

Using SAS® Viya™ Microservices Logging for Performance Analysis of SAS® Visual Analytics 8.1 Reports

Scott Sweetland, SAS Institute Inc.

ABSTRACT

Your SAS Visual Analytics users begin to create and share reports. As an administrator, you want to track performance of the reports over time, analyzing timing metrics for key tasks such as data query and rendering, relative to total user workload for the system. Logging levels can be set for the SAS Visual Analytics reporting services that provide timing metrics for each report execution. The log files can then be mined to create a data source for a time series plot in SAS Visual Analytics. You see report performance over time with peak workloads and how this impacts the user experience. Isolation on key metrics can identify performance bottlenecks for improvement. First we look at how logging levels are modified for the reporting services and focus on tracking a single user viewing a report. Next, we extract data from a long running log file to create a report performance data source. Using SAS Visual Analytics, we analyze the data with a time series plot, looking at times of peak work load and how the user experience changes.

INTRODUCTION

In the SAS Viya architecture, Cloud Analytics Services (CAS) consists of the microservices and the CAS server (Figure 1). The CAS server is the in-memory engine for data analysis. The microservices provide the Viya web applications with security, access to the CAS server, and other life cycle functions to create, manage and share reports and visualizations. Each microservice is designed to do exactly one function and to do it well, using stateless REST interfaces. CAS provides platform-as-a-service capability for cloud deployment, while application developers are abstracted from low-level details of the cloud environment.

SAS Environment Manager monitors the microservices and the in-memory analytics server. SAS Environment Manager also manages run-time configuration updates for microservices and applications.

SAS Viya Topology for Web Applications

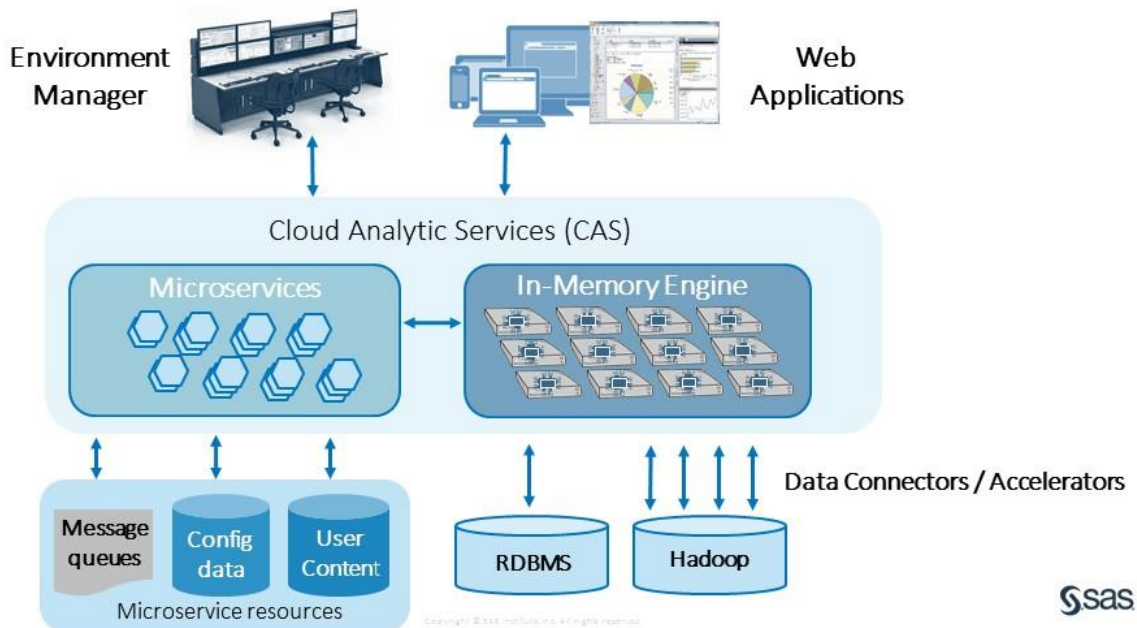


Figure 1. SAS Viya Architecture for Web Applications

Administrators are often asked to investigate user concerns about the performance of their favorite reports. With SAS Environment Manager, it is easy to modify a microservice configuration while the service is running. You can update logging levels to capture performance metrics for user events.

By adding DEBUG level logging to key microservices, you can capture performance metrics for each SAS Visual Analytics report execution. You can then use this data to analyze a single report execution or to look at all report executions as a time series.

