Performance Analysis

Table 1: Version History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Author | Version | Comments |
| 2019-03-18 | Przemysław Jagodziński | 0.1 | First draft |

Contents

[1 Overview 1](#_Toc28509433)

[1.1 Context 2](#_Toc28509434)

[1.2 Test Types, Test Levels and Testing Tools 2](#_Toc28509435)

[1.3 Environments 3](#_Toc28509436)

[1.4 Test Scenarios 4](#_Toc28509437)

[2 Tests Execution 4](#_Toc28509438)

[2.1 Data Warehouse 4](#_Toc28509439)

[2.2 API / Tableau Server 5](#_Toc28509440)

[2.2.1 API 5](#_Toc28509441)

[2.2.2 Tableau 6](#_Toc28509442)

[2.3 UI 8](#_Toc28509443)

[3 Conclusions 10](#_Toc28509444)

[4 References 11](#_Toc28509445)

# Overview

In order to test performance of a system correctly we need to take into account several levels of an application. Two major levels are API and UI. Since we know that the majority of business logic is in DB we decided to include DB performance testing to get precise information about performance, hence the formula for total time of each request should be as follow

where

for API solution, or

To measure time on UI we will use:

## Context

Figure 1: Topology diagram

## Test Types, Test Levels and Testing Tools

ILS solution will be tested on three levels:

1. DB level – artificial traffic will be generated by running procedures on several instances. We will be using pure sql queries running from PowerShell script instances (Figure 2). During testing we will be increasing test data from returned by stored procedure from 100 records to 1000 records.
2. API
   1. ILS API – artificial traffic will be generated using JMeter. We will be increasing parallel requests from 1-10 as well as number of test data returned by stored procedure from 100 records to 1000 records. We will be measuring response time of a single request (Figure 3).
   2. Tableau – we will be using embedded Tableau recorder to tests response time. We will be increasing test data returned by stored procedure from 100 records to 1000 records (Figure 4).
3. Web UI – We will be measuring HTML/CSS/JS rendering times using Page Speed Chrome plugin (Figure 5)

