- Larger workloads usually take longer to run. Over time, many of the original queries will no longer be valid due to schema changes, e.g. dropped tables.
- Even in the absence of schema changes, many customers are constantly ingesting new data into existing tables. To enable results comparisons, Snowtrail uses a fixed time travel version for each run, which is typically set at the beginning of the run. For long runs, we may end up beyond the time travel retention time.
- Since the workloads are fixed up front, we are missing opportunities to select the latest set of queries that satisfy the search criteria.

To address these problems, Snowtrail provides a streaming mode for running queries. The goal of the streaming mode is to rerun queries relatively soon after they were originally run by the customer, and hence improve the query success rate and more closely match the customer's use case. Figure 2 shows the workflow for running Snowtrail under streaming mode. The developer needs to provide a workload configuration and run settings similar to the

ones provided in the batch mode. However, instead of selecting all queries in the workload up front, the developer now controls the streaming speed which specifies how many queries to select for each unit time period.

In streaming mode, workloads are selected incrementally in micro-batches and added to a query queue which then submits them to the query runner. The query runner takes in each query and multiplexes it to run with different settings before performing verification and comparison. The results of the streaming runs are periodically flushed to the Snowflake database so that developers can poll the latest results ongoing streaming runs.

Since the queries are submitted to the query runner individually, they can be run using different time travel versions. In streaming mode, each query simply uses the current timestamp they are submitted to the runner as the time travel version, and there is no need to travel far back in time. Additionally, since workloads are selected incrementally, they are guaranteed to only contain very recent queries. This means there are fewer data changes or schema changes when running these queries, resulting in fewer invalid queries. With the streaming mode, Snowtrail is able to support much longer runs with higher success rates.

3 SNOWTRAIL USAGE

3.1 Release Testing

Snowtrail has been incorporated as part of Snowflake's weekly release process. We use Snowtrail's streaming mode for release testing to pick up the latest customer queries and run them on both the current version and the new release version to detect potential regressions. The Snowtrail run is used to ensure the stability of the release and is run as the final mandatory sanity check mechanism before migrating customer workloads to the new release. The workflow has been automated for the release pipeline, and reports are automatically generated.

It is desirable to minimize the time window between when new release artifacts are available and when we start migrating customers to a new release. In order to achieve this, the release run uses high concurrency and runs queries on a dedicated multi-cluster warehouse, [2], which can automatically scale up and down to handle query workloads. This allows us to reduce the runtime to a few hours, while running a sufficiently large set of queries.

Our workload selection technique ensures that the sample of chosen queries covers a diverse set of use cases. We demonstrate this in Figure 3 where each point in the plot represents a query embedded into a two dimensional space. The sample selected using Query2Vec based summarization is more spread out and diverse (it guarantees picking one query sample from each query cluster), and thus ensures coverage of all types of queries before release.

Figure 4 shows the improvements of using stream runs vs batch runs. The data is collected for 10 stream runs and 10 batch runs with 4000 queries in each run. The batch runs are performed immediately after the workloads are selected. The first column shows skipped queries during runs, which are mostly caused by schema changes such as dropped objects in the original query. The second column shows queries with changed data between the time they were selected and rerun. The last column shows false positives detected by verification. As shown in the figure, stream runs achieve much higher success rates and much lower false positive rates, and are thus much more efficient.

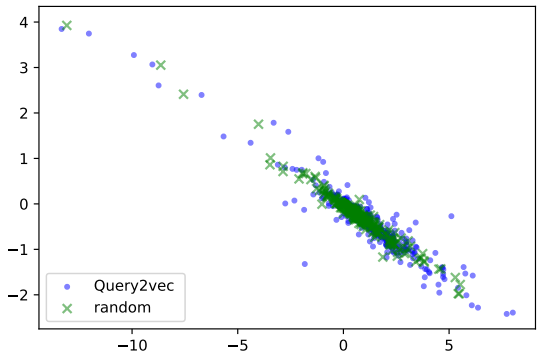

Figure 3: Comparison of Query2Vec based workload sampling and random sampling. The plot shows queries embedded into a 2D space (using T-SNE [6]). The distance between queries represents syntactic differences. The sample of queries selected using Query2Vec summarization picks queries that cover many different types of queries and are more spread out, whereas the random sample tends to pick query types that are more populous (and thus less diverse).

