SAS® GLOBAL FORUM 2020 #SASGF

Paper SAS4408-2020

Essential Performance Tips for SAS® Visual Analytics

Meera Venkataramani, SAS Institute Inc.

ABSTRACT

Troubleshooting performance-related issues across distributed systems is a challenging task, and it requires a step-by-step approach to identify the cause of the bottleneck. When it comes to performance, there are a myriad of factors involved, with each one contributing in its own way to the overall problem. Because SAS® Viya® is a broad system with many distributed layers, the variance and complexity is what makes the problem hard to solve, and often causes approaches to fail on given architectures. This paper takes a holistic look at the SAS Viya and explores the various processes and methodology involved in diagnosing performance problems with SAS® Visual Analytics. Using a customer use case, we will demonstrate to you the diagnostic steps, forensic process, and the tools and methods involved in this process. In order to get the best user experience and optimum performance with SAS Visual Analytics, thought and effort should be put into architecting the application and the underlying infrastructure layers. The various components of the system should be built to be highly responsive on all the different components of the application service. As if peeling an onion, we'll examine each layer of the SAS Visual Analytics until we arrive at its core. Each layer has a rich set of performance-related information to offer along the way…and we promise that it won't bring tears to your eyes.

INTRODUCTION

Identifying performance issues distributed across SAS Viya is a challenging task and it will need a step-by-step approach to identify the cause of the bottleneck. When it comes to performance, there are a myriad of factors involved with each one contributing in its own way to the overall problem. Sometimes, it's like finding a needle in a haystack. Isolating the various factors is the key.

This paper provides guidelines and basic validation tests to identify what and how to solve these issues. Eliminating and identifying what is SAS and what is outside SAS early on gives us an edge. We have seen over the years that everyone has a different approach to solving performance issues and everyone has their own set of steps and tools they use to start this evaluation. This paper is an effort to gather, review, test, and document all of these tools in one place and to provide a step-by-step framework to solve performance issues.

WHY IS APPLICATION PERFORMANCE IMPORTANT?

Inadequate software performance that refers to slow system response times and transaction throughput rates is the biggest problem in production systems these days. Slow degraded application performance is highly disruptive, inefficient, and can cost the business a lot of money in terms of system downtime, delayed decisions, and lost revenue. Business continuity and application performance are dependent on each other, and critical for success.

In recent years, software applications have evolved to be more complex, dynamic, robust, and distributed in nature. Applications have also changed their formats, starting with stand-alone hardware to client-server and then mobile and distributed cloud-based solutions. The introduction of mobile and cloud computing technology has also changed how applications are used today. In today's world, corporations cannot afford to have any degraded performance or system downtime. Applications must be fully functional, efficient, and readily available around-the-clock to deliver the most optimal performance and a high-quality experience at all times to their customers. Would you like it if your Netflix movie took far too much time buffering and occasionally it might freeze and drop out?

SAS is committed to delivering the best optimum performance with our application to our customers. However, dealing with system performance is a joint responsibility between SAS, the customer, the customer’s IT organization, and even other third-party service providers. Customers expect SAS to understand how to get the best performance, even though we don’t control all aspects of the issue. Solving such problems often involves getting several teams in a room to work through the various layers of the application and infrastructure.

TROUBLESHOOTING A PERFORMANCE ISSUE CAN BE A NIGHTMARE

Why is troubleshooting so hard you may ask. Application performance is impacted by the components used to deliver the service to the user, the application’s user interface, and the connectivity between these components. The variance and complexity make the problem hard to solve, and often causes approaches to fail on given architectures.

Applications are distributed by nature, and unless the underlying infrastructure is responsive on all the different components of the application service, the entire application service is impacted. One of the most critical factors that affect application performance, and often the hardest to identify and track, are application dependencies on supporting applications, as well as the underlying system and network components that connect them all together. With the advent of virtualized servers and networks, the complexity of the application delivery infrastructure has increased significantly. The challenge is finding an application performance monitoring solution that can automatically discover and monitor the network and server topologies for the entire application service.