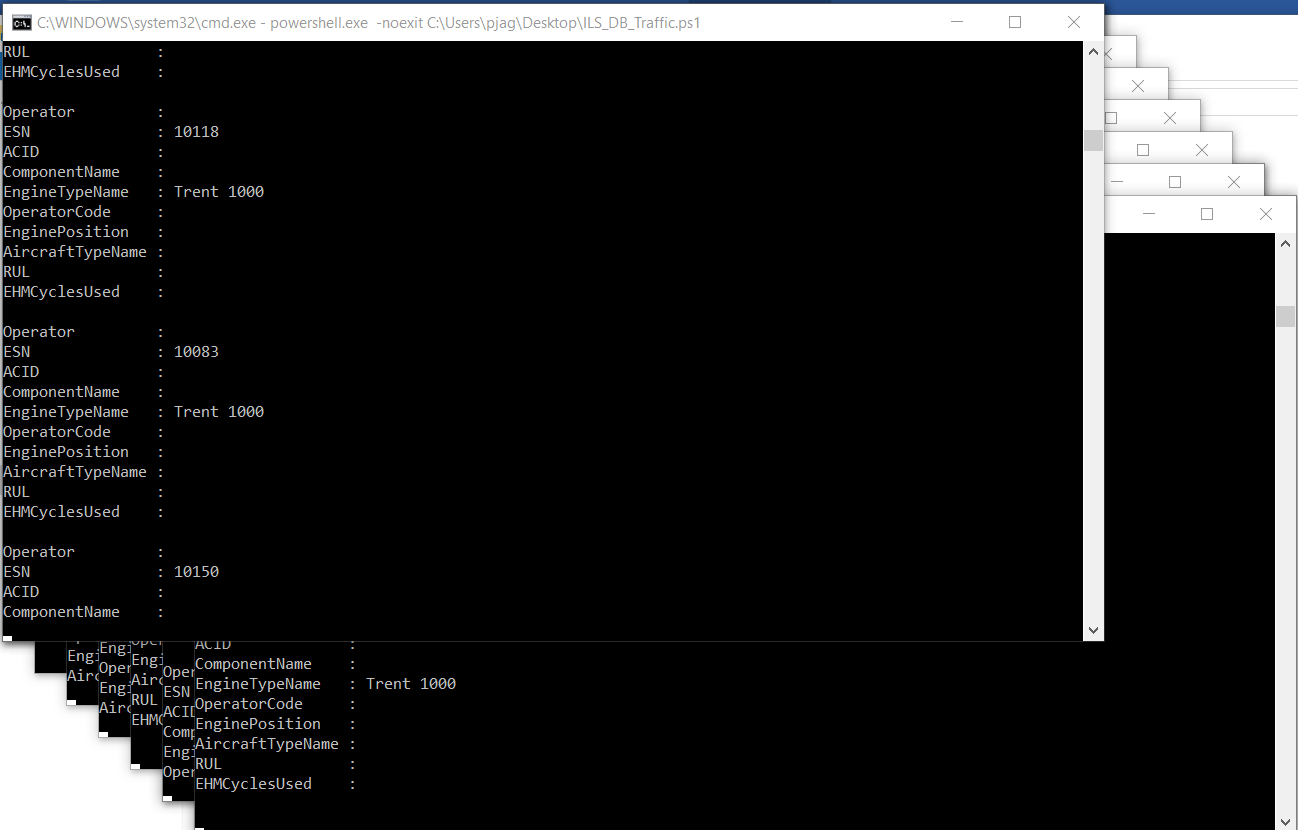


Figure 2: PowerShell script run in 10 CMD instances

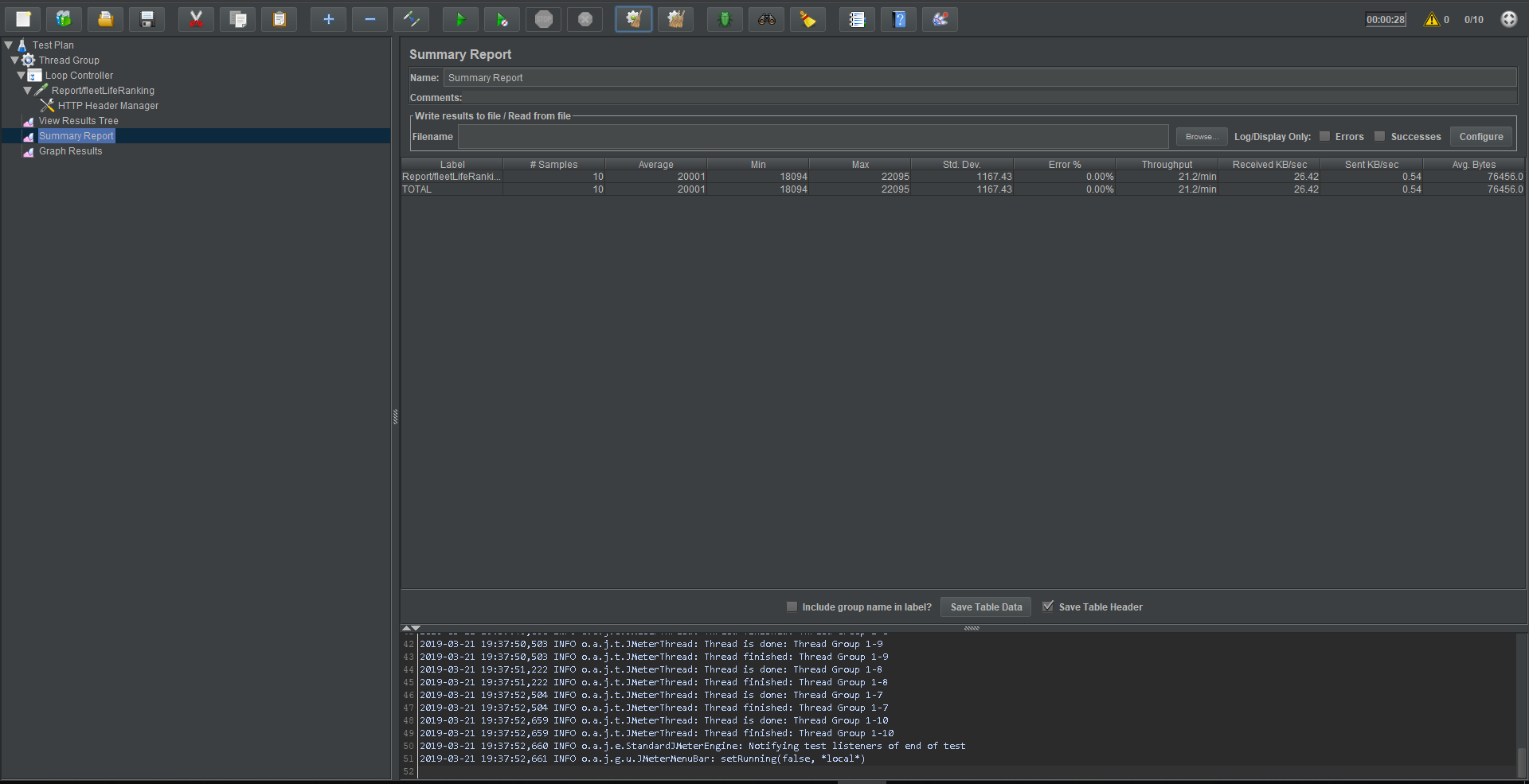


Figure 3: JMeter workbook