VISUAL ANALYTICS VIEWER AND REPORT MICROSERVICES

Once a report has been created and saved in SAS Visual Analytics, it is shared to consumers with the SAS Visual Analytics viewer. The viewer web application relies on the Report Packages microservice for the tasks needed to render and view the report, such as the following:

- Retrieving the saved report definition
- Executing CAS server actions
- Retrieving and caching analytic results
- Applying the report theme
- Rendering the report

The Report Packages microservice interacts with other microservices for these tasks, and the Report Data microservice is key to executing CAS server actions and retrieving results (Figure 2).

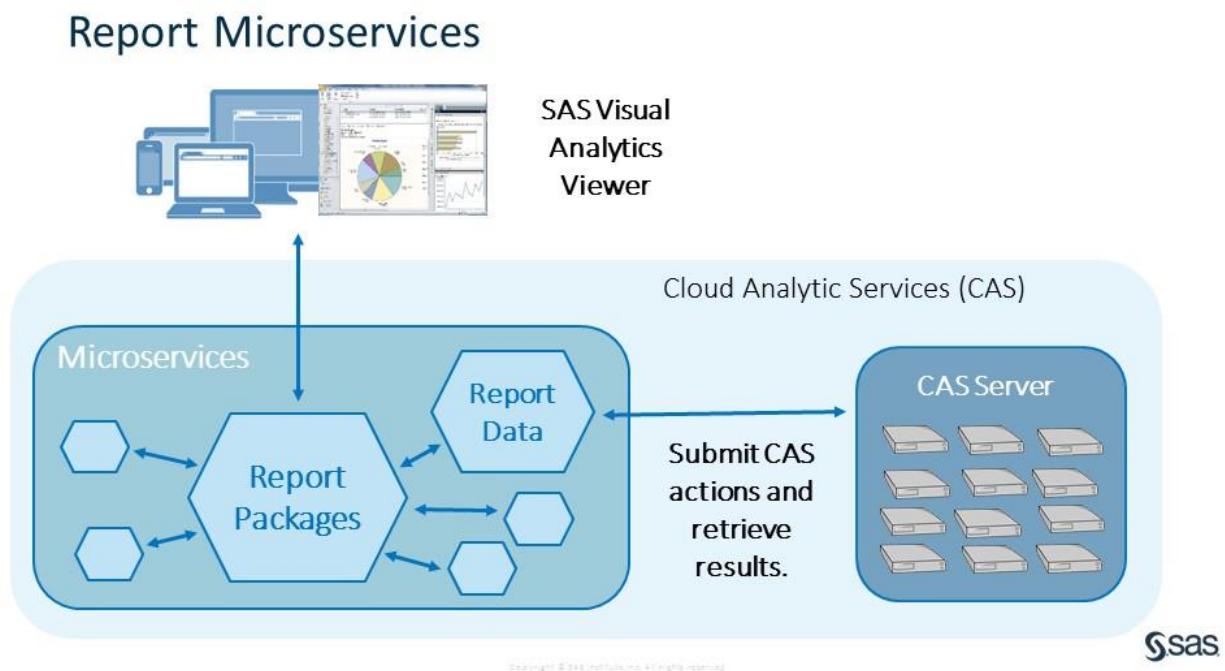


Figure 2. SAS Viya Architecture for Web Applications

ENVIRONMENT MANAGER FOR MICROSERVICE CONFIGURATION

UPDATING MICROSERVICE CONFIGURATION

Microservices are started with an initial configuration. SAS Environment Manager is then used to modify the configuration as needed. Because the microservices were developed using the Java™ Spring Boot framework, they respond to configuration changes without the need to re-start the microservice.

SAS Environment Manager communicates to the Configuration service, which manages configuration for the other microservices. Each microservice listens to the Configuration service for updates and implements the updates without being restarted (Figure 3).