Today's distributed applications, such as SAS Viya — particularly for large organizations — can have thousands of individual connections stretching across many tiers and even reaching outside services. We've moved beyond simple 3-tiered web applications into complex distributed applications (made up of load-balanced web and application servers, multiple layers of middleware and databases, storage arrays, mainframe transactions, and even outside services). In this world, problems are no longer concentrated in application code, instead they are randomly distributed throughout the application infrastructure. It could be anywhere in the stack, for example, servers, operating systems, storage, networking services, LDAP, anti-virus, database, firewall, and DNS misconfiguration all can create application problems; and that's just the tip of the iceberg.

In these situations, you need to be able to think outside of the box for new ways to tear into the problem. That might involve breaking out medieval tools, such as a protocol analyzer (like Wireshark) or process tracer (for example, truss, strace, procmon, regmon, and so on). However, being able to use those tools to good effect basically requires you to be curious about how things work at a basic level. As with communication, not every IT pro has this skill.

However, the most difficult performance problems won't be untangled that easily. The really fun ones leave no trace of their existence. Performance graphs will look well within limits, error logs will be devoid of useful errors, and hardware/software specifications will all be satisfied

SAS VIYA ARCHITECTURE

Not only does SAS Viya bring exciting advancements in high-performance analytics, it introduces several modern practices to SAS software architecture. SAS Viya brings a more resilient, elastic, unified, and open architecture, which leverages cloud-friendly microservices and a next generation analytics run-time engine.

To administer SAS Viya, you should have a good understanding of each of the following components, gain a better understanding of the components of the SAS Viya architecture, and how they can be collectively managed to keep your environment available, secure, and performant for the users and processes you support. Your system is only as good as the weakest link in your chain. For SAS Viya to perform optimally, each of the components in SAS Viya should also be performing optimally. Figure 1 shows a high-level architectural view of SAS Viya.

