Simple Replay Tool

Introduction

Customers are always trying to reproduce issues or workloads from clusters or to do what-if scenarios. A customer can easily clone a production cluster, but replicating the workload is more complicated. This set of tools was created to bridge that gap.

- **ScriptRunner.sh**: Uses list of files [**sqls]** and executes all scripts at a pre-determined parallelism
- **SimpleReplay.sh** : Uses a chronologically ordered list of files (sqls) and replay them at the same cadence as originally. The tool also also “fast forward” the workload by pushing of a higher concurrency than original
- **ParseUserActivityLog.py**: Converts the User Activity Log (audit) into a series of sql scripts named <DB>-<USERNAME>_<PID>.sql
- **order_list.sh** : It will order the list of scripts by the time their original session started.
- **mitigate_duplicates.sh** : SED script to allow for duplicated CTAS on the same transaction.

Usage

PREPARATION

1. Enable Amazon Redshift Audit Log. Specficially the User Activity Log https://docs.aws.amazon.com/redshift/latest/mgmt/db-auditing.html
2. Collect the User Activit Logs (AUDIT) files from S3 for the desired time range. You can aggregate them into a single file or run one at a time in chronological order on the same folder.
3. Run

```
ParseUserActivityLog.py [-h] [-c CREDS] [-r] [-s] auditfile
```

A number of .sql files will be generated **<db>-<user>_<pid>.sql**.

- The optional parameter -c defines credentials to be added to the scripts if you want to replay COPY and UNLOAD statements. Note that the script will try to read/write the original bucket used on the workload
- The optional -r parameter tries to remove duplicate statements due to rewrites (it might remove user issued statements, so use with caution).
- The optional parameter -s will prefix any file that contain a write operation with rw-, so you can easily separate that workload.
- Remove any files that you do not want to replay and make a new file listing all the files you want to replay. Normally I will move the files to a sub-folder
- If you want to replay on the same cadence as the original workload you should use **order_list.sh** script to order the **sqls** file.

REPLAY AT THE SAME CADENCE AND CONCURRENCY AS THE ORIGINAL WORKLOAD

```
./SimpleReplay.sh [-b <begin time> ] [-e <end time> | -r <seconds to run> ] \
   [ <script list> ] [ <target concurrency> ]
```

<begin time> optional, Skip the scritps that originally started before this time
<end time> optional, stop replay when the session start time reaches this time

<seconds to run> optional, limits how long the replay runs
<sqls> optional, is the file with the list of scripts to be executed.**Important**: Scritps in this file need to be properly ordered. Use order_list.sh for that. Uses a file names sqls if not declared
<target concurrency> optional, is the target concurrency in case you want to speed up issuing of statements. defaults to 0 meaning original cadence is respected
For long running replays, it is recomended to use

```
nohup ./SimpleReplay.sh [-b <begin time> ] [-e <end time> | -r <seconds to run> ] \
   [ <script list> ] [ <target concurrency> ] &

tail -f nohup.out
```

REPLAY AT ARBITRARY CONCURRENCY

```
./ScriptRunner.sh <n> <sqls>
```

<n> is the number of sessions created and the scrpts referenced on sqls file will be distributed amonst the sessions.
<sqls> is the file that has the list of scripts to run
For long running replays, it is recomended to use

```
nohup ./ScriptRunner.sh <n> <sqls> &
tail -f nohup.out
```

Known Limitations

1. When the original workload was executed with Redshift's JDBC driver the statements might show twice in the log.
2. The replay script does not consider idle time within the sessions, only delay of starting each session.
3. You should customize the **ParseUserActivityLog.py** file to keep/avoid any specific statement (avoid writes or COPY, etc.). Look at Remove_Problem_Keywords() function.
4. Both “runners” use psql as a database CLI and assume you have the environment variables set to connect to the database. You can use your favorite DB CLI for that by replacing psql call in the code

```
export PGPASSWORD=...
export PGUSER=...
export PGHOST=...
export PGPORT=...
```