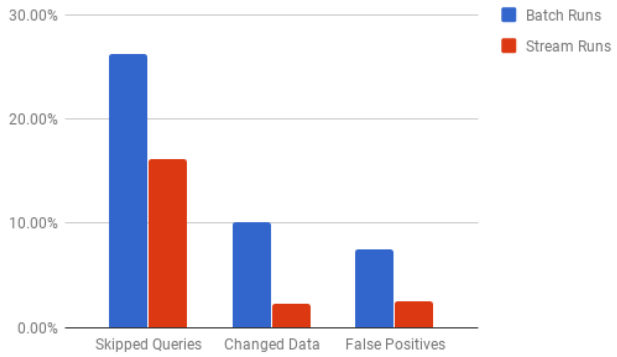

Figure 4: Improvements of stream runs over batch runs. Vertical axis shows the percentage among all selected queries.

3.2 Feature Testing

Before a major SQL related improvement is released, it is tested on production queries using Snowtrail to gain insights on the impacts of the change to customers. To test new features and improvements, developers set up feature flags and follow the workflow as described in Section 2.1.

Besides query history, Snowflake also maintains logs of production queries and makes them available to query using SQL. This means we can easily know exactly which queries will potentially benefit or regress due to a particular change. During feature development, a developer can add log lines where the target feature is

expected to have an impact. After this logging code is in production for some time, the developer can simply query the production logs to determine candidate queries for testing the impact of the feature, and construct workloads out of these queries. We have also implemented a feature usage tracking mechanism that presents this information in a more structured form to simplify this process. Our frequent release schedule enables us to take advantage of the feature usage and logging information to integrate Snowtrail testing as part of the development process and make quick iterations.

Another typical workflow is for testing the impact of plan changes on performance. The developer would issue two runs to first determine queries with changed plans under the feature. This can also help us gauge how widespread the impact would be. Afterwards, the developer would pick only the queries with changed plans and execute them in order to investigate the performance impact. Today these processes are still largely done manually by the developer, but we plan on integrating these patterns into Snowtrail itself.

Our frequent release schedule and Snowtrail's ability to test in production also allows us to quickly assess the potential impact of a feature before we commit to fully implementing and supporting it. The combination of Snowtrail with a service-oriented model and short release cycles has really changed how we prioritize projects and evaluate what is worth doing.

3.3 Workload Player

Besides testing purposes, Snowtrail is also used as a workload player. During online upgrades, and before migrating some customers over to the new version, Snowtrail is used to run workloads that are representative of the access patterns of customer queries to "warm up" various caches for target cloud services clusters. The goal of this exercise is to avoid performance peaks caused by hitting "cold" caches after the upgrade, especially for customers with strict SLAs on query response time.

Besides the release process, Snowtrail has also been used to capture and replay customer workloads in Proof of Concept (POC) projects, and it has been used to perform high concurrency runs for stress testing virtual warehouses.

4 LESSONS LEARNED AND FUTURE WORK

Running Snowtrail in production has provided us with some valuable lessons and feedback. There are some limitations to the approach of rerunning customer production queries: schema changes might render the query invalid; non-deterministic queries cannot be used for result verification; furthermore, queries that consume a lot of resources are expensive to rerun. These limitations could lead to missed testing opportunities, especially for non-deterministic queries which count toward a fairly large percentage of total queries.

We have also learned that it is not an easy task to find a representative workload that can achieve good coverage with a relatively small subset of queries. Query2Vec based summarization provides significantly better coverage than random sampling, however it requires a pre-trained model over a large set of queries. Despite our best efforts, Snowtrail is not always able to detect regressions before customers are migrated to the new release. Therefore, additional safety mechanisms, such as automatically rerunning failed queries on the old version, are still required.