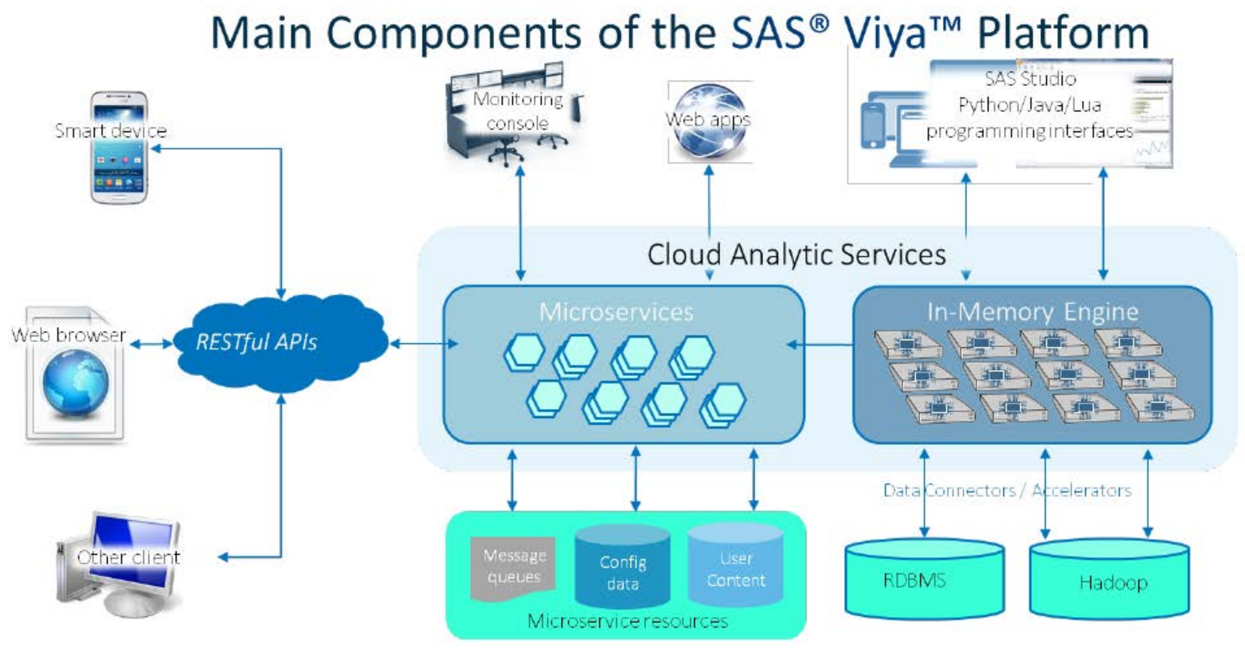

Figure 1. SAS Viya Architecture

IT INFRASTRUCTURE

Let’s see an example of what a customer’s IT infrastructure might look like.

Figure 2 illustrates how a modern organization might function in a global environment. It provides a high-level map or plan of the information assets in an organization, which guides the current operations. Notice the following in the figure:

- A front-end web server talks with an application server that talks with a middleware server that queries one or more database servers.
- Then, all of those servers might talk with DNS servers to look up IP addresses or map them back to server names.

When that happens, just one weak link slows the whole application down.

In order for SAS Viya to perform optimally – unless the underlying infrastructure that SAS Viya depends upon, is responsive on all the different components of the application service, the entire application service is impacted.

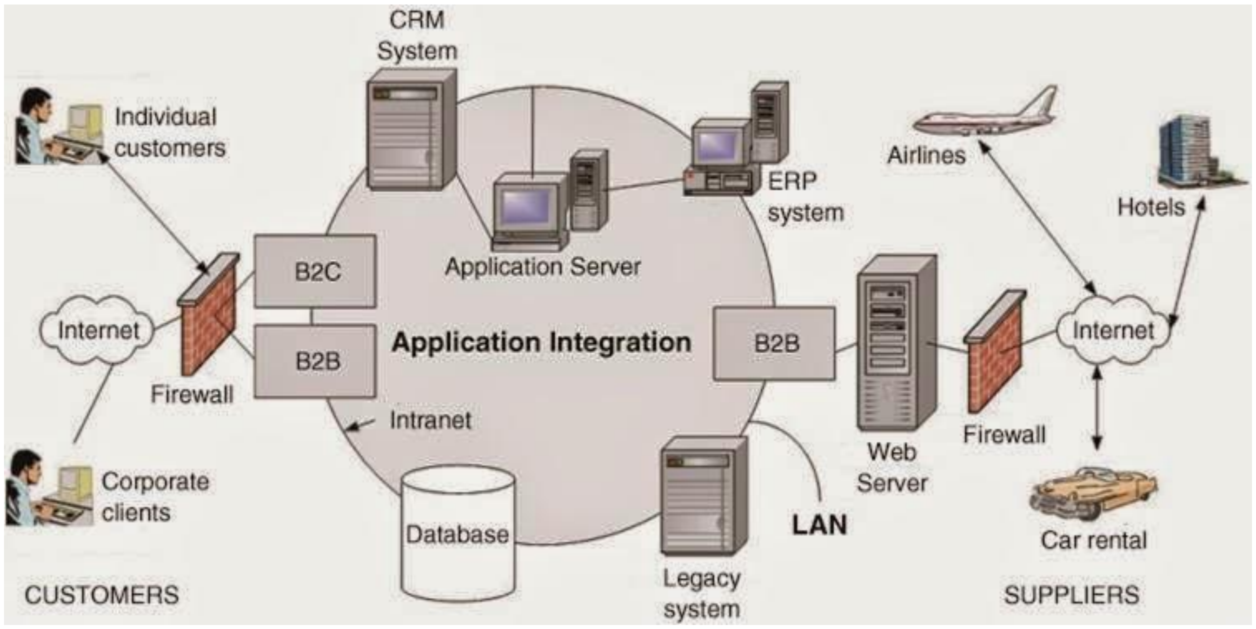

Figure 2. An Illustration of a High-Level Map of the Information Assets In an Organization

TROUBLESHOOTING: A STEP-BY-STEP PROCESS

Troubleshooting a performance-related issue is like peeling an onion, to get to the core you must progressively peel away and asking layers of questions as you go along, By asking deeper and deeper questions that build on the previous answers, you are more likely to peel the onion (so to speak) and discover what really matters. If this scenario brings tears to your eyes, I promise that the following process of uncovering core information by removing each layer, will not leave you red and irritated.

