Paper SAS4102-2020

SAS® System Elasticity in Amazon Web Services Using SAS® Grid Manager for Platform

Glenn Horton, SAS Institute Inc.

ABSTRACT

In many systems, demand for computational resources varies over time. Wouldn't it be nice if the resources available at any given time to a SAS® system, and the associated cost, were scaled based on the demand for resources at that time? Now, that can happen when the SAS system lives in Amazon Web Services (AWS) where you pay for only what you use. SAS® Grid Manager for Platform knows how busy your system is and can dynamically grow the system when demand gets close to exceeding availability. And we can shrink the system when the demand decreases. The system can grow to a finite upper limit or unbounded. And all this is based on system attributes you define. This presentation describes how a SAS® system is configured to make this happen, the AWS infrastructure requirements, and system attributes used to grow and shrink resources.

INTRODUCTION

As SAS systems move to cloud platforms, interest is rising in using the elasticity features of those cloud platforms to keep cloud infrastructure costs down by scaling SAS systems up and down to meet changing and possibly unpredictable workload requirements.

It's always been possible to easily add machines to and remove machines from a SAS Grid Manager for Platform system while the system is running. However, those adjustments to capacity required someone to monitor system attributes and to take action to adjust capacity based on their interpretation of the attributes. The lack of automation also meant that either the same machine names must be used over and over or SAS system configuration changes must be made to accommodate the names of the machines currently in use.

Ideally the SAS system would monitor its own attributes and automatically take action to adjust capacity, either up or down, in response to attribute thresholds defined by the system administrator. As capacity is added, the names of the added machines are automatically added to the SAS system configuration, and as capacity is removed the names of the removed machines are automatically removed from the SAS system configuration.

This paper focuses on how those two capabilities can be added to SAS Grid Manager for Platform. First, we show how a SAS system can add and remove resources when administrator-defined thresholds are met. Second, we show how resources can be added to the SAS system configuration as they are started, and removed from the SAS system configuration as they are destroyed.

HOW CAN WE TELL THE **SYSTEM WHEN IT'S TIM**E TO ADJUST CAPACITY?

LSF has added a new component called resource connector. This component is a bridge between LSF and cloud providers. Although multiple cloud providers are supported by resource connector, in this case we are focusing exclusively on Amazon Web Services.

Resource connector allows an administrator to define system characteristics that trigger resource connector to add resources from the cloud provider to an existing LSF cluster. Resource connector also allows an administrator to define system characteristics that trigger resource connector to terminate instances provisioned by the cloud provider

HOW DO WE DEFINE WHEN RESOURCE CONNECTOR WILL ADD RESOURCES?

Demand for additional resources is calculated by a demand calculation program. The default demand calculation program can be replaced by the LSF administrator.

In the current version of resource connector, LSF must have at least one pending job before resource connector calls the demand calculation program.

We don't want to wait for SAS jobs to be pending before we try to scale up our resources. To avoid that, we can trigger resource connector to call the user-defined demand calculation by submitting a job (not a SAS job) that will never be dispatched.

We don't want to use the default demand calculation program, because that program only considers the number of pending jobs and how long they have been pending. Remember, we want to scale up before we run out of resources, not after. Waiting until resources are exhausted before scaling up would mean a negative user experience, because that would lead to poor response time.

SAS worked with IBM to create a demand calculation program that can trigger requests for additional resources based on a CPU utilization threshold or a job slot utilization threshold. The testing we did during our research for this paper was all performed with a CPU utilization threshold, because SAS believes that CPU utilization, rather than job slot utilization, is more indicative of the true load on the system.

HOW DO WE DEFINE WHEN RESOURCE CONNECTOR WILL SHUT DOWN RESOURCES IT HAD PREVIOUSLY ADDED?

LSF resource connector has two mechanisms that can be used to shut down instances. One, specified by the option LSB_RC_EXTERNAL_HOST_IDLE_TIME, causes an instance to be shut down after an amount of time (specified by an administrator) has passed with no jobs running on the instance. The other, specified by the option LSB_RC_EXTERNAL_HOST_MAX_TTL, causes an instance to be shut down after the instance has been in the cluster for an administrator-specified amount of time.

The risk with using LSB_RC_EXTERNAL_HOST_MAX_TTL is that an instance will be shut down while it is running SAS workload. LSF requeues any jobs that are running on an instance when it is shut down. However, for the job to be restarted successfully, additional system requirements must be met, and they might have adverse performance implications. Current best practice is to not set LSB_RC_EXTERNAL_HOST_MAX_TTL, but to rely on LSB_RC_EXTERNAL_HOST_IDLE_TIME.