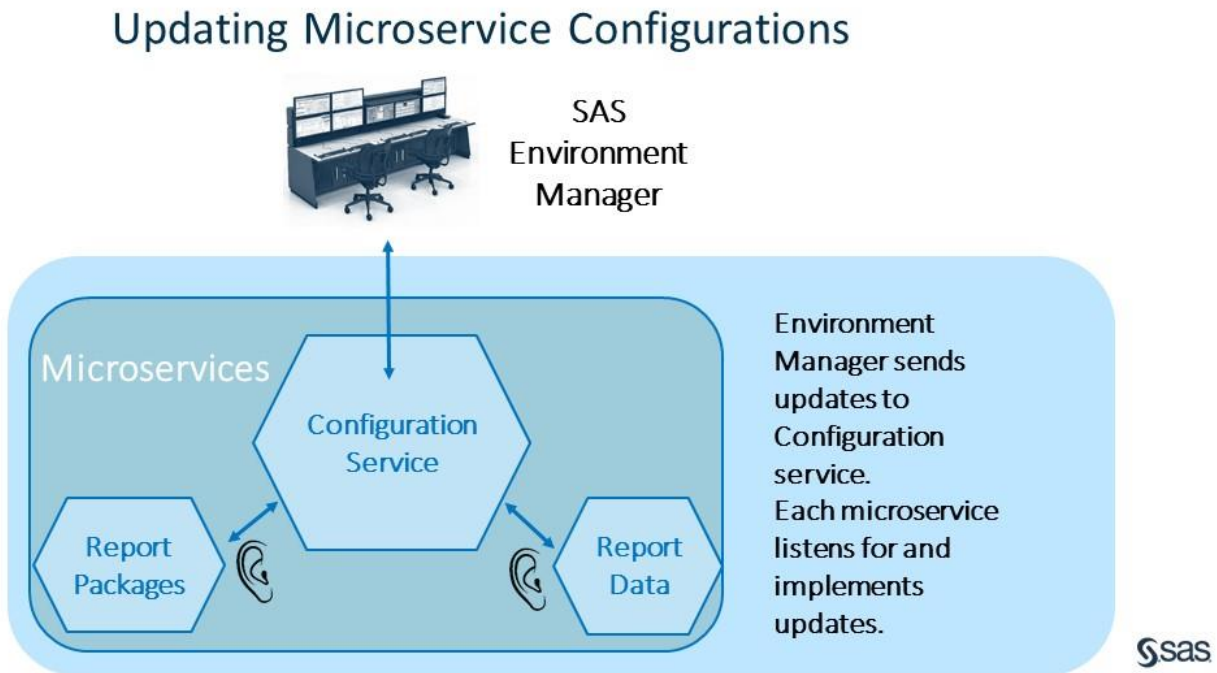
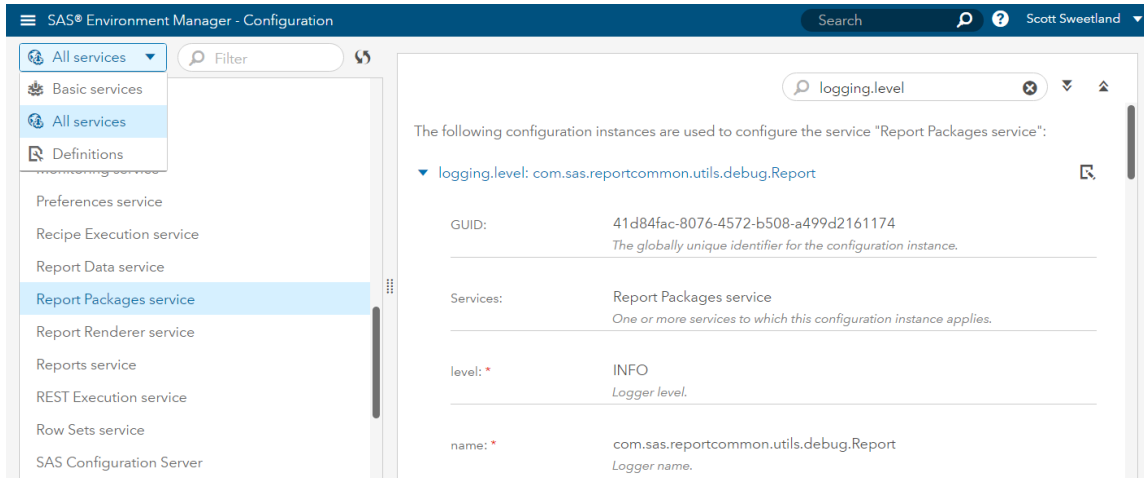


