Fast Salesforce Record Deletion
or
How to Clear a Sandbox in Minutes

Gregory Smith
Capstorm, LLC

Rather than schedule a sandbox refresh and wait, it is possible to clear a Salesforce sandbox in just a few minutes by exploiting the parallelism capabilities of Salesforce's SOAP or REST API. How fast can records be deleted from Salesforce using this approach? You can easily delete 10 million records a hour and with a little more work push this to 24 million.

Table of Contents

The Approach

The major objection to performing large data operations using Salesforce's SOAP API is that only 200 records can be sent to Salesforce in a single API call. However, there is one huge advantage that SOAP has over the BULK API:

- SOAP executes on the Salesforce infrastructure at a much higher priority than BULK.

We have discovered that SOAP's one advantage can be exploited to make it as fast (or much faster) than bulk deletes with the added flexibility of not having to wait for a sandbox refresh.

- By issuing many SOAP delete calls in parallel the amount of wall time to delete a million records is often much faster than doing the same operation using the BULK API.

The One Problem

There is one problem with "Refreshing a Sandbox by Deleting Records" that does not occur when asking Salesforce to refresh a sandbox. The problem is that preparation work is often needed to simplify record deletion. Example:

- If an Account has a parent Account, the child Account needs to be deleted first.

We approached the "preparation work" problem by using the Python/SQLForce module to issue

UPDATE statements to prepare Account records for deletion. For example, we ran the following UPDATE statement against our 1,000,000 test Salesforce accounts to prepare them for deletion.

- UPDATE Account SET parentId=null WHERE parentId<>NULL

Note 1: The SQL like UPDATE statement is a feature of the Python/SQLForce module.

Note 2: Patching 260,075 Accounts took around 90 seconds and also used parallel SOAP requests.

The Advantage of the "One Problem"

The "One Problem" comes with the huge advantage of being able to refresh just the part of a sandbox important to a project rather than the entire sandbox.

Experiments

Now that the "Approach" and the "One Problem" are defined the results of three experiments are presented.

- Experiment 1: Delete 100,000 records using a single Salesforce session.
- Experiment 2: Delete 1,000,0000 records using a single Salesforce session.
- Experiment 3: Delete 1,000,000 records using multiple Salesforce sessions.

Following the experiments are a set of conclusions, recommendations, and appendices containing the raw performance data.

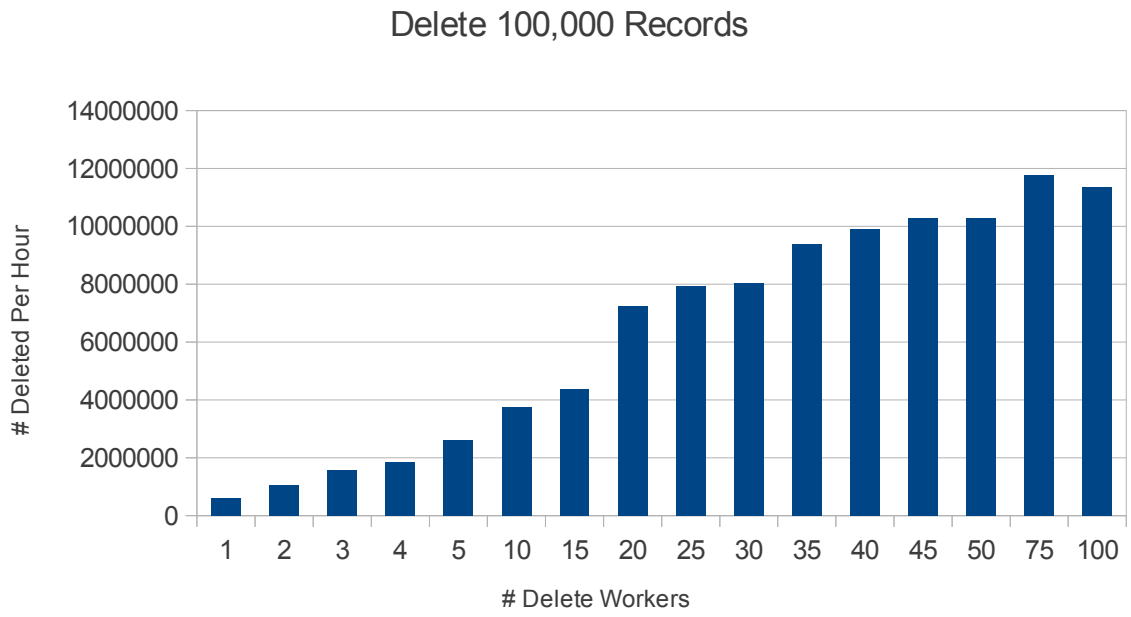
Experiment 1: Delete 100,000 Account Records