Now that we have talked in detail about the complexities of identifying a performance bottleneck in SAS Viya, let us take a deep dive into the how's of solving performance problems and designing high performant systems.

GENERAL CONSIDERATIONS AND GUIDELINES

Here are some general guidelines and best practices to keep in mind as you start your troubleshooting journey:

Designing for High-Performant Systems from the Start

Of course, the best way to troubleshoot performance problems is to prevent them in the first place by designing for high-performance from the start. Details, in the form of requirements, are just as important when designing a new high-performance solution as they are when you are troubleshooting problems. Here are questions to ask: How much data is involved? How much data traverses the system when a job is run? How much goes over the network or between the server and storage? Can we move data local to the system instead of remote? How many jobs are run, per hour, per day? How much data is processed during those jobs? How quickly do the jobs need to complete?

One of the biggest factors that impacts application performance is design. Performance must be designed in. When applications are specified, performance goals need to be delineated along with the details of the environment the applications will run in. Often development is left out of this and applications are monitored, analyzed and "fixed" after they are released into production. This never works as well as when performance is one of the key goals of the application design before a line of code is written.

Identifying the Exact Issue

The first step in troubleshooting a performance problem (really any problem) is to get a solid understanding of what the problem is. We often get performance problem reports that say simply, "It's slow." We need more details. For example: What's slow? What's the specific problem you're having that we can troubleshoot? The more detail you can give us the better. If there are multiple problems, tell us which one is the most important so that we can start there. If you can replicate the problem, please give us the specific steps that we can use to replicate your problem.

Getting an accurate description of the problem soon after it is first reported it's perhaps the most valuable clue you can work with in the troubleshooting process. I can't tell you how many times I have been in situations where we are chasing a problem that didn't actually exist -- at least not in the way that it was originally described. Not only is it a waste of time and money to chase ghost problems, but it's also intensely frustrating for stakeholders who see no progress being made on an issue they've reported.

Document Problems Correctly and Accurately

Often, asking an affected user to keep a log of exactly what happens -- and when it happens -- can really help when it comes to matching user complaints to system logs and performance charts. If you can coach the user on what to look for and how to document events accurately, you can save yourself a lot of trouble chasing the ghosts created by approximation, hearsay, or an overactive imagination.

Hardware Sizing

Hardware sizing is never an exact science, particularly when you consider the potential these days for database and user population growth. So, when you receive a sizing from SAS consider going over it with your SAS representative and confirming whether the sizing takes into account your best estimates for storage and usage growth in both the near term and the foreseeable future. All customers should have plans in place to manage application growth. Make sure you have adequately accounted for memory, CPU, and disk space.

Usage: Peaks and Lows

One of the most important factors that affects application performance is a poor understanding of how the application will be used (for example, how many people will simultaneously use it and for what kind of transactions), and the corresponding application architecture and its scaling assumptions that go into its design and deployment. This lack of understanding real user transactions and performance manifests itself as bottlenecks in performance during the most critical peak usage period.

Virtualization

In addition to latency and bandwidth being an issue, there is a good chance that customers are running SAS software on virtualized hardware that is shared with other applications. It is very challenging in this dynamic environment to understand what will impact your application performance, as it requires intimate knowledge of your ever-changing application structure at any given moment. For hardware or virtual machines, virtualization usually adds an overhead.

The modern application is complex, and a single transaction trace can sprawl across many layers in a virtualized, cloud world – a perfect storm impacting application performance. This growing complexity impacts application performance from the end-user experience all the way back through transactions, the application layer, application infrastructure, and IT infrastructure. The IT team should be able to run tests to validate that the virtualized hardware is not overcommitted and that resources are used optimally.

Browsers

Today’s web-based applications tend to push user-interaction work — often accompanied by lots of data — to the client workstation. From there, JavaScript code processes hundreds or thousands of rows of data, which can cause multi-second pauses before the client displays the updates.