Figure 3. SAS Viya Architecture for Web Applications

SAS ENVIRONMENT MANAGER CONFIGURATION WINDOW


In the SAS Environment Manager Configuration window, changing the configuration for a service is quick and straightforward. You must be a SAS administrator to use the Configuration window. Starting at the Side Menu icon (☰) at the upper left, navigate to the Resources Configuration window. Here are the steps to modify logging.level for the Report Package service (Display 1):

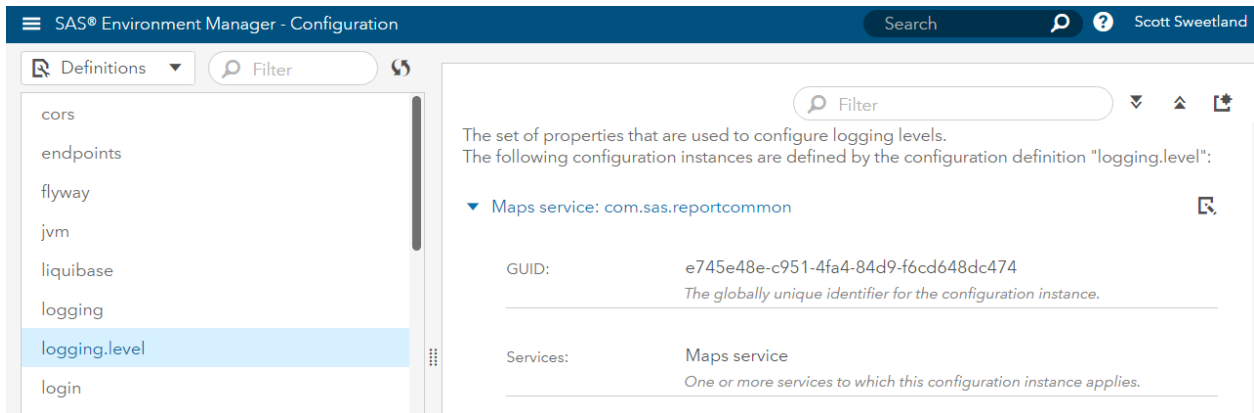
1. In the top left pull down menu, select 'All services'.
2. In the list of services, scroll down and select the Report Packages service.
3. Use the Filter in the configuration view to focus only on logging.level entries.
4. Select the edit icon (✎) to edit the level value for an existing configuration entry.



Display 1. SAS Environment Manager Configuration Window, All Services

Alternatively, you might want to create a new logging.level configuration for a service. Here are the steps to create a new logging.level configuration (Display 2):

1. In the Resources Configuration window, select Definitions in the top left pull down menu.
2. Scroll down the list of definitions and select the 'logging.level' definition.
3. Select the Create New icon () in the top right.
4. Complete the fields and Save (Display 3)



Display 2. SAS Environment Manager Configuration Window, Definitions

New logging.level Configuration

Services: ▼
One or more services to which this configuration instance applies.

level: *
Logger level.

name: *
Logger name.

Display 3. SAS Environment Manager Configuration Window, Definitions

Set logging.level configurations as described in SAS Viya documentation or as directed by SAS Technical Support. As a general rule, Logger levels progress from sparse (fewer log statements) to verbose (more log statements), starting first with INFO, then WARN, then ERROR, and finally DEBUG which generates the most log statements.

MICROSERVICE LOGGING FOR REPORT PERFORMANCE METRICS

UPDATING MICROSERVICE LOGGING LEVELS

Viewing a report in the SAS Visual Analytics viewer uses two key reporting microservices: Report Packages and Report Data. To begin collecting performance metrics for reports, the administrator needs to update three logging levels in two microservices (Table 1).

Table 1. Logging Level Updates For Report Performance Metrics

Service: Report Packages		
Configuration Instance	Name	Level
logging.level: com.sas.reportpackages.service.utils.ReportGenerator	com.sas.reportpackages.service.utils.ReportGenerator	DEBUG
Results: Log statements showing user ID, report ID, and elapsed time for each report view.		
Service: Report Packages		
Configuration Instance	Name	Level
Logging.level: com.sas.reportcommon.utils.debug.Time	com.sas.reportcommon.utils.debug.Time	DEBUG
Result: Log statements showing time in milliseconds for each step in report generation.		
Service: Report Data		
Configuration Instance	Name	Level
Logging.level: com.sas.reportcommon.utils.debug.Time	com.sas.reportcommon.utils.debug.Time	DEBUG
Result: Log statements showing time in milliseconds for each step in report generation.		

