Google Cloud

Blog Menu

DATABASES

Increase visibility into Cloud Spanner performance with transaction stats

Santhosh Yeduguri

Find an article...

Latest stories

Products

Topics

About

RSS Feed

database. It helps to understand how transactions work in Spanner to best troubleshoot any contentions.

Transaction statistics

Transaction statistics bring you insight into how an application is using the database and are useful when investigating performance issues. For example, you can check whether there are any slow-running transactions that might be causing contention, or you can identify potential sources of high load, such as large volumes of updates to a particular column.

Spanner provides built-in tables that store statistics about transactions. You can retrieve statistics from these `SPANNER_SYS.TXN_STATS*` tables using SQL statements.

Aggregated transaction statistics

Spanner captures aggregated transaction statistics in the following system tables:

- `SPANNER_SYS.TXN_STATS_TOTAL_MINUTE` : Transactions during one-minute intervals
- `SPANNER_SYS.TXN_STATS_TOTAL_10MINUTE` : Transactions during 10-minute

Find an article...

Latest stories

Products

Topics

About

RSS Feed

If Spanner is unable to store statistics for all transactions run during the interval in these tables, the system prioritizes transactions with the highest latency, commit attempts, and bytes written during the specified interval.

Find the root cause of a database contention in Spanner

Transaction statistics can be useful in debugging and identifying transactions that are causing contentions in the database. Next, you'll see how this feature can be used to debug, using an example database where write latencies are high because of database contentions.

Step 1: Identify the time period with high latencies

This can be found in the application that's using Cloud Spanner. For example, the issue started occurring around "`2020-05-17T17:20:00`".

Step 2: See how aggregated transactions metrics changed over a period of time

Query the TXN_STATS_TOTAL_10MINUTE table around the start of the issue. The results of this query may give clues about how latency and other transaction statistics changed over that period of time.

For example, this query can get aggregated transaction statistics, inclusive from "`2020-05-17T16:40:00`" to "`2020-05-17T19:40:00`". This brings back results,

Find an article...

Latest stories

Products

Topics

About

RSS Feed

interval_end	avg_total_latency_seconds	commit_attempt_count	commit_abort_count
2020-05-17 16:40:00-07:00	0.028394498742159258	315691	5170
2020-05-17 16:50:00-07:00	0.025045555854970147	302124	3828
2020-05-17 17:00:00-07:00	0.046048877656441875	346087	11382
2020-05-17 17:10:00-07:00	0.086375063087579362	379964	33826
2020-05-17 17:20:00-07:00	0.12910987654813588	390343	52549
2020-05-17 17:30:00-07:00	0.13144569291013578	456455	76392
2020-05-17 17:40:00-07:00	0.1598060462555369	507774	121458
2020-05-17 17:50:00-07:00	0.16408940591090757	516587	115875
2020-05-17 18:00:00-07:00	0.1578391080373088	552711	122626
2020-05-17 18:10:00-07:00	0.17498201654516557	569460	154205
2020-05-17 18:20:00-07:00	0.17265656768813875	613571	160772
2020-05-17 18:30:00-07:00	0.15877077393258404	601994	143044
2020-05-17 18:40:00-07:00	0.2024720043504819	604211	170019
2020-05-17 18:50:00-07:00	0.16145049947825879	601622	135601
2020-05-17 19:00:00-07:00	0.16532854950745302	596804	129511
2020-05-17 19:10:00-07:00	0.14136634505183712	560023	112247
2020-05-17 19:20:00-07:00	0.13667812808809277	570864	100596
2020-05-17 19:30:00-07:00	0.089365603486055017	539729	65316
2020-05-17 19:40:00-07:00	0.082036892942460721	479151	40398

In the results, you can see that aggregated latency and abort count is higher in the highlighted period of time. We can pick any 10-minute interval (for example, interval ending at "`2020-05-17T18:40:00`") where aggregated latency and/or abort count are high. Then, in the next step, you can see which transactions are contributing to high latency and abort count.

Step 3: Identify the exact transactions that are causing high latency

Query the `TXN_STATS_TOP_10MINUTE` table for the interval you picked in the

Find an article...