A web browser is the key blind spot for gaining true end-to-end visibility into application performance. With new approaches to application design and the increased usage of web services, the ability to monitor the processing that takes place within the browser has become one of the key requirements for full visibility into application performance.

Applications perform differently in different browsers. Adequate testing and use cases should be validated using different browsers on different operating systems to make sure that your application performs optimally on all of the supported browsers.

INFRASTRUCTURE TESTS

Now that you have deployed SAS Viya, everything looks OK in your development and test environment, system is rolled out to production but you are starting to see problems. Tests that once seemed to operate smoothly in the lab are showing degraded performance and performing sluggishly. End users are complaining. Your boss is upset. The pressure builds for you to finally fix that slow application everyone depends upon. Where do you start?

Start by performing system-level baseline tests to rule out issues related to network and connectivity. Performing system-level baseline tests (outside of SAS Viya) to assess the heath of your system infrastructure is key in moving further with the diagnosis and will help save a tremendous amount of time. These tests allow us to rule out basic obvious issues associated with memory, network connectivity, latency, bandwidth, input/output, and disk space requirements. Let us look at some of these tests in greater detail.

Network Latency and Bandwidth

Latency and bandwidth are one of the most important pieces for applications that are distributed and span multiple layers. Monitoring network bandwidth and web application performance from multiple locations helps isolate the problem to the network tier. The network on which the application is used impacts performance tremendously, especially for mobile and cloud. Inconsistent bandwidth, high jitter, increased latency, and packet loss all work to degrade application performance. While you might not be able to control mobile or most cloud networks, you can build and test apps with these network conditions in mind. This gives organizations the best chance to optimize application performance before the network impacts are felt by users. Bandwidth bottlenecks are a big problem as they can cause network queues to develop and data to be lost, thus impacting the performance of applications.

tgridperf

tkgridperf is a network test provided by SAS that can be run on all of your SAS Viya nodes to isolate latency issues. The tkgridperf network test only takes a few minutes to run and has proven invaluable in several situations. In your SAS Viya deployment, you will find the script in the `/opt/sas/viya/home/SASFoundation/utilities/bin` directory.

To run the test, go to your SAS Cloud Analytics Services (CAS) controller node and issue the following command:

```
export GRIDHOST=<name of controller node>
export GRIDINSTALLLOC=/opt/sas/viya/home/SASFoundation/utilities
export GRIDRSHCOMMAND=/usr/bin/ssh
export TKPATH=/opt/sas/viya/home/SASFoundation/sasexe
cd /opt/sas/viya/home/SASFoundation/utilities/bin
./tkgridperf
```

Note: This example assumes that SAS is installed is under /opt/sas

The tkgridperf network test returns 3 results. Let us look at an example output from one of my systems that has one controller and three worker nodes.

```
bash@system02>./tkgridperf
Grid initialized with 4 machines
Time for bcast(100M bytes, 20 times): 6013 ms
Time for reduce(max, 4 bytes, 10K times): 5220 ms
Time for allreduce(max, 4 bytes, 10K times): 5156 ms
```

The bcast test sends 100M of data to all nodes 20 times. This is a test of throughput between nodes. This is more a measurement of bandwidth, because it's a big file, few times. The reduce test gathers 4 bytes from each node to the controller 10,000 times. This is a small amount of data, so this is mostly a latency test. The allreduce test gathers 4 bytes to all nodes 10,000 times. Again, this is a latency test.

The reduce and allreduce tests use really tiny data (4bytes), but it's 10,000 times. It tests latency, so if the latency is high, the value in milliseconds will be high. Ideally you want the value in milliseconds to be as low as possible.

The tkgridperf test will not tell you which node is slow. They will only tell you if the CAS grid as a whole is working well together. If you get a bad result and want to narrow down the results to find out which node is slow, you could start by lowering the number of worker nodes with the -procs option. You can also remove nodes from the machine list and try to isolate the slow node (or nodes) that way.

