Paper 24

UNCORKING SAS/AF® BOTTLENECKS WITH THE SCL DYNAMIC PERFORMANCE ANALYZER

Jeff Lessenberry, Jeff Lessenberry Consulting Group

Abstract

This paper will discuss a relatively unknown but extremely useful SAS/AF tool: SCL Dynamic Performance Analyzer. This tool allows you to find the unseen gremlins slowing your SAS/AF or SAS/EIS code and eliminate them. The SCL Dynamic Performance Analyzer allows the program to run normally in a local or client/server environment while keeping detailed runtime information. When the SCL Dynamic Performance Analyzer is disengaged, it creates an interactive report, which shows a break down of each statement that ran during the application. This paper will discuss the usage of the SCL Dynamic Performance Analyzer and give examples of problem applications optimized using this tool.

Introduction

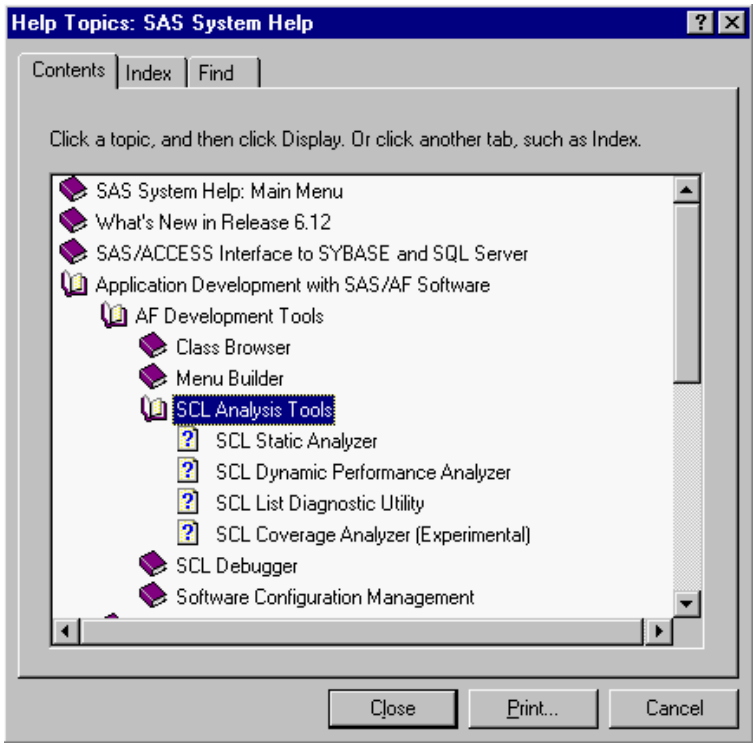
Software development is being influenced more by the deadline than by quality control, but both are just as important when it comes to making the user happy with the product. Most users run software with an imaginary clock that distorts reality and their ability to sense time. What seems like 10 minutes to user in reality might be 4 minutes. This only proves the need for speed optimization in software development. Developers do their best to create efficient and fast code but no one can for see every possible roadblock. These are some of the causes for most roadblocks:

1. Developing code locally but running it in a client/server environment
2. Developing code with many remote submits.
3. EIS applications with overriding user methods.
4. Finally, just inefficient coding.

All of the above causes can be found and corrected, but without the right analysis tool, you are just searching in the dark. The SCL Dynamic Performance Analyzer does not fix your problems, but it helps you find and identify them, which is the first step to a more efficient application.

Client/Server Problems

In 1996, I was given a new project using PC SAS, SAS/Connect, and a Unix machine, to put it mildly; I was ecstatic because this would be my first client/server application. Up to this point, I had only developed applications that would run locally, but I considered myself up to the task. It turned out to be one of the largest single projects I had worked on to date. I approached the design and coding of the application as I had any other. This included designing and coding the screens and base code on my PC first before going client/server. I thought 'Data is data whether it resides on my PC's hard drive or on the Unix box.' After getting the code to run perfectly on my PC, I decided to dive into the deep end and go client/server. I had the administrator set up my SAS/Connect script and I had coded the application with a macro variable that would let me run it locally or remotely. I took a deep breath, and started the application by logging onto the Unix box and activating the first screen. Everything was going great; the logon was successful, all of my remote library names were active, and the screen looked ok. After picking the data subset selections, I clicked the button to go to the next screen. I had done this process a hundred times before in testing, so I expected to see the next screen full of data in about 5-10 seconds. I waited and waited and waited. After 10 minutes, I hit ctl-break to cancel the screen, but this would not work in the client/server application. When I was able to break out of the screen, the SAS session crashed and it locked my connect session. I had no idea what the problem was because the log was gone. I went through this same process a couple of times until I was almost ready for the men in white coats to pay me a visit. I started looking for something, anything, in the online SAS help to shed some light on this problem. I happened to stumble into the AF Development Tools section, with which I had never worked with. There it was, the SCL Dynamic Performance Analyzer. I did not know what it was, but the words '**Performance**' and '**Analyzer**' got my attention.



