Paper 3490-2019

Solving SAS® Performance Problems: Our Methodology

Jim Kuell, SAS Institute Inc., Cary, NC

ABSTRACT

Diagnosing performance issues can be a lengthy and complicated process. For many, the most difficult step is figuring out where to begin. This typically leads to a track being opened with SAS® Technical Support. The SAS Performance Lab has developed a standard methodology for diagnosing performance issues based on years of experience doing so both internally and at customer sites. This process is regularly applied when assisting with performance issues in SAS Technical Support tracks. This presentation goes through the methodology used by the SAS Performance Lab to diagnose performance issues and discusses resolutions to the most common problems.

INTRODUCTION

There are a multitude of factors that go into creating and maintaining a consistently well performing SAS compute infrastructure. Because of this, properly diagnosing and correcting performance problems can be a very tedious process. Through years of experience, the SAS Performance Lab (SPL) has created and refined a standard methodology for diagnosing performance issues. The methodology always begins with the same step: gathering as much information about the performance issue as possible. All the information is then analyzed and correlated, creating a story that leads us to the root cause of the issue. While every situation varies in complexity, the following causes account for the vast majority of performance problems: I/O infrastructure bandwidth, server hardware or network provisioning, operating system-level tuning, and application or data management.

This paper provides an overview of the SPL's methodology and how it is used to diagnose SAS 9.4 performance issues in Red Hat Enterprise Linux (RHEL) environments.

GATHERING INFORMATION

Gathering information is the most vital step in the SPL's diagnosis process. It is extremely important to fully understand all aspects of the environment and of the performance problem at hand. This information is used to create a story that eventually leads us to the root cause of the issue.

Below is the standard set of initial information that the SPL requests when approached to assist with performance issues. While some situations require additional data collection, this set of information is often enough to determine the source of most issues.

PROBLEM DEFINITION

The first piece of information needed is a very detailed definition of the performance problem. Finding the origin of a problem is impossible without first clearly understanding what the problem is.

APPLICATION DEFINITION

Different applications have varying impacts on the performance of a system. It is important to compile and map out a list of *all* applications (SAS 9.4 and others) that interact with each of the problematic compute systems. This includes the interface that is used to run the SAS jobs in question (that is, SAS Enterprise Guide, SAS Studio, batch, and so on).

In this list, you should identify the specific SAS 9.4 applications that appear to be exhibiting the performance problems.

INFRASTRUCTURE DEFINITION

Many performance problems can be traced back to issues within the infrastructure, either hardware- or software-based. This includes the servers, networks, operating system-level tunings, and I/O subsystems. Gathering and understanding this information might require the assistance of the Systems, Network, or Storage Administrators. Collecting the following information for each of the problematic systems helps us begin creating a detailed infrastructure mapping:

1. Server and Network Information
    a. Manufacturer and model of the systems
    b. Virtualization software being used, if any
    c. Model and speed of the CPUs
    d. Number of physical CPU cores
    e. Amount of physical RAM
    f. Network connection type and speed
2. OS-Level Information
    a. Operating system version
    b. File systems being used for both permanent SAS data files (SAS Data) and temporary SAS data files (SAS Work)
    c. Source data locations (for example, SAS data files, external database, Hadoop, and so on)
3. I/O Subsystem Information
    a. Manufacturer and model number of the storage array and/or devices
    b. Storage types and physical disk sizes, as well as any relevant striping information (that is, RAID, and so on)
    c. Types of connections used (for example, NICs, HBAs, and so on), the number of cards and ports, and the bandwidth capabilities of each (for example, 8 Gbit, 10 Gbit, and so on)

The SPL has also developed a tool that gathers and packages the output of various operating system-level commands and files. This tool is called the *RHEL Gather Information Script* and a link to download this tool can be found in the *Output From Tools* section below. The output of this tool provides a more detailed look at the OS-level information and is used as a supplementary knowledge base to the information that is listed above. It gives us with a closer look at specific OS tunings, user `ulimit` information, logical volume configurations, and much more.