For some services, logging.levels have already been defined and preset to the INFO logging level. You can update these existing configurations by selecting the edit icon for the logging level configuration and updating the Logger name value (Display 4). This field refers to the file namespace for which logging is requested. A short value, like “com.sas.reportpackages” requests logging from many files, while a fully qualified value like “com.sas.reportpackages.service.utils.ReportGenerator” requests logging from a single specific file.

If you want to stop logging for a defined DEBUG logging.level configuration, set the Logger level value to INFO.

Edit logging.level Configuration

GUID:
The globally unique identifier for the configuration instance.

Services:
One or more services to which this configuration instance applies.

level: *
Logger level.

name: *
Logger name.

Display 4. Editing an Existing logging.level Configuration

MICROSERVICE LOG FILE LOCATIONS

In a typical single host deployment, CAS microservice log files follow this directory pattern for locations:

Directory: /opt/sas/viya/config/var/log/<microservice-name>/default/

Log file: sas-<microservice-name>_<start-timestamp>.log

Note that there can be multiple log files present if the service has been restarted. In most cases, you work with the most recent log file, or with the log file that covers the time period of interest.

Here is an example location for a typical deployment for the Report Packages microservice:

/opt/sas/viya/config/var/log/report-packages/default/sas-report-packages_2017-01-06_21-38-00.log

LOG METRICS FOR REPORT VIEWING

After completing the logging configuration updates, execute a report and look at the end of the Report Packages log (Figure 4).

```

2017-01-12 21:44:39.396 DEBUG 1344 --- [pool-6-thread-3] c.s.r.service.utils.ReportGenerator : sasszs [] 0: get bird provider for report - 0
2017-01-12 21:44:39.396      781: retrieving report - 781
2017-01-12 21:44:39.396      821: capturing original xml - 40
2017-01-12 21:44:39.396      1102: done gathering style-related files - 281
2017-01-12 21:44:39.396      1102: start gathering images - 0
2017-01-12 21:44:39.396      1102: done gathering images - 0
2017-01-12 21:44:39.396      1765: executing queries - 663
2017-01-12 21:44:39.396      1765: done gathering results - 0
2017-01-12 21:44:39.396      1963: retrieving file metadataSupplement - 198
2017-01-12 21:44:39.396      1964: saving file to cache metadataSupplement - 1
2017-01-12 21:44:39.396      1964: filling out data sources - 0
2017-01-12 21:44:39.396      1967: start saving report to cache - 3
2017-01-12 21:44:39.396      1967: done saving report to cache - 0
2017-01-12 21:44:39.396      1967: checking report status - 0
2017-01-12 21:44:39.396      1968: start saving cached report to cache - 1
2017-01-12 21:44:39.396      1968: done saving cached report to cache - 0
2017-01-12 21:44:39.396      1968: done building package - 0
2017-01-12 21:44:39.396      Total elapsed time (ms) for Generating Report - /reports/reports/aa20f9e5-d49f-4340-a0ac-ca4ad381b4b7 - nodata = 1968

```

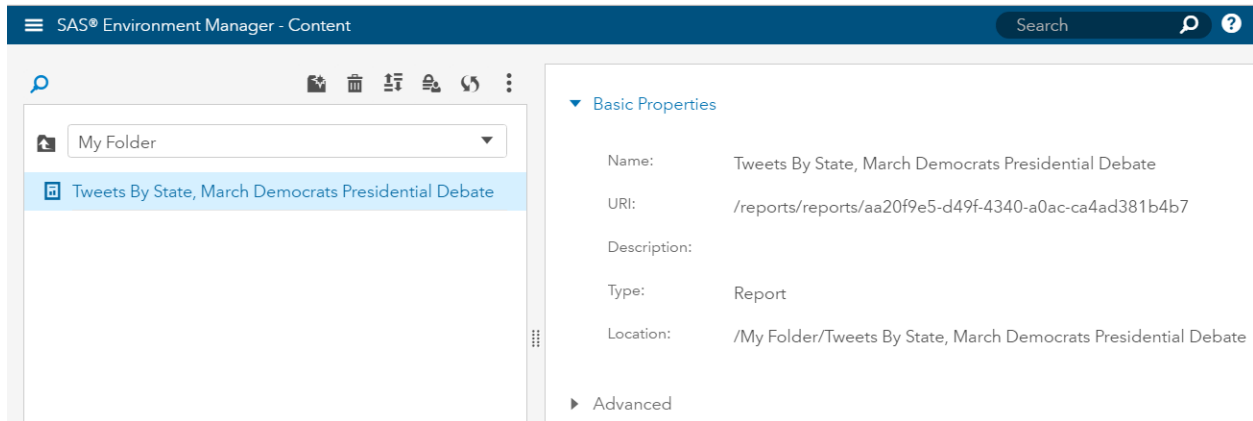
Figure 4. Report Packages Log Snippet, Showing Report Performance Metrics