This test can be used as a first shot to identify basic obvious latency, bandwidth, and other networking issues. CAS is not used at all in these tests; however, the test uses the same communication library that CAS uses.

qperf

The Linux command, qperf, is another way to measure network bandwidth and latency performance. The qperf command works over TCP/IP, RDMA, and many other transports. It connects two nodes, with one node designated as the server (with no arguments) and a second node that runs with two arguments. Many options are available.

Live Performance Monitoring with Tmux and Dstat

Tmux and Dstat are great open-source performance monitoring tools to understand what is happening on your SAS Viya systems.

For Unix-like operating systems, tmux is a terminal multiplexer. You can access multiple sessions simultaneously from a single window or run multiple command-line programs simultaneously. Using tmux, you can create a monitoring system that enables you to remotely check your server for an overview of what’s going on.

Dstat is handy system administration tool for viewing Linux system resources. It generates useful system resource statistics in real time. (Dstat replaces multiple Linux tools, including vmstat, netstat, ifstat, iostat, and mpstat.) Dstat lets you monitor systems while you are troubleshooting, testing performance tuning, or benchmarking.

Opens Source Tools: Prometheus and Grafana

If you're using open-source tools for monitoring and alerts, then Prometheus is another toolkit to consider. Grafana is a visualization tool that integrates out-of-the-box with Prometheus, as well as many other databases. You can use Grafana to create a monitoring dashboard that consists of multiple charts and can easily be shared with your team.

SAS VIYA COMPONENTRY TESTS

Now that you have performed all the infrastructure-level baseline tests and are convinced that the various components of your infrastructure are performing optimally, you should test the SAS Viya components next.

CAS: gridmon Console Application

The very first test you want to use is the gridmon console application that helps you see what is happening further inside SAS Viya., However, gridmon is available only to a more sophisticated super user or administrator.

gridmon.sh is a console or terminal application that can be run from a Linux terminal or a terminal emulator like PuTTY. gridmon.sh displays data that is streamed from all of the machines on your CAS server. It shows information about jobs, individual machines on the server, and the attached disks. The gridmon application shows the status closer to real time, and it has some useful additional details, such as the specific percentage of CPU, memory, and number of ranks (both active and pending).. gridmon.sh also lets you show ranks, kill jobs, or run the gstack application to collect results

gridmon.sh is a popular tool for anyone wanting to know everything about a SAS Viya system. You could use all the menu choices of gridman as a cheat sheet. It even has a record/playback feature so that you can make a recording at a customer site and send it to SAS Technical Support so that we can analyze it further.

The SAS documentation for gridmon.sh provides a complete list of the available commands. The descriptions should give you enough information for cases where you are looking to see for examples of which sessions/process are running which actions and using how much memory, disk space, and CPU are being used. More difficult problems often require a combination of detailed machine, operating system, and CAS knowledge. This is where the record/playback feature comes in. Customers can send their recordings, CAS logs, and a description of the symptoms to SAS Tech support, so that we at SAS can hopefully provide a diagnosis, or at least next steps.

Microservices Analysis

SAS Viya is based on a microservices architecture that structures an application as a collection of loosely coupled services. In a microservices architecture, the services are fine-grained and the protocols are lightweight. The benefit of decomposing an application into different smaller services is that it improves modularity. This makes the application easier to understand, develop, test, and become more resilient to architecture erosion.

We could use many of the operating system-level commands to identify the microservice that is consuming the largest amount of resources. Commands such as ps, top, and htop can be used to find this information. Once you have identified the most CPU-hungry process, you can find out which microservice it is.

Here is an example:

Use the `ps -ef | grep` sas command on the SAS Viya services machine. This gives you a list of processes and the memory consumption of each of the microservice. If you look closely, it also gives you the name of the microservice. The JAR name is in the full command line of the Java process. However, it isn't always visible when you use the `top` command.

Then you can use the JAR name to identify which microservices are consuming which resources.