SAS LOGS

SAS logs from the jobs that are surfacing the performance issues are vital to the diagnosis process. They contain several performance metrics and session options from a SAS job's run that help us narrow down the source of an issue. However, there are several additional SAS options that print much more information to the logs and greatly increase their value. This

information is already collected by SAS, but it is not printed to the logs by default. The following statements should be added to the problematic SAS jobs:

```
options fullstimer source source2 msglevel=i mprint notes;
options sastrace=",,,dsa" sastraceloc=saslog nostsuffix;
proc options;
run;
libname _all_ list;
/*  YOUR EXISTING PROGRAM goes here   */
```

The FULLSTIMER options statement tells SAS to print performance metrics about each SAS step to the SAS log. The SASTRACE options statement enables the reporting of information about external database activity to the SAS log.

If the SAS job in question ever ran without performance issues, it would be extremely beneficial to also collect the log from that run.

OUTPUT FROM TOOLS

The output files created by the tools in this section provide a vast amount of information about many different aspects of the environment in question. Further details about what each tool collects can be found in the links provided below.

1. RHEL Gather Information Script – This tool gathers and packages the output of various OS-level commands and files. It should be run on all problematic systems. For more information, see http://support.sas.com/kb/57/825.html.
2. RHEL I/O Test Script – This tool tests and provides the estimated available throughput of a file system for use with SAS by mimicking the way that SAS does I/O. It should be run against the file systems that house both the permanent (SAS Data) and temporary (SAS Work) SAS data files. For more information, see http://support.sas.com/kb/59/680.html.
3. IBM's `nmon` script – This free tool is our preferred hardware monitor for RHEL systems. It collects a plethora of information from the system kernel monitors, and its output can later be converted into a graphical Microsoft Excel spreadsheet. This tool should be run during the SAS jobs in question. For more information, see http://support.sas.com/kb/48/290.html.

When correlated with the information listed in the previous sections, the output from these tools is often enough to track down the root cause of most performance issues. Additional tools are available and used on a case-by-case basis when more complex issues arise.

ANALYZING THE INFORMATION

Issues with the hardware infrastructure account for nearly all the performance issues that the SPL encounters. Because of this, we begin our analysis by focusing primarily on the hardware. We start by looking for tell-tale signs of the most prevalent offenders: I/O waits, server hardware or network bottlenecks, and incorrect operating system tuning. Once we're able to rule out the hardware and operating system, we start looking into the environment's applications and data management.

The following sections provide a high-level overview of how we use the gathered information to track down several of the most common causes of performance issues.

Due to the concise nature of this paper, the information discussed in these sections is very limited. Please refer to the papers below for more details about these topics.

SAS LOG FULLSTIMER OPTION

The SAS log is typically where we begin our investigation. One of the options enabled by the SAS statements provided previously is FULLSTIMER. FULLSTIMER prints a set of performance metrics (already collected by default) to the log for each step in the SAS job. These metrics show us the execution times of each step and help us pinpoint where the problematic step is in the code, as well as the exact time when it was executed.

The three main FULLSTIMER metrics that we look at are Real Time, User CPU Time, and System CPU Time. Real Time is the amount of wall-clock time it takes for a step to execute. User CPU Time is the amount of time the CPU spends executing SAS code (both back-end and user-written code). System CPU Time is the amount of time the CPU spends executing operating system tasks. System CPU Time includes tasks that support running the SAS application. Total CPU Time is User CPU Time + System CPU Time.

The most common scenario we see is when the Real Time is significantly larger—15% or more—than the Total CPU Time (User + System). This indicates that SAS is in a wait state, very likely due to a hardware bottleneck. If you're experiencing sub-optimal performance when Real Time and Total CPU Time are within 15% of each other, this is very likely caused by the task being CPU-bound.

In addition, the SAS log also contains information about which file systems are used by each step. We can then examine those file systems that are used by the problematic step to determine whether they are configured correctly.

We cannot determine what the bottleneck is from the SAS log alone. We need to corroborate this information with the `nmon` hardware monitor. Since the log tells us the exact time that the step was executed, we can overlay that with the `nmon` output and isolate the data from that specific time frame only. We can then use the step's metrics from the SAS log to give us a better idea of where to look in the output.