With the DEBUG logging levels in place, each report execution generates these performance metric log lines. Not every report generates the same number and types of events. Times are reported in milliseconds for each step in report package generation for the viewer.

As expected, for most reports, executing queries (CAS data queries and analytic actions) takes most of the time, and this area is where you will likely focus for performance enhancements. The DEBUG log statement also shows the user ID for the user viewing the report (the user ID in Figure 4 is sasszs) and the site's unique report URI (Uniform Resource Identifier), which is

`/reports/reports/aa20f9e5-d49f-4340-a0ac-ca4ad381b4b7`

The URI can be mapped to the report's title and folder using the SAS Environment Manager Content window (Display 4). As an administrator investigating a user issue, you might want to first locate the user's report and URI, and then search the microservice logs for the URI string.



Display 5. SAS Environment Manager Content Window and Report URI

ANALYZING THE DATA

A favorite tool for processing log files is Python (python.org), because of its speed in processing text files. Looking at the performance log data shown in Figure 4, we see:

- A log line, identified by a key word, for the start of a report viewing event
- A list of report rendering tasks, with durations in milliseconds for each task
- A log line, identified by a key word, for the end of the report viewing event.

We can use Python to read the Report Package microservice log, find each report viewing event, save the user ID and report URI, and load the data into a list of Dict data structures. After processing the log file, the code writes the data to a CSV file for import as a CAS data source to use in SAS Visual Analytics.

Working with a time series analysis, you immediately see that the CAS server query step (data query and analytic action results) comprise the largest amount of time for most reports. This is not surprising, but does cause Y-axis scaling issues for our time series (Figure 5). With query times as the longest task, the

times for other steps are lost, because they are too small compared to the larger scale of the query times. To remedy this problem, scale the time for each step by dividing by the total time. This provides a measure of time spent in each task relative to total time for report execution, as a percent of total time.

After this adjustment, we can better see user patterns over time and how performance might change with time of day workload (Figure 6).