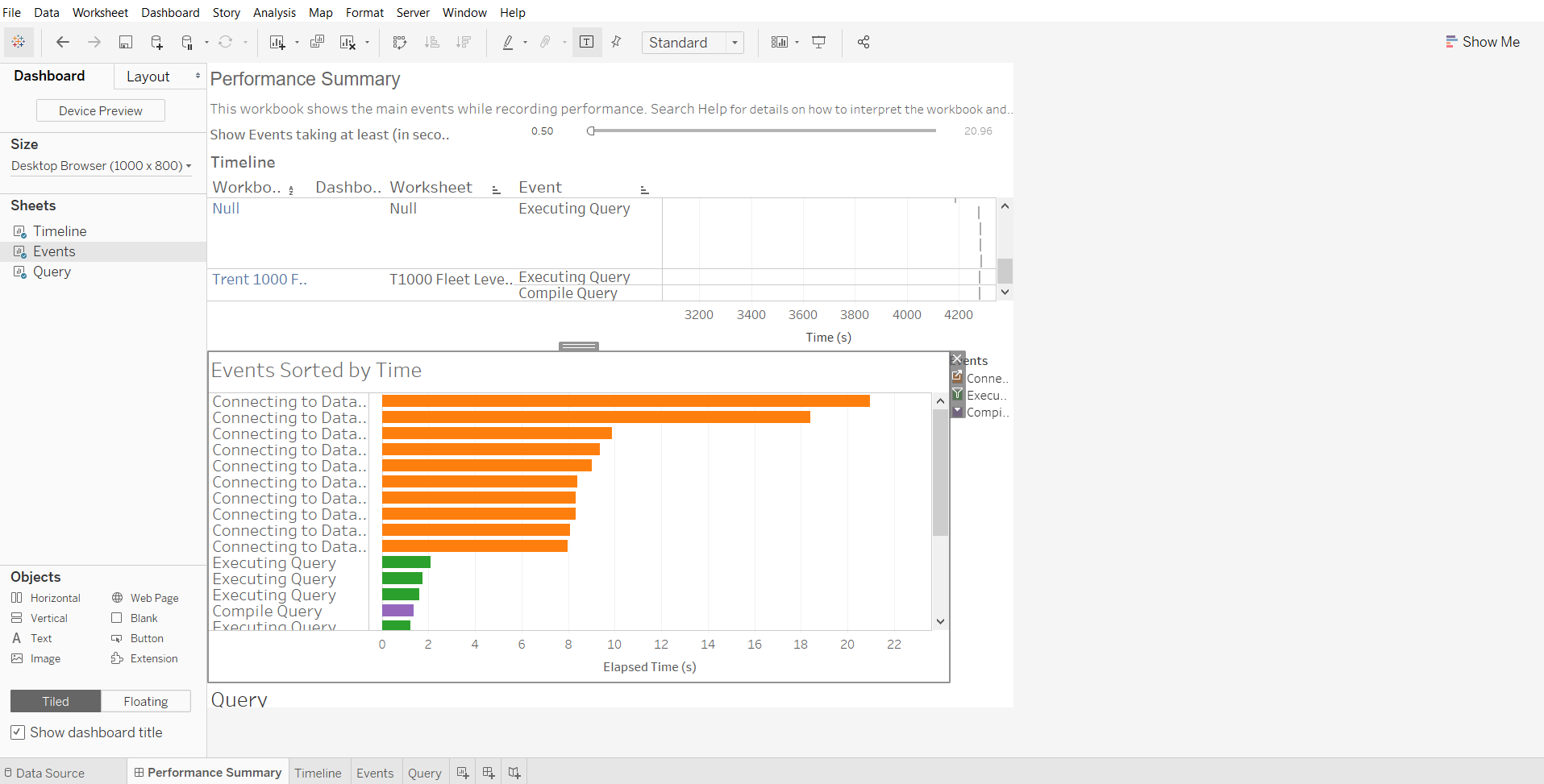


Figure 4: Tableau performance recorder

## Environments

For the purpose of this exercise following environments are set up:

1. Data Warehouse is available and running on Azure
2. Tableau Server running on
3. API and Web are deployed on local machine

## Test Scenarios

Following 3 Scenarios will be verified during performance tests phase:

1. Loading of Table Grid data / Tableau Report
2. Sorting data in Table Grid / Tableau Report
3. Filtering data in Table Grid / Tableau Report

# Tests Execution

## Data Warehouse

We will be running stored procedure on Data Warehouse in three different configurations. No DB Cache clearing. Additionally, by DW instance we mean separate DB connection opened by powershell script.

Test Case 1 – No traffic in Data Warehouse. 600 input records.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time [ms]** | **Comments** |
| 1 | 6000 |  |
| 2 | 7000 |  |
| 3 | 7000 |  |
| 4 | 7000 |  |
| 5 | 7000 |  |
| 6 | 7000 |  |
| 7 | 7000 |  |
| 8 | 7000 |  |
| 9 | 6000 |  |
| 10 | 10000 |  |

Table 1: No traffic in Data Warehouse. 600 input records.

Test Case 2 – Procedures executed in 10 DW instances. 100 input records.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 2: Procedures executed in 10 DW instances. 100 input records.

Test Case 3 – Procedures executed in 10 DW instances. 600 input records.

10 procedure executions in 10 Powershell scripts

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 | 9000 | We started procedure execution in SSMS too fast, before Powershell scripts were able to connect to DW |
| 2 | 12000 |  |
| 3 | 32000 |  |
| 4 | 17000 |  |
| 5 | 21000 |  |
| 6 | 21000 |  |
| 7 | 30000 |  |
| 8 | 13000 |  |
| 9 | 17000 |  |
| 10 | 26000 |  |

Table 3: Procedures executed in 10 DW instances. 600 input records.

## API / Tableau Server

We will be testing both ILS API and Tableau Server responses

### API

In case of API testing we will be using JMeter and generating parallel requests in two different configurations: 100 records returned by procedure and 1000 records returned by procedure.

Test Case 1 – 10 parallel requests. 100 input records.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 1: 10 parallel requests. 100 input records.

Test Case 2 – 10 parallel requests. 600 input records.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time [ms]** | **Comments** |
| 1 | 12292 |  |
| 2 | 12756 |  |
| 3 | 15978 |  |
| 4 | 20683 |  |
| 5 | 18433 |  |
| 6 | 34627 |  |
| 7 | 19354 |  |
| 8 | 19079 |  |
| 9 | 18837 |  |
| 10 | 20001 |  |

Table 2: 10 parallel requests. 600 input records.

### Tableau

We will use embedded in Tableau Desktop Performance recorder plugin. This will allow to measure a time of response + time of rendering elements. We will cover two scenarios for loading report and two scenarios for sorting and filtering report. A prerequisite for each test is cleared out Tableau Cache.

Test Case 1 – Procedures executed in 10 DW instances. 100 input records.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 1: Procedures executed in 10 DW instances. 100 input records.

Test Case 2 – ~~Procedures executed in 10 DW instances~~. 600 input records.

Connecting to Data Source + Executing Query

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time [ms]** | **Comments** |
| 1 | 20960 + 750 | First run after DW was resumed |
| 2 | 18420 + 2090 |  |
| 3 | 9870 + 710 |  |
| 4 | 9350 + 710 |  |
| 5 | 9010 + 1030 |  |
| 6 | 8390 + 520 |  |
| 7 | 8330 + 930 |  |
| 8 | 8330 + 1060 |  |
| 9 | 8080 + 650 |  |
| 10 | 7990 + 660 |  |

Table 2: ~~Procedures executed in 10 DW instances.~~ 600 input records.

Test Case 3 – Sorting by operator

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 3: Sorting by operator.

Test Case 4 – Filtering by operator.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 4: Filtering by operator.