In this experiment we deleted 100,000 Account records using a single Salesforce Session and a varying number of "Delete Workers."

A "Delete Worker" is thread of execution that has been assigned a list of records to delete. Multiple "Delete Workers" ran in parallel and and each worker sent batches of 200 records to Salesforce as quickly as it could.

Example: If an experiment used 5 Delete Workers then 1000 (5*200) record deletion requests were active at Salesforce at the same time. With 100 "Delete Workers" 20,000 (100*200) deletion requests were active.

The following table show that delete performance peaked at 10 million per hour with about 40 "Delete Workers" though there was a marginal benefit increasing the number of "Delete Workers" to 100.



Note that the experiments were run on a modest internet connection (20 down, 4 up) with a reported latency of 19ms.

The Python code used in running this test was:

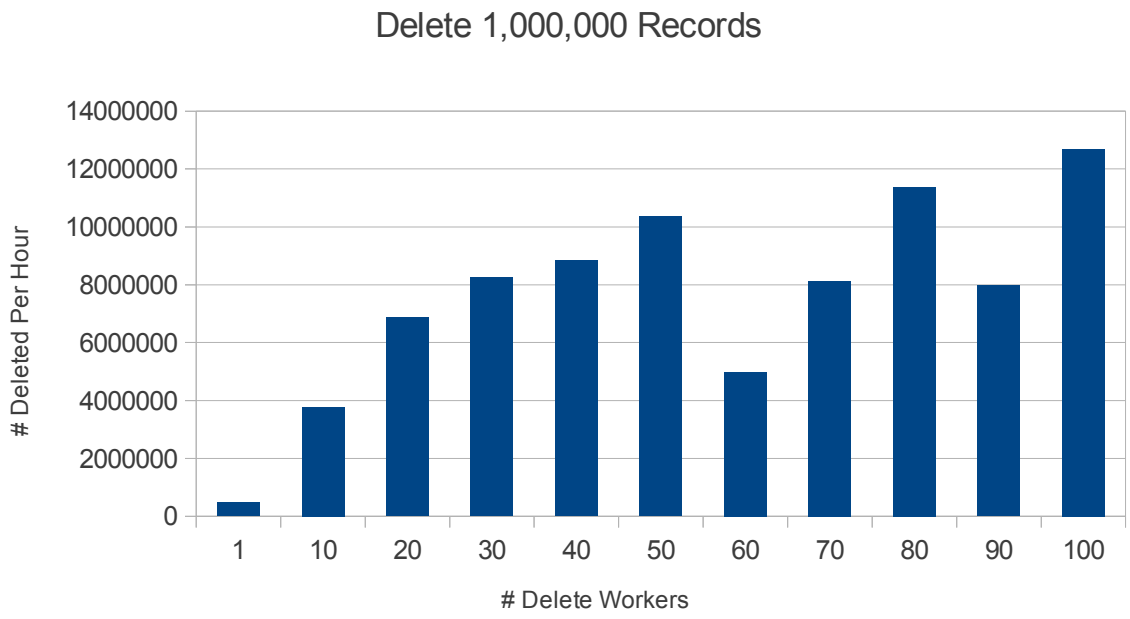
```
import SQLForce
import time

session = SQLForce.Session("Capstorm5gb")

def deleteRecords( session, tableName, soql ):

        nTotal = 0
        while True:

                session.runCommands(soql)
                nDeleted = int(session.getenv('ROW_COUNT'))
                if nDeleted <= 0:
                        return

                nTotal += nDeleted

                print("Deleted batch of " + str(nDeleted) + " records. Total deleted is " + str(nTotal) + " from " +
tableName )


session.setenv("DELETE_THREADS", 90 )
session.setenv( "UPDATE_THREADS", 20 )
session.setenv("QUERY_ALL", False )

startTime = time.time()

print("Delete All Account Parent/Child Relationships")
session.runCommands("UPDATE Account SET parentId=null WHERE parentId<>NULL" )


seconds = (time.time() - startTime )
print("Prep of Accounts took " + str(seconds) + " seconds and updated " + session.getenv("ROW_COUNT") + "
rows")
```

```
print("Delete Accounts")
deleteRecords( session, "Account", "DELETE FROM Account WHERE id<>null  LIMIT 100000" )

endTime = time.time()
seconds = (endTime-startTime)

print("Elapsed Time: " + str(seconds) + " seconds" )

session.logout()
```