NMON

IBM's `nmon` script collects a large amount of very useful information and monitors from the system. It has helped us diagnose countless performance problems across hundreds of systems. It's proven most helpful when used in conjunction with the SAS logs. Once we've reviewed the SAS log, we know the exact time frame to focus on in the `nmon` spreadsheet, and we have a general idea of what to look for. Keep in mind that individual tabs in the `nmon` spreadsheet give hints as to what the cause is, but no single tab paints the full picture. The information must all be pieced together to create a story that eventually leads to the root cause of a performance problem.

Before reviewing the `nmon` output, we convert it to a graphical Microsoft Excel spreadsheet using the `nmon_analyzer` tool. The resulting spreadsheet has the `nmon` output split into a series of tabs. Note that not all the information in the spreadsheet is represented in the tab's graphs. Scrolling up shows additional metrics and scrolling down occasionally shows additional graphs.

Below is a high-level overview of the `nmon` spreadsheet and where some of the most useful information can be found:

- Server Specifications and Information
- CPU
- Memory
- I/O
- Network

Server Specifications and Information

Useful tabs: AAA, BBBP

These tabs contain an abundance of server specifications and other output from OS-level commands.

CPU

Useful tabs: CPU_ALL, CPU001-CPU*nnn*

When looking into CPU usage, the CPU_ALL tab is a good place to start. This tab maps the different CPU percentages and allows you to easily see whether you are CPU-bound or have a high amount of CPU Wait. A high amount of CPU Wait indicates a hardware bottleneck. The CPU001-CPU*nnn* tabs break out the information from the CPU_ALL tab into the individual CPUs.

Memory

Useful tabs: MEM, VM

The MEM tab contains a lot of useful information about the memory activity on the system. Note that free memory dropping to zero does not necessarily mean that the system is out of memory. This means memory is likely sitting in the host system file cache waiting to be used. We typically only get concerned when free memory is at zero and there is a lot of paging activity on the system.

The VM tab contains virtual memory metrics. Specifically, this tab indicates file-backed paging and swap-space activity. Paging of several million KB/sec is normal. Paging starts to become worth looking into when it reaches 20 million or more.

I/O

Useful tabs: DISK_SUMM, DISKBUSY, DISKREAD, DISKWRITE

The DISK_SUMM tab contains read and write rates, as well as I/O per second, aggregated across all devices in the system. The DISKREAD and DISKWRITE tabs break out the respective I/O rates per device.

Note that all these tabs show how much activity the host believes is happening on the devices. These metrics can work as a rough estimate, but they are not always correct. Reports from the SAN are much more accurate and should be used if they are available.

Also, if storage is shared with any other applications, you must be careful how you interpret what you're seeing because the usage metrics reflect *all* activity.

Network

Useful tabs: NET

The NET tab contains metrics and several charts showing the per-device and total network usage.

RHEL I/O TEST SCRIPT

The most prevalent and noticeable cause of performance issues with SAS 9 is an insufficient I/O infrastructure bandwidth. This is because SAS handles very large amounts of data and has very different I/O patterns than most other applications.

The *RHEL I/O Test Script* mimics the way that SAS 9 does I/O and tests the throughput of a file system, giving an estimate measurement of its available bandwidth for use with SAS.

Running this script against all file systems that are used with SAS 9 – typically the permanent (SAS Data) and temporary (SAS Work) SAS data file systems – shows us how well the I/O infrastructure is performing. The minimum throughput that SAS recommends for compute nodes is 100-150 MB/sec per physical core. Keep in mind that this varies from site-to-site, depending on the number of users, data sizes, applications (and their expected response times), and so on.

There are many physical and software-based attributes that make up an I/O infrastructure, so it is important to work with your System, Network, and Storage Administrators if this is the location of the bottleneck.

RHEL GATHER INFORMATION SCRIPT

The SPL has worked very closely with Red Hat engineers to develop a list of best practices for Operating System tuning for use with SAS 9.4. A link to this document is in the References section. These best practices contain considerations and configurations that are generally applicable to most RHEL environments. However, they are not a one-size-fits-all recipe, so it is important to work with your System, Network, and Storage Administrators when tuning your environment.