I took the time to read about the tool and decided it could be the answer to my client/server application woes. I started the analyzer in my application startup and ran the application as before. This time I did allow the application get to the next screen; it only took about 30 minutes. I exited the application to see the analyzer menu appear on my screen. I went into the FUNCTIONS section and sorted the functions by time used. I found the function LVARLEVEL ran 5 times and took 26 minutes. That meant each one processed over 5 minutes. I had never seen this in my local testing and would not have expected the function to act in this manner. To fix this problem, I ran a Proc Summary against the dataset to retrieve the variables' unique values, then Proc Download it to the PC. This is not as fast as running the application against local data, but the time difference was small. This was the first of many findings in my local to client/server transition phase.

Next Up: SAS/EIS

The next development challenge for the SCL Dynamic Performance Analyzer was SAS/EIS. I had written some very involved SAS/EIS applications that included more method overrides than I like to think about. If you have done any SAS/EIS coding then you know that you need a flow chart to know what methods are running. I am not talking about a small list. A list of methods runs when you start a SAS/EIS application and a different list of methods runs depending on your screen action. This can be confusing if you are trying to debug a problem. Is your method running? Is it running in the right order so that your changes are not over written? Which method is taking so long? Well, the analyzer helped me find my way through this maze. It not only tells you about your methods but it also tells you about those little hidden internal SAS Institute methods. It helped me determine if my methods or the internal SAS Institute methods caused the problems. You might be surprised at the findings.

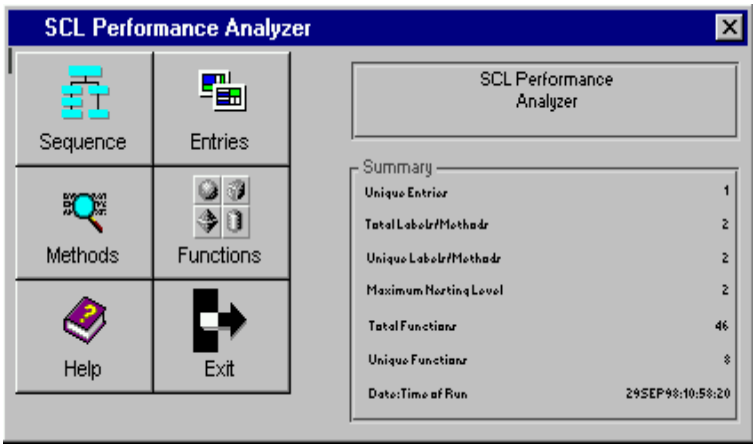
The SCL Performance Analyzer

To make this paper easier to read, SCL Dynamic Performance Analyzer will be referred to as 'the analyzer'. The analyzer may be used during the build process as well as a runtime tool. To activate the analyzer in a build environment simply issue the following command.

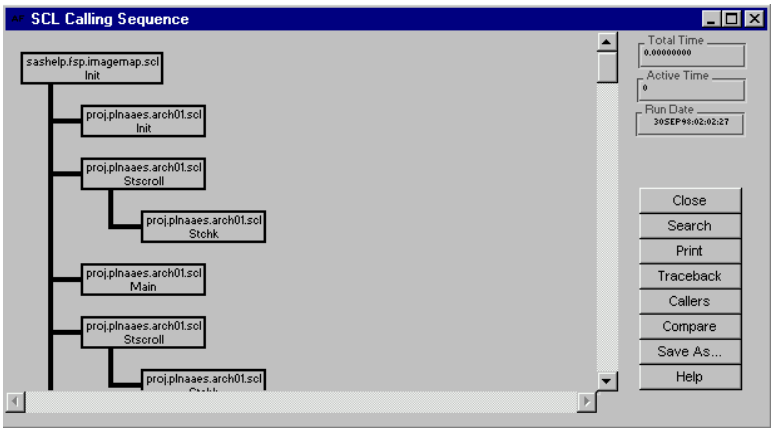
Command ===> SCLPROF TIMER ON (OFF)
or
AF CAT=MASTER.LOANS.MAIN.FRAME SCLPROF=TIMER