Experiment 2: Delete 1,000,000 Account Records

This experiment uses the same Python code as the first experiment and validates that the approach scales to a lot of records. A few facts about the experiment may be interesting:

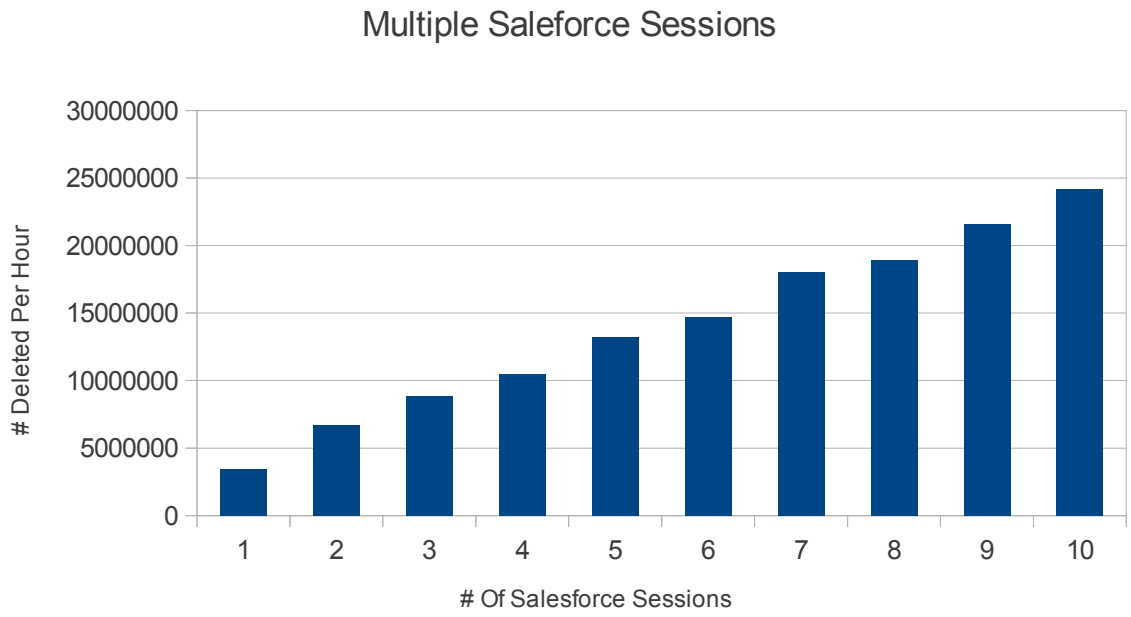
- The 1,000,000 Accounts were repeatedly restored to a 5GB Salesforce sandbox using CopyStorm/Restore. Each restore took about 20 minutes.(e.g. 50,000 records per minute restored).
- Record deletion performance can be measurably impacted by the time of day.



