Top ten slowest database queries

Slow database queries often result in slow response times for the upstream consumer of the information, resulting in user dissatisfaction or SLA violations. You want to identify the worst performing queries so you can investigate them.

Data required

Database data

Procedure

1. Verify that you have deployed Splunk Stream.
2. Run the following search. You can optimize it by specifying an index and adjusting the time range.

   ```splunk
   | eval time_taken_sec = time_taken / 1000000
   | stats count sum(time_taken_sec) AS total_time_taken_sec 
   | avg(time_taken_sec) AS avg_time_taken_sec 
   | perc95(time_taken_sec) AS perc95_time_taken_sec 
   | max(time_taken_sec) AS max_time_taken_sec 
   | BY query
   | sort - perc95_time_taken_sec
   | head 10
   | table query count *_time_taken_sec
   | foreach *_time_taken_sec [eval &lt;&lt;FIELD&gt;&gt;=round(&lt;&lt;FIELD&gt;&gt; , 3)]
   ```

Search explanation

The table provides an explanation of what each part of this search achieves. You can adjust this query based on the specifics of your environment.

<table>
<thead>
<tr>
<th>Splunk Search</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>eval time_taken_sec = time_taken / 1000000</td>
<td>Convert microseconds to seconds for better readability.</td>
</tr>
<tr>
<td>stats count sum(time_taken_sec) AS total_time_taken_sec</td>
<td>Calculate the total time taken, average time taken, max time taken, and the 95th percentile of times taken (all in seconds) for each database query.</td>
</tr>
<tr>
<td>avg(time_taken_sec) AS avg_time_taken_sec</td>
<td></td>
</tr>
<tr>
<td>perc95(time_taken_sec) AS perc95_time_taken_sec</td>
<td></td>
</tr>
<tr>
<td>max(time_taken_sec) AS max_time_taken_sec</td>
<td></td>
</tr>
<tr>
<td>BY query</td>
<td></td>
</tr>
</tbody>
</table>
Splunk Search

|sort - perc95_time_taken_sec

Explanation
Sort the results according to the 95th percentile with the largest value first.

|head 10

Display the top 10 results in a table.

|table query count * _time_taken_sec

Display the results in a table with columns in the order shown.

|foreach * _time_taken_sec [eval &lt;&lt;FIELD&gt;&gt;=round('&lt;&lt;FIELD&gt;&gt;', 3)]
Round the results to three decimal places.

Next steps

The following is a sample result of this search.

<table>
<thead>
<tr>
<th>query</th>
<th>count</th>
<th>avg_time_taken_sec</th>
<th>max_time_taken_sec</th>
<th>perc95_time_taken_sec</th>
<th>total_time_taken_sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELECT * FROM customers</td>
<td>7</td>
<td>1.08</td>
<td>1.66</td>
<td>1.61</td>
<td>7.55</td>
</tr>
<tr>
<td>WHERE customer_id=uid</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>887</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
<td>39.85</td>
</tr>
<tr>
<td>SELECT password FROM SY$</td>
<td>1700</td>
<td>0.04</td>
<td>0.06</td>
<td>0.06</td>
<td>76.35</td>
</tr>
<tr>
<td>WHERE name='SYS'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The information provided in Splunk Lantern is intended for informational and educational purposes only. All information is provided in good faith, however, Splunk disclaims any and all representations and warranties, express and implied, regarding the information provided, including without limitation any warranties and representations regarding the completeness, adequacy or accuracy of the information. You agree to take full responsibility for the results arising from the use of the information provided.
The output shows the longest running SQL statements sorted by the percentile column so you can pinpoint which SQL statements take the most time. It also shows the frequency of execution. Total time may not be the highest for the longest running query because of a lower count value. The worst case is when the longest running query also has the highest count value.

Another helpful observation is to look at total time taken. In the sample results, the highest total time taken from the 1,700 executions of the shown SELECT statement is just over 76 seconds. This may also be a place for optimization on the client side, such as reducing the calls by caching the results on the client. It may also be that the call is misplaced in the client code and is called more frequently than necessary.

Performance profiling of SQL statements is also commonly found in the database itself. Having profiling of the statements also in Splunk lets you group it with other related performance metrics such as cpu, memory, storage, middleware, network, and web interface from all the components that make up the service or application. You can put all these results side-by-side on a single dashboard for more insight than you’d have from one of these measurements alone.

Finally, you might also want to look at similar searches in our article [Analyzing wire data from databases](#).