The *RHEL Gather Information Script* collects and packages the output of various OS-level commands and files. It provides the information needed to validate that the operating system is correctly tuned as well as other useful system configurations. This tool should be run on all problematic systems in the environment.

REAL-LIFE SCENARIOS

Below are a few simple examples of real-life customer performance problems that were resolved using the SAS Performance Lab's methodology.

SCENARIO 1

A customer opened a SAS Technical Support track complaining of slower performance in their new, upgraded environment versus with their old, outdated hardware. The new environment had upgraded CPUs, faster storage, more memory, the works. However, the same code, running with the same data, was taking about 30% longer to run.

We reviewed the SAS logs from the job in both environments. On the old system, the Real Time and Total CPU Time of the steps were fairly balanced. On the new system, the Real Time was much higher than the Total CPU Time.

We asked the customer to run the RHEL I/O Test Script against the file system used by the job in both environments. The RHEL I/O Test Script results showed that the file system in the new environment had a much slower throughput than it did in the old, despite the new having faster storage.

We then looked at the RHEL Gather Information Script output for both systems. This showed us that the file system in the old environment consisted of several LUNs striped together using Logical Volume Management (LVM), and the file system in the new

environment consisted of several LUNs that were concatenated (not striped) together. Striping allows you to take advantage of the throughput of several LUNs at the same time. Concatenation limits you to the throughput of a single LUN.

Once the customer's administrators striped the LUNs instead of concatenating them for the file system in the new environment, the job's performance drastically improved and was then much faster than it had been in the old environment.

SCENARIO 2

This customer was not complaining about poor performance when they approached SAS. They approached us because their SAS log was showing a significantly longer Real Time than Total CPU Time and they couldn't figure out why.

While this is typically indicative of a hardware bottleneck, neither `nmon` or the other information we gathered were corroborating this story. The system appeared to be healthy and efficiently operating. We referred back to the SAS log and noticed that the steps with the disparate times were all connecting to and pulling their data from Oracle.

By default, SAS logs only print the CPU execution times of the SAS processes, so the time spent inside of Oracle was not accounted for. Once we enabled the SASTRACE options statement in the SAS job (as shown in the SAS Logs section previously), we were able to see information in the SAS log about how much time was used executing statements inside of Oracle and transferring data to SAS. The sum of the CPU times from the SAS processes and the SASTRACE output (Oracle) equaled the Real Time in the SAS log and validated that the system was, in fact, healthy and efficiently operating.

SCENARIO 3

This customer was experiencing a much longer execution time for a monthly reporting job than they had previously. The job was being run using the same hardware and data sizes and was always run during the same non-peak hours. This was confirmed by the information collected by the SAS log and RHEL Gather Information Script.

When correlating the SAS log and `nmon` output, we were seeing that the system was experiencing a much higher I/O wait during the most recent run than it did during the previous runs. We continued investigating and found another process, via the `nmon` output, that was running at the same time and had a very high I/O demand.

The customer brought the information to their IT administrators and found out that this process was a nightly system backup that had been enabled a few weeks prior.

CONCLUSION

When a performance issue arises, properly diagnosing it without the right tools and know-how can be extremely difficult. The SAS Performance Lab has used their methodology and standard toolset to diagnose performance problems for years. The purpose of this paper is to provide a high-level overview of how this is done in hope that it will help educate customers. This information can be used by customers to either begin a path toward self-diagnosis, or preemptively gather all the required information before opening a track with SAS Technical Support, leading to an expedited resolution.

REFERENCES

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

Red Hat. "Optimizing SAS on Red Hat Enterprise Linux (RHEL) 6 & 7." Available http://support.sas.com/resources/papers/proceedings11/342794_OptimizingSASonRHEL6and7.pdf.

ACKNOWLEDGMENTS

A special thank you to Margaret Crevar and Tony Brown for their contributions to this paper.

CONTACT INFORMATION

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

Jim Kuell
SAS Institute Inc
Jim.Kuell@sas.com