Latest stories

Products

Topics

About

RSS Feed

```
11  ORDER BY avg_total_latency_seconds DESC;
```

interval_end	fprint	avg_total_latency_seconds	avg_commit_latency_seconds	commit_attempt_count	commit_abort_count
2020-05-17 18:40:00-07:00	15185072816865185658	0.350793536508159 86	0.013915420509874821	278802	142205
2020-05-17 18:40:00-07:00	15435530087434255496	0.163292904877189 73	0.014157147146761417	129012	27177
2020-05-17 18:40:00-07:00	14175643543447671202	0.142327049157177 48	0.013371019624173641	5357	636
2020-05-17 18:40:00-07:00	898069986622520747	0.019757032833333 33	0.015826366841793 06	6	0
2020-05-17 18:40:00-07:00	10510121182038036893	0.016828849428571428	0.012515991926193237	7	0
2020-05-17 18:40:00-07:00	9287748709638024175	0.015886873628718677	0.011834535747766495	4269	1
2020-05-17 18:40:00-07:00	7129109266372596045	0.014234268175533813	0.010198219679296017	182227	0
2020-05-17 18:40:00-07:00	15630228555662391800	0.011981482551724138	0.010687483474612236	58	0
2020-05-17 18:40:00-07:00	7907238229716746451	0.010764966415384618	0.009681967087090015 4	65	0
2020-05-17 18:40:00-07:00	10158167220149989178	0.009458042113781125	0.004664749372750520 7	3454	0
2020-05-17 18:40:00-07:00	9353100217060788102	0.009327139718620688 9	0.004458911716938018 8	725	0
2020-05-17 18:40:00-07:00	9521689070912159706	0.009251667524390244 6	0.004433728288859129	164	0
2020-05-17 18:40:00-07:00	11079878968512225881	0.006382125953846153 7	0.001829018234275281 4	65	0

The highlighted row in the preceding table is an example of a transaction experiencing high latency because of a high number of commit aborts.

Step 4: Check for similarities among high-latency transactions

We can fetch `read_columns, write_constructive_columns` and `write_delete_tables` columns for transactions with high abort count (also note the `fprint` value, which will be useful in the next step). This is to check whether high-latency transactions are operating on the same set of columns.

Query

```
SELECT
  fprint,
  read_columns,
  write_constructive_columns,
  write_delete_tables
```

Find an article...

Latest stories

Products

Topics

About

RSS Feed

As the output shows, the transactions with the highest average total latency are reading

You can see how the statistics associated with this transaction shape have changed over a period of time. Use the following query, where `$FPRINT` is the fingerprint of the high-latency transaction from the previous step.

Query

```
SELECT
  interval_end
  ROUND(avg_total_latency_seconds, 3) AS latency,
  ROUND(avg_commit_latency_seconds, 3) AS commit_latency,
  commit_attempt_count,
  commit_abort_count,
  commit_failed_precondition_count,
  avg_bytes
FROM SPANNER_SYS.TXN_STATS_TOP_10MINUTE
WHERE
  interval_end >= "2020-05-17T16:40:00"
  AND interval_end <= "2020-05-17T19:40:00"
  AND fprint = $FPRINT
ORDER BY interval_end;
```

Output

```
+---------------------------+---------+----------------+------------+-----------+----------------------+-------+
| interval_end              | latency | commit_latency | commit_att | commit_ab | commit_failed_precon | avg_b |
|                           |         |                | empt_count | ort_count | dition_count         | ytes  |
+---------------------------+---------+----------------+------------+-----------+----------------------+-------+
| 2020-05-17 16:40:00-07:00 | 0.095   | 0.010          |      53230 |      4752 |                    0 | 91    |
| 2020-05-17 16:50:00-07:00 | 0.069   | 0.009          |      61264 |      3589 |                    0 | 91    |
```

Find an article...

Latest stories

Products

Topics

About

RSS Feed

getting aborted more frequently, commit attempts are also high because of commit

step is to look at the commit abort error messages received by the application to know the reason for aborts. By inspecting logs in the application, we see the application actually changed its workload during this time. That likely means that some other transaction shape showed up with high attempts_per_second, and that a different transaction (maybe a nightly cleanup job) was responsible for the additional lock conflicts.

Cloud Spanner transaction statistics provides greater observability and insight into your database behaviors. Use both transaction statistics and query statistics to tune and optimize your workloads on Spanner.

To get started with Spanner, create an instance in the Cloud Console or try it out with a Spanner Qwiklab.

POSTED IN: DATABASES—GOOGLE CLOUD PLATFORM

RELATED ARTICLES

Databases that transform businesses—What happened at Google Cloud

3 reasons to consider Cloud Spanner for your next project

Find an article...

Latest stories

Products

Topics

About

RSS Feed