On the other hand, Snowtrail has been more successful in detecting wide-spread issues that would impact lots of queries in the system. Also, even if a particular query was not covered in the pre-upgrade test runs, the query obfuscation and redirection mechanisms implemented for Snowtrail allows us to easily rerun the query on different release versions to identify regressions. False positives were another major issue for early versions of Snowtrail, especially for performance regressions. We have taken various steps to dramatically lower the false positive rate.

We believe there are still many opportunities for further improvements for Snowtrail. For example, we continue to work on improving automatic workload selection to achieve better coverage, and plan to extend Snowtrail to cover more types of queries.

5 CONCLUSION

Snowflake's service-oriented, multi-tenant, elastic architecture offers great opportunities for testing. In this paper, we presented Snowtrail, an internal tool for testing Snowflake with production customer queries. Snowtrail provides functionality to perform workload selection, workload running, and results verification and analysis. Snowtrail has been adopted as an integral part of Snowflake's release pipeline to prevent regressions during online upgrade. It has also been used to test the impact of new features on customer workloads. Snowtrail is also used as a workload player for replaying workloads on the production environment. We believe that infrastructures like Snowtrail are essential to the development and testing pipeline of a large production cloud data warehouse.

ACKNOWLEDGMENTS

We would like to thank Max Chebotarev, Thierry Cruanes, Benoit Dageville, Max Heimel, Neda Nikoo, Shige Takeda, Peter Povinec, William Waddington, Rodney Weaver and Marcin Zukowski for their contribution to this project. We would also like to thank all members of the Snowflake Engineering team for their support.

REFERENCES

[1] Benoit Dageville, Thierry Cruanes, Marcin Zukowski, Vadim Antonov, Artin Avanes, Jon Bock, Jonathan Claybaugh, Daniel Engovatov, Martin Hentschel, Jiansheng Huang, Allison W. Lee, Ashish Motivala, Abdul Q. Munir, Steven Pelley, Peter Povinec, Greg Rahn, Spyridon Triantafyllis, and Philipp Unterbrunner. 2016. The Snowflake Elastic Data Warehouse. In *Proceedings of the 2016 International Conference on Management of Data (SIGMOD '16)*. ACM, New York, NY, USA, 215–226. https://doi.org/10.1145/2882903.2903741

[2] Snowflake Documentation. 2016. Multi-cluster Warehouses. (2016). https://docs.snowflake.net/manuals/user-guide/warehouses-multicluster.html

[3] Leonidas Galanis, Supiti Buranawatanachoke, Romain Colle, Benoît Dageville, Karl Dias, Jonathan Klein, Stratos Papadomanolakis, Leng Leng Tan, Venkateshwaran Venkataramani, Yujun Wang, et al. 2008. Oracle database replay. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. ACM, 1159–1170.

[4] S. Jain, B. Howe, J. Yan, and T. Cruanes. 2018. Query2Vec: An Evaluation of NLP Techniques for Generalized Workload Analytics. *ArXiv e-prints* (Jan. 2018). arXiv:cs.DB/1801.05613

[5] Trupti M Kodinariya and Prashant R Makwana. 2013. Review on determining number of Cluster in K-Means Clustering. *International Journal* 1, 6 (2013), 90–95.

[6] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[7] Ming-Chuan Wu, Jingren Zhou, Nicolas Bruno, Yu Zhang, and Jon Fowler. 2012. Scope Playback: Self-validation in the Cloud. In *Proceedings of the Fifth International Workshop on Testing Database Systems (DBTest '12)*. ACM, New York, NY, USA, Article 3, 6 pages. https://doi.org/10.1145/2304510.2304514

[8] Khaled Yagoub, Peter Belknap, Benoit Dageville, Karl Dias, Shantanu Joshi, and Hailing Yu. 2008. Oracle's SQL Performance Analyzer. (2008).