The analyzer is now active and ready to monitor an application run in the TESTAF environment. The analyzer will run until the program is terminated or until the analyzer is given the deactivate command. The analyzer can be activated in the middle of an application to monitor only one section of the code and then turned of in the previous manner. This is good way to monitor what happens when a button is pushed in different circumstances.

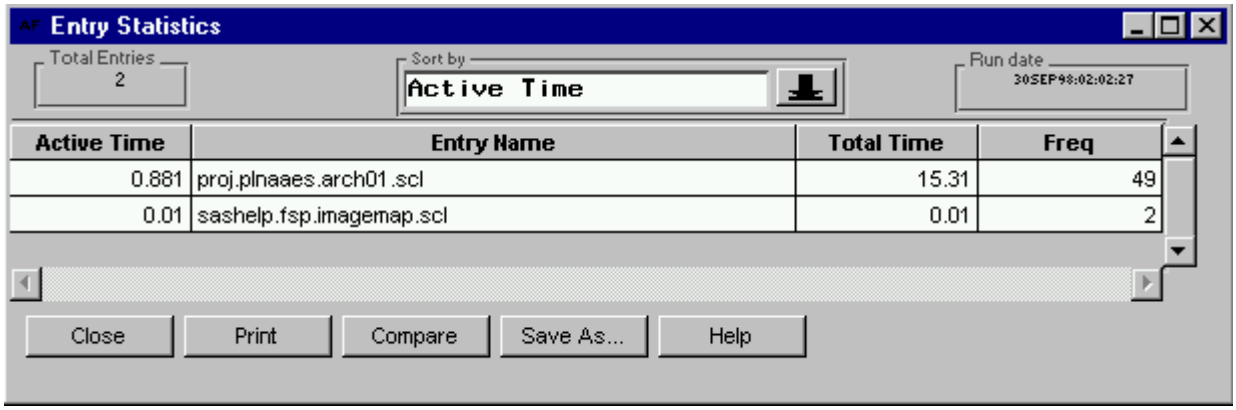
When the analyzer is deactivated, a menu screen is presented in the top left corner of your screen. The screen does contain some summary information about the monitored process.