Experiment 3: Delete Records Using Multiple Salesforce Sessions

This experiment used multiple active Salesforce sessions where each session had 10 active "Delete Workers." This means that each Salesforce session sent 2000 record deletion requests to Salesforce at the same time.

Not surprisingly, this experiment showed the greatest record deletion performance – over 24 million per hour.



In the experiment three different Salesforce credentials were used and distributed equally. Example:

- If there were 6 Salesforce sessions the experiment used 2 logins for each credential.

From the results, we suspect that this approach will continue to scale linearly with additional Salesforce credentials but we did attempt to prove our suspicions.

Conclusions

If Salesforce SOAP is used with enough parallelism it is a simple task to "refresh" a Sandbox in a short amount of time and avoid waiting for a regular Sandbox refresh from Salesforce.

In other experiments we have found that similar performance levels are obtainable for INSERT and UPDATE operations by following the same design pattern.

References

SQLForce – a Python Module for Salesforce

This free tool can be found at https://www.capstorm.com/sqlforce-project

The Salesforce Bulk API – Maximizing Parallelism and Throughput Performance When Integrating or Loading Large Data Volumes

This excellent article was our motivation to explore whether similar performance metrics could be achieved via SOAP. See

- https://developer.salesforce.com/page/The_Salesforce_Bulk_API_-_Maximizing_Parallelism_and_Throughput_Performance_When_Integrating_or_Loading_Large_Data_Volumes

Appendix 1: Insert & Updates

It is reasonable to ask whether this same approach can be applied to INSERT/UPDATE operations. In

short, the answer is “yes” with a few restrictions. For a single Salesforce session:

- Saleforce appears to limit the INSERT/UPDATE rate for standard objects to about 50,000 per minute (for a single Salesforce session)
- Salesforce appears to limits the INSERT/UPDATE rate for custom objects to about 10,000 per minute.

These restrictions appear to be avoidable by using multiple Salesforce sessions with a maximum of 10 sessions per Salesforce user name. By increasing the number of user names we have not seen a practical limit to the rate at which records can be inserted/updated into Salesforce.

The best INSERT performance we have achieved is 250,000 Accounts per minute (15 million an hour) using 5 Saleforce sessions each employing 10 writer threads.

Appendix 2: Raw Data

This appendix contains the raw data recorded by the experiments and use to create the graphs in the paper.

Single Session Data

Column Label	Description
Number of Records	Total number of records deleted.
# Delete Workers	Number of parallel threads used to delete records.
# Update Workers	Number of parallel threads used to prepare records for deletion.
Total Time (Seconds)	Total seconds to prepare and delete records. This value is “wall” time and NOT Salesforce CPU time.
Record Prep Time	Total seconds to prepare records for deletion.
Delete Time	Total seconds to delete records.
Deletes Per Second	Average number of records deleted per second.
Deletes Per Minute	Average number of records deleted per minute.
Deletes Per Hour	Average number of records deleted per hour.
Seconds Per 100K	Average number of seconds to delete 100,000 records.
Improvement	Improvement factor over the the single delete worker case. Example: An improvement of 3.5 means 350% faster than the single delete worker case.