Boemska Enterprise Session Monitor (ESM) for SAS^{TM}

ESM has been designed and built from the ground up for the purpose of monitoring the SAS platform. It monitors individual server-side processes for processor, memory, and disk space usage, and lets users contrast the performance of their sessions against the overall performance of the server they are executing on – all via an intuitive, user-friendly web-based interface.

ESM enables developers to proactively profile and predict the performance of their programs, reports, and stored processes prior to releasing them to end-users. The functionality offered by ESM can save developers a considerable amount of time and effort in diagnosing and solving common problems that are otherwise difficult to pinpoint, regarding both the performance of individual jobs and the stability of an environment as a whole.

As well as streaming them in real time, ESM stores the metrics it collects for later inspection, letting users and administrators retrospectively analyze the performance of jobs that run unattended during the overnight batch window. The profiles of individual jobs are searchable by Job Name or Completion Status and presented in an easy to read Gantt-style chart, making the root cause of multiple failures easier to spot. Any warnings or errors are flagged as they occur, and clicking on a warning or error instantly displays it in the wider context of the job log.

ESM is invaluable for remedying deployments where stability and timely completion of the batch are critical. It provides unprecedented visibility of overnight system behavior, reducing the time taken to diagnose a failure by an order of magnitude and giving administrators the time and information that they need to stop it happening again.

To get access to ESM contact your SAS account manager, who will be able to put you in touch with Boemska. Alternatively, you can contact them directly via their website: boemskats.com.

SAS VISUAL ANALYTICS TESTS

Now that we have done all the infrastructre-level baseline tests and tested the SAS Viya components, we can start looking at the SAS Visual Analytics application layer itself.

Today's web-based applications tend to push user-interaction work — often accompanied by lots of data — to the client workstation. From there, JavaScript code processes hundreds or thousands of rows of data, which can cause multi-second pauses before the client display updates. The web application becomes a very important layer and can be the cause of many bottlenecks.

By using a real browser, you can measure the time it takes the browser to completely finish rendering the page. This takes into account things like CSS/JavaScript download, JavaScript processing, DOM layout rendering, and so on. Yet, with the advancement of technology, HTML5, JavaScript, and CSS improvements, more and more logic and behavior have been pushed down to the client. This adds to the overall perceived performance of website or web application.

There are several ways you could go deeper into the SAS Visual Analytics application layer itself, such as analyzing the SAS Visual Analytics logs from a SAS Viya system inspecting the web page using browsers development tools, or running a network sniffing tool such as Wireshark or Fiddler.

Analyzing the SAS Visual Analytics Logs

The SAS Visual Analytics logs hold a wealth of information about your application, user activity, timings, and behind the scenes queries. The logs can be enabled for debug to get more verbose and detail information from the logs. When all other tests turn out clean, and you still need to be able to identify the culprit, you can turn to these logs for more information.

My SAS Communities article Debugging SAS Visual Analytics Report Performance Problems explains in detail the step-by-step procedure used to debug report-level performance issues in SAS Visual Analytics application. The article also explains how to look at the logs, what to look for, and how to identify the culprit that is causing the bottleneck.

Developer Tools

All modern browsers and most other environments support debugging tools, a special UI in an application that makes debugging much easier. Developer tools enable you to trace the code step-by-step to see what exactly is going on. These are all tools that are built into the browser and do not require additional modules or configuration. If you are ambitious and curious, these tools can give you a lot of insight into how your SAS Visual Analytics application is behaving.

The Fiddler Web Debugging Tool

Fiddler is a web debugging proxy tool that logs all of the HTTP(S) traffic between your computer and the internet. Fiddler allows you to inspect traffic, set breakpoints, and “fiddle” with incoming or outgoing data. Fiddler is freeware and can debug traffic from virtually any application that supports a proxy, including Internet Explorer, Google Chrome, Apple Safari, Mozilla Firefox, Opera, and thousands more. Fiddler is particularly useful in recording the HTTP requests between your browser and the remote HTTP servers.

Fiddler can also export the recorded HTTP requests in various formats include HAR files. It is very easy to download and use and I like the way the UI displays the traffic captures.