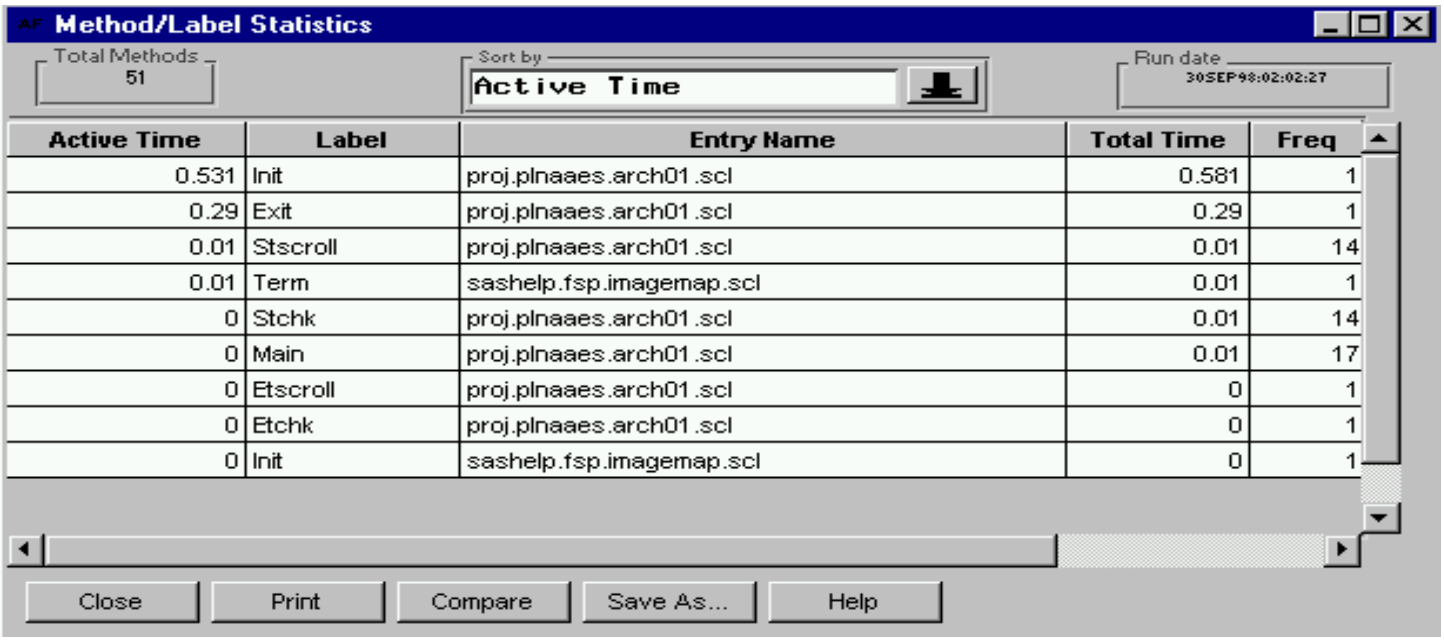
The **SEQUENCE** option is used to create a process tree of the sequences that ran while the application was active.

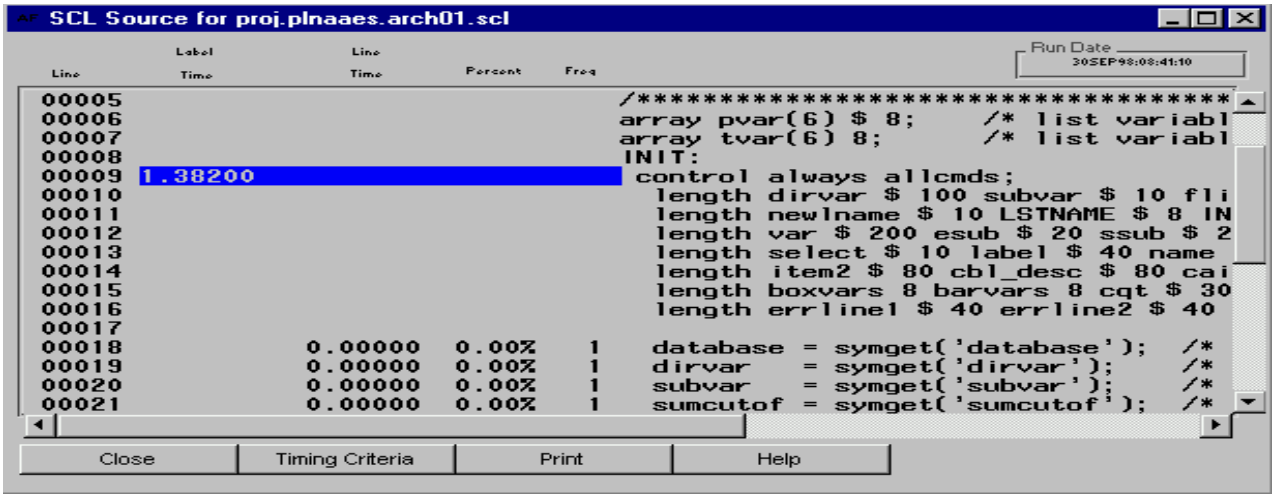


The **ENTRIES** option shows each entry that ran during the execution of your application. It gives the break down of time used for each one. By clicking one of the entries, the analyzer will show all methods/labels that were executed in that entry. This screen gives the same results as the **METHODS** option, except the methods option shows all methods/labels that were executed during the application.