## UI

Finally, we will be testing rendering elements on UI. This will be mostly important in case of filtering and sorting data in table grid as well as Rendering embedded Tableau Report in UI. We will be testing it on Chrome and using Page Speed plugin to Chrome. A prerequisite for each test is cleared Browser Cache.

Test Case 1 – 100 input records on grid.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 1: 100 input records on grid.

Test Case 2 – 100 input records in tableau.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 2: 100 input records in tableau.

Test Case 3 – Sorting on grid by operator

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 3: Sorting on grid by operator.

Test Case 4 – Filtering on grid by operator.

|  |  |  |
| --- | --- | --- |
| **Run Number** | **Execution Time** | **Comments** |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |
| 8 |  |  |
| 9 |  |  |
| 10 |  |  |

Table 4: Filtering on grid by operator.

# Conclusions

We presented the results of performance analysis of system. We were not able to determine which solution (Grid or Tableau) is better for small amount of data. We found that the biggest problem in both solutions is stored procedure execution time. It takes between 10 – 30s for stored procedure to process and return 600 rows (Figure 6). This is also approximately time needed for UI to display data Times of sorting and filtering small amount of data for both solutions are negligible hence we do not present results of performance analysis.

For better comparison of Grid and Tableau solution we will need to deploy ILS Web API into Azure and load bigger amount of test data.

To determine real life systems traffic we will need to answer following questions:

1. How many users will be using system? How often? On a daily-basis, once a week?
2. How many requests (stored procedures triggering) users will be making per defined amount of time. What is the acceptable response time for single request (data loading)?
3. What is the amount of data stored procedures will need to process

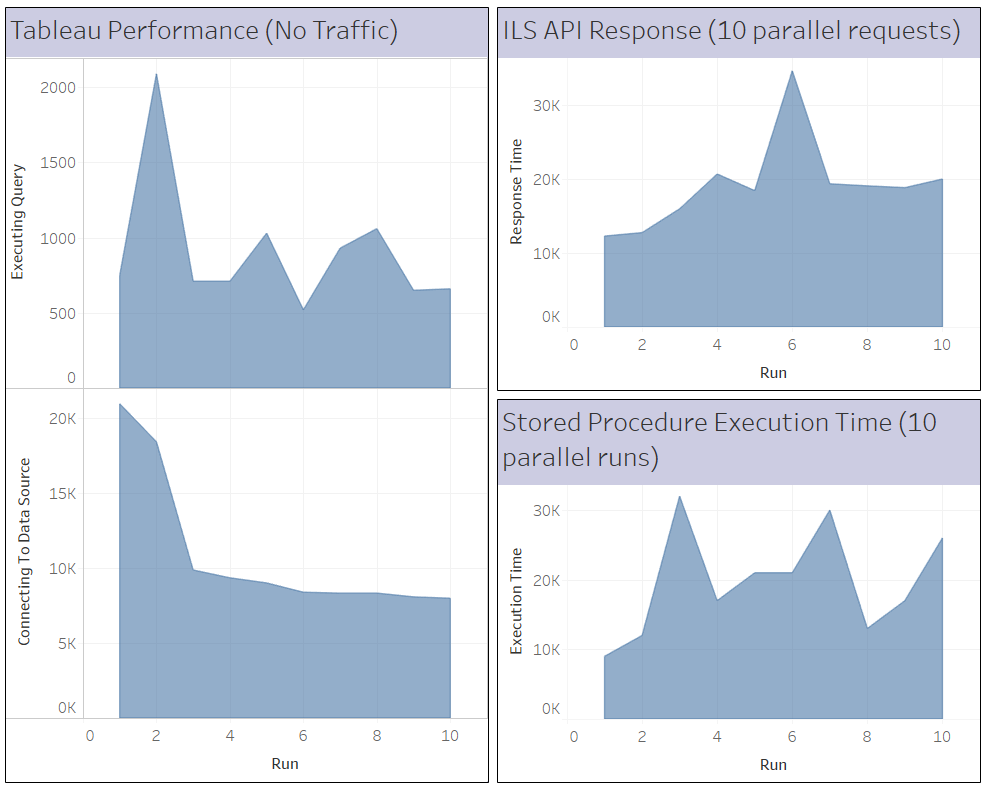


Figure 6: Tableau Performance, API requests responses and DB stored procedure execution times in [ms]

# References

| **ID** | **Reference** | **Date** | **Version** | **Description** |
| --- | --- | --- | --- | --- |
|  |  |  |  |  |