You can download Fiddler from here: www.telerik.com/fiddler. Installation is very easy and straightforward. Once you’ve installed Fiddler, open it, and in the upper left corner select File > Capture Traffic to start capturing HTTP traffic.

SAS TECHNICAL SUPPORT

SAS Technical Support's mission is to help our customers make the best use of our software products through effective and responsive support, active advocacy, and a broad and flexible range of self-help resources. Get world class technical support via our SAS tracking system and please visit our wealth of knowledge-based resources at SAS Technical Support.

Also please visit the SAS Support Communities for help if you are stuck on a problem. While you're there, get a SAS tip and share what you know. This community of SAS experts is there to help you to succeed.

CONCLUSION

While each problem is different and brings its own complexity with it, the general guidelines I've laid out and tests I recommended should help you diagnose or get closer to identifying the bottlenecks in your SAS Viya applications. Most often, precious time is lost from when the problem starts occurring to when a final diagnosis is made. And, lot of this downtime and frustration caused by this time loss can be avoided if the obvious basic issues with infrastructure, network, connectivity, latency, and other such issues are ruled out early on and we are focusing only on the complex and difficult hard to find issues.

Lot of times the problems are complex and come masked in as something so hairy and weird that we spend endless wasted amount of time chasing the wrong issues, when a simple test to check network or connectivity could have given us useful information to begin with.

The forensic process is almost like watching a mystery thriller and requires you to be curious and adventurous. It is not for everyone and certainly not for the faint at heart. For those who do want to be adventurous and embark on this adventure, you get to be the detective here. Either way, remember SAS Technical Support is always there to help you, so engage them early in your troubleshooting journey.

REFERENCES

Brown, Tony. 2019. "Engineering CAS Performance Hardware Network, and Storage Considerations for CAS Servers." *Proceedings of the SAS Global Forum 2019 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2019/3351-2019.pdf.

Crevar, Margaret. 2020. "Important Performance Considerations When Moving SAS® to a Public Cloud." *Proceedings of the SAS Global Forum 2020 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4312-2020.pdf.

Ellington, Bryan. 2020. "SAS® Viya® Monitoring Using Open-Source Tools." *Proceedings of the SAS Global Forum 2020 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4214-2020.pdf.

George, Johann. "qperf(1) – Linux man page. https://linux.die.net/man/1/qperf.

Kuell, Jim. 2020. "Diagnosing the Most Common SAS® Viya® Performance Problems" *Proceedings of the SAS Global Forum 2020 Conference*. Cary, NC: SAS Institute Inc. https://www.sas.com/content/dam/SAS/support/en/sas-global-forum-proceedings/2020/4296-2020.pdf.

SAS Institute Inc. *SAS® Cloud Analytic Services*. Cary, NC: SAS Institute Inc. https://go.documentation.sas.com/?cdcId=calcdc&cdcVersion=3.5&docsetId=calserverscas&docsetTarget=n08000viyaservers000000admin.htm&locale=en#n08193viyaservers000000admin.

SAS Institute Inc. SAS Note 42197. “A list of papers useful for troubleshooting system performance problems.” http://support.sas.com/kb/42/197.html.

SAS Institute Inc. SAS Support Communities. https://communities.sas.com/.

Venkataramani, Meera. “Debugging SAS Visual Analytics Report Performance Problems”. https://communities.sas.com/t5/SAS-Communities-Library/Debugging-SAS-Visual-Analytics-Report-Performance-Problems/tac-p/474291#M2953. Last modified September 7, 2018.

Website of Boemska. https://boemskats.com/products/esm/.

Website of Grafana Labs. https://grafana.com/grafana/.

Website of Prometheus. https://prometheus.io/.

Website of Telerik. https://www.telerik.com/fiddler.

Website of Tmux. https://github.com/tmux/tmux.

Wieers, Dag. “Dstat: Versatile Resource Statistics Tool”. http://dag.wiee.rs/home-made/dstat/. Last modified March 28, 2016.

CONTACT INFORMATION

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

Meera Venkataramani
SAS Campus Drive
Cary, NC 27513
SAS Institute Inc.
meera.venkataramani@sas.com
www.sas.com