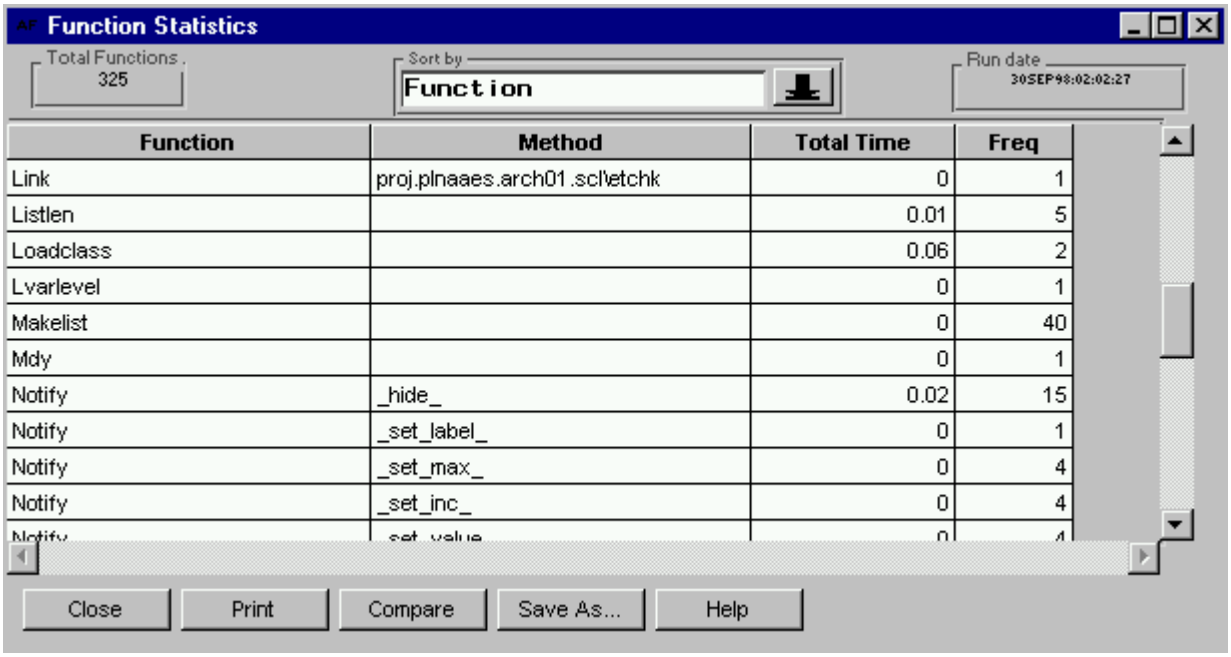


When a label is clicked, the analyzer shows that label in the SCL code. The next screen shows the total run time for that labeled section. The method screen will not only show the labeled section of your application, it will also show any internal SAS Institute label that ran during the application. The only problem is SAS Institute hides their SCL so if you click on their label you will only get a blank screen. Sometimes, it is good to know the SAS Institute labels which are running because they could be the gremlins slowing your SAS/AF application