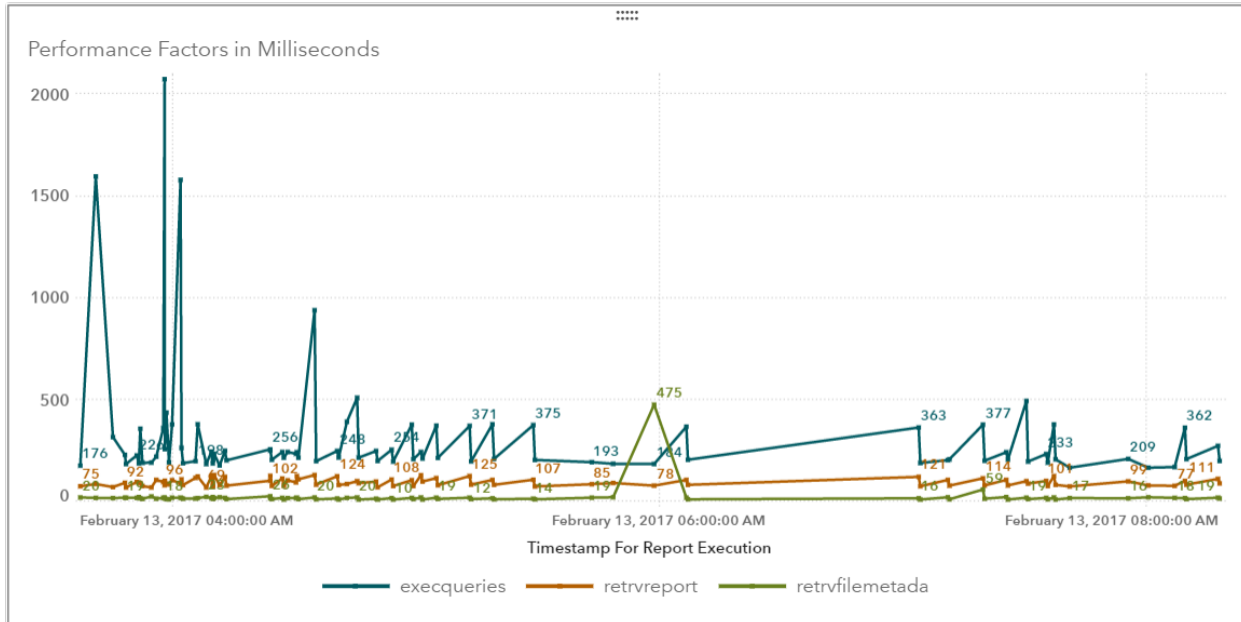


Figure 5. Time Series Analysis of Selected Report Timing Metrics (milliseconds for each report task)

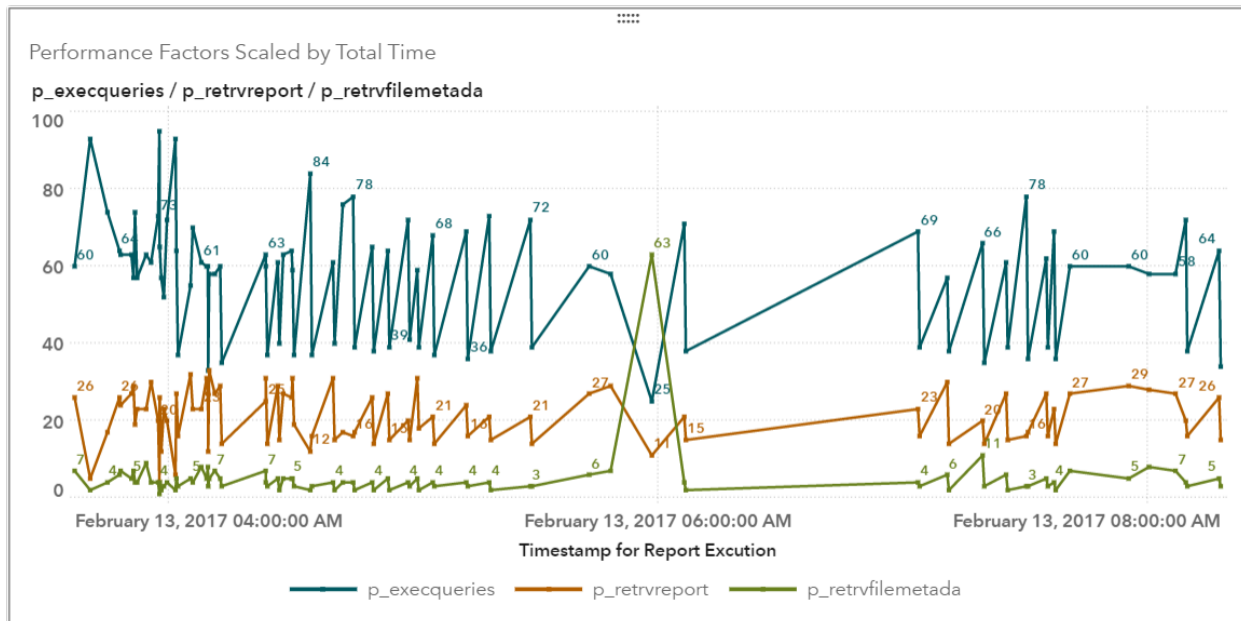


Figure 6. Time Series Analysis of Report Timing Metrics (task times as percent of total report time)

CONCLUSION

Cloud Analytics Services is the core technology supporting SAS web applications in SAS Viya. SAS Environment Manager provides an easy to use interface to update CAS microservice configurations. The design of CAS microservices allows for configuration updates to be made while the service is running.

For microservice logging, the SAS administrator can investigate user concerns from the console by adding strategic DEBUG logging levels. Administrators can activate service logging to collect performance metrics from the reporting microservices that support the SAS Visual Analytics viewer. The data collected can be analyzed for a single user and report execution, or as a time series for all users and reports over time.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Scott Sweetland
Visual Analytics R&D
100 SAS Campus Drive
Cary NC 27513
SAS Institute Inc.
Scott.Sweetland@sas.com
<http://www.sas.com>

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.