Number of Records	# Delete Workers	# Update Workers	Total Time (Seconds)	Record Prep Time	Delete Time	Deletes Per Second	Deletes Per Minute	Deletes Per Hour	Seconds per 100K	Improvement
100,000	1	10	601.17	1.98	599.19	167	10,014	600,811	599.19	1.00
100,000	2	10	348.86	3.52	345.34	290	17,374	1,042,451	345.34	1.74
100,000	3	10	232.98	1.94	231.04	433	25,970	1,558,172	231.04	2.59
100,000	4	10	197.62	1.95	195.67	511	30,664	1,839,832	195.67	3.06
100,000	5	10	139.97	1.86	138.11	724	43,444	2,606,618	138.11	4.34
100,000	10	10	98.67	2.35	96.32	1,038	62,292	3,737,542	96.32	6.22
100,000	15	10	84.70	2.08	82.62	1,210	72,622	4,357,298	82.62	7.25
100,000	20	10	51.50	1.84	49.66	2,014	120,822	7,249,295	49.66	12.07
100,000	25	10	46.52	1.18	45.34	2,205	132,322	7,939,308	45.34	13.21
100,000	30	10	46.54	1.71	44.83	2,231	133,839	8,030,337	44.83	13.37
100,000	35	10	39.37	0.94	38.43	2,602	156,128	9,367,681	38.43	15.59
100,000	40	10	37.23	0.92	36.31	2,754	165,244	9,914,624	36.31	16.50
100,000	45	10	35.99	0.94	35.05	2,853	171,184	10,271,041	35.05	17.10
100,000	50	10	36.08	1.01	35.07	2,851	171,086	10,265,184	35.07	17.09
100,000	75	10	31.58	1.01	30.57	3,271	196,271	11,776,251	30.57	19.60
100,000	100	10	32.62	0.93	31.69	3,156	189,334	11,360,050	31.69	18.91
1,000,000	1	20	7704	104.14	7,599.86	132	7,895	473,693	759.99	1.00
1,000,000	10	10	1,140.40	187.14	953.26	1,049	62,942	3,776,514	95.33	7.97
1,000,000	20	20	618.92	94.10	524.82	1,905	114,324	6,859,455	52.48	14.48
1,000,000	30	30	541.36	104.96	436.40	2,291	137,489	8,249,313	43.64	17.41
1,000,000	40	40	491.01	83.85	407.16	2,456	147,362	8,841,733	40.72	18.67
1,000,000	50	10	488.36	140.73	347.63	2,877	172,597	10,355,838	34.76	21.86
1,000,000	60	20	924.13	202.35	721.78	1,385	83,128	4,987,669	72.18	10.53
1,000,000	70	20	550.51	107.83	442.68	2,259	135,538	8,132,285	44.27	17.17
1,000,000	80	40	383.68	67.52	316.16	3,163	189,777	11,386,640	31.62	24.04
1,000,000	90	20	624.23	172.56	451.67	2,214	132,840	7,970,421	45.17	16.83
1,000,000	100	10	428.41	144.88	283.53	3,527	211,618	12,697,069	28.35	26.80

Multiple Salesforce Sessions

Column Label	Description
Number of Sessions	Total number of Salesforce sessions created.
Number of Users	Number of distinct Salesforce credentials used.
Threads Per Session	Number of parallel threads used delete records in each Salesforce session.
# Deleted	Total number of records deleted.
Total Time (Seconds)	Total number of seconds to delete records.
Deletes Per Second	Average number of records deleted per second.
Deletes Per Second	Average number of records deleted per minute.
Deletes Per Hour	Average number of records deleted per hour.
Seconds Per 100K	Average number of seconds to delete 100,000 records.

Number of Sessions	Number of Users	Threads Per Session	# Deleted	Total Time (Seconds)	Deletes Per Second	Deletes Per Minute	Deletes Per Hour	Seconds per 100K
1	1	10	100,000	104	962	57,692	3,461,538	104.00
2	2	10	200,000	108	1,852	111,111	6,666,667	54.00
3	3	10	300,000	122	2,459	147,541	8,852,459	40.67
4	3	10	400,000	138	2,899	173,913	10,434,783	34.50
5	3	10	500,000	136	3,676	220,588	13,235,294	27.20
6	3	10	600,000	147	4,082	244,898	14,693,878	24.50
7	3	10	700,000	140	5,000	300,000	18,000,000	20.00
8	3	10	800,000	152	5,263	315,789	18,947,368	19.00
9	3	10	900,000	150	6,000	360,000	21,600,000	16.67
10	3	10	1,000,000	149	6,711	402,685	24,161,074	14.90