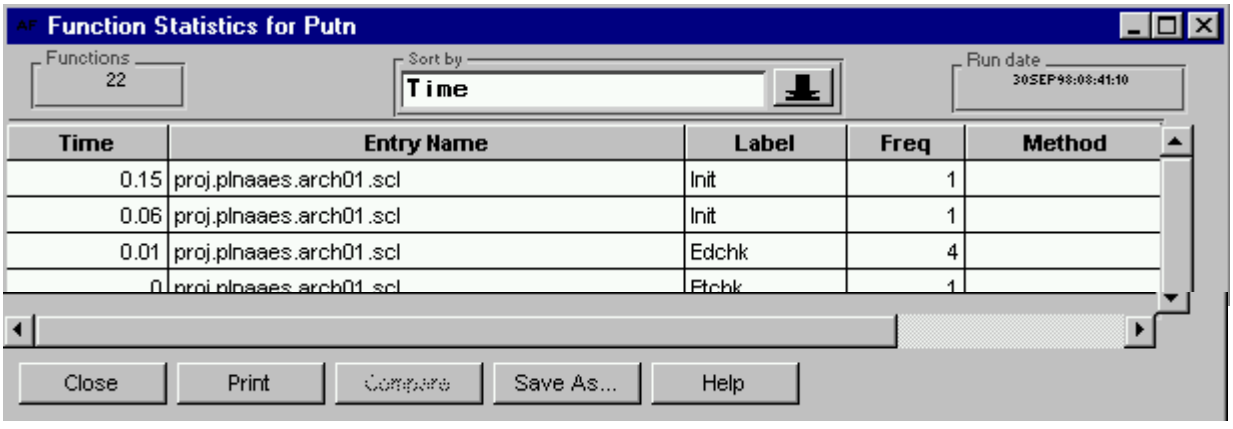


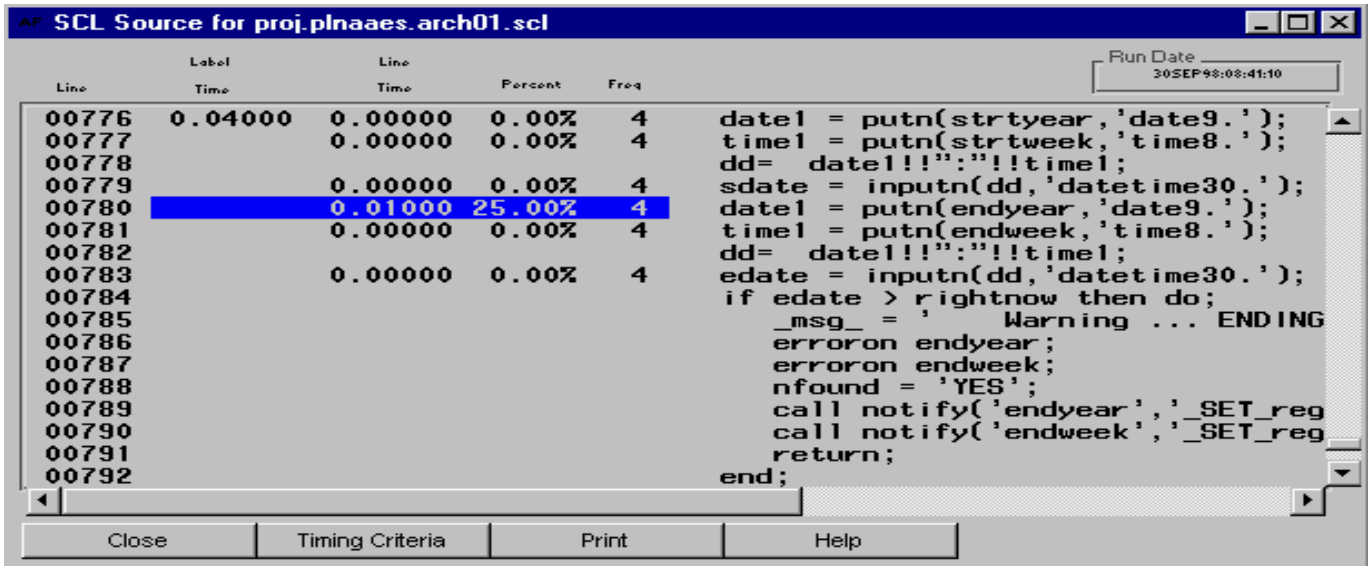


The **FUNCTIONS** option shows each function that executed during the application. This is the best way to gauge how well the application is coded. If a function like LVARLEVEL takes five minutes to execute one time, then there is probably a problem. The analyzer is the only way to know the total frequency and the total time it took to execute each line of code. When you click on a function, the analyzer lists the entries where the command executed.



The following screen shows where the PUTN command executed in the application.





By clicking on an entry name, the analyzer will display that function in the SCL code, which displays the runtime, percentage of total runtime for that section, and the number of times it executed.

Conclusion

The SCL Dynamic Performance Analyzer has proven its worth to me on many occasions. If used properly this tool can help make anyone a more knowledgeable and informed developer, but it is only one piece of the pie. We must continue to strive to increase our knowledge base and skill sets or we may find ourselves being outclassed by the individuals that do. If you want a repetitive job, I heard McDonalds' is hiring.

Acknowledgements

SAS, SAS/EIS, SAS/AF, SAS/CONNECT and all other SAS references are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

In memory of Steve Craig.
Like the starship Enterprise,
we took SAS where few had gone before.
You are sadly missed and the journey
will not be the same without you.

Jeff Lessenberry
Jeff Lessenberry Consulting Group
Telephone: (864) 243-9761
Email: JLCG@Mindspring.com