Figure 1 shows an example of what you might see in AWS after instances have been added to your grid and then terminated.

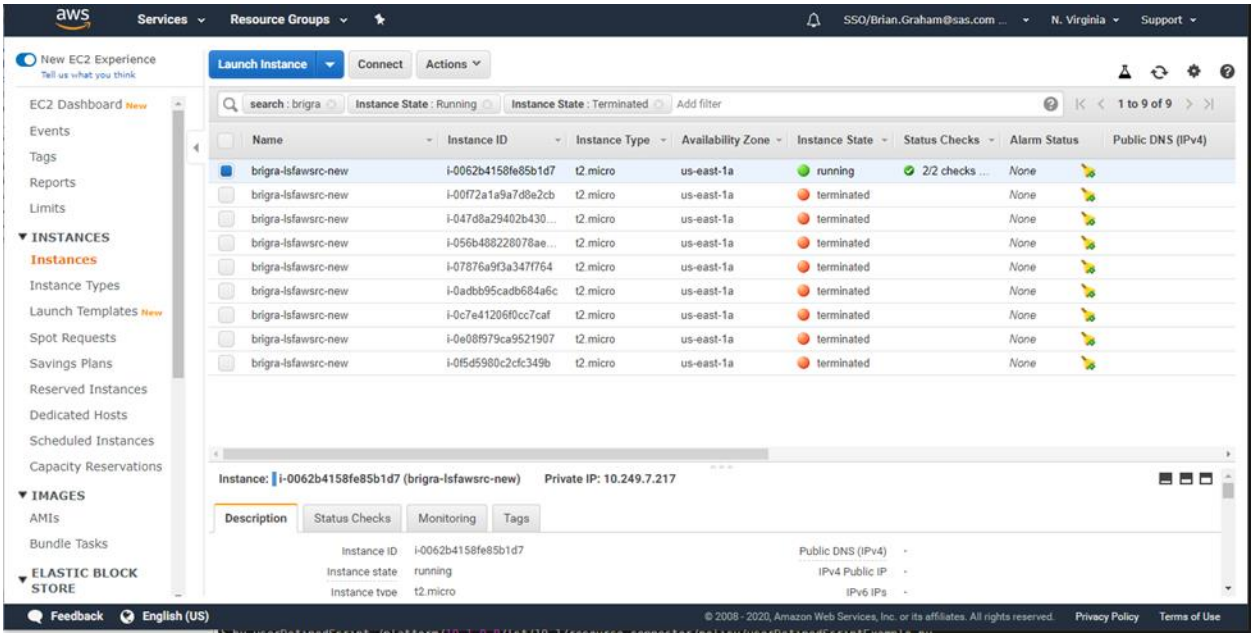

Figure 1: An Example of the View in AWS After Your SAS Grid Has Scaled Up and Down

HOW DO I CONTROL GROWTH?

There are a few ways we can control growth. One is to control the rate at which the system grows. That's expressed as the number of instances that can be launched in a given unit of time. For example, we might specify that one instance can be launched every five minutes. Or we might say five instances can be launched every 60 minutes. Another method is to specify the upper limit on the number instances that can be launched.

Both of those limits can protect us from the unwanted effects of someone making a mistake in their SAS program that generates very high, and possibly unlimited, demands on system resources.

HOW CAN WE MANAGE THE SAS SYSTEM CONFIGURATION WHEN RESOURCES ARE ADDED OR DELETED?

SAS Grid Manager for Platform configuration exists in SAS Metadata Server as well as in LSF. We covered LSF in the previous topic. Now we will focus on updating the SAS configuration when allocating and destroying AWS instances.

SAS has developed a SAS program that can take a machine name as an argument, add that machine name to the SAS Workspace Server configuration, and refresh the SAS Object Spawner. The refresh of the SAS Object Spawner is required so that the SAS Object Spawner is aware of the newly added instance and can use it to host the SAS Workspace Server. LSF resource connector is configured to run this SAS program after a new instance is started. Figures 2 and 3 show what SAS Workspace Server configuration might look like before and after an AWS instance is added to SAS Workspace Server configuration.

Similarly, SAS has developed a SAS program that can take a machine name as an argument, remove that machine name from the SAS Workspace Server configuration, and refresh the SAS Object Spawner. The refresh of the SAS Object Spawner is required so that the SAS Object Spawner won't try to use the deleted instance to host the SAS Workspace Server. LSF resource connector is configured to run this SAS program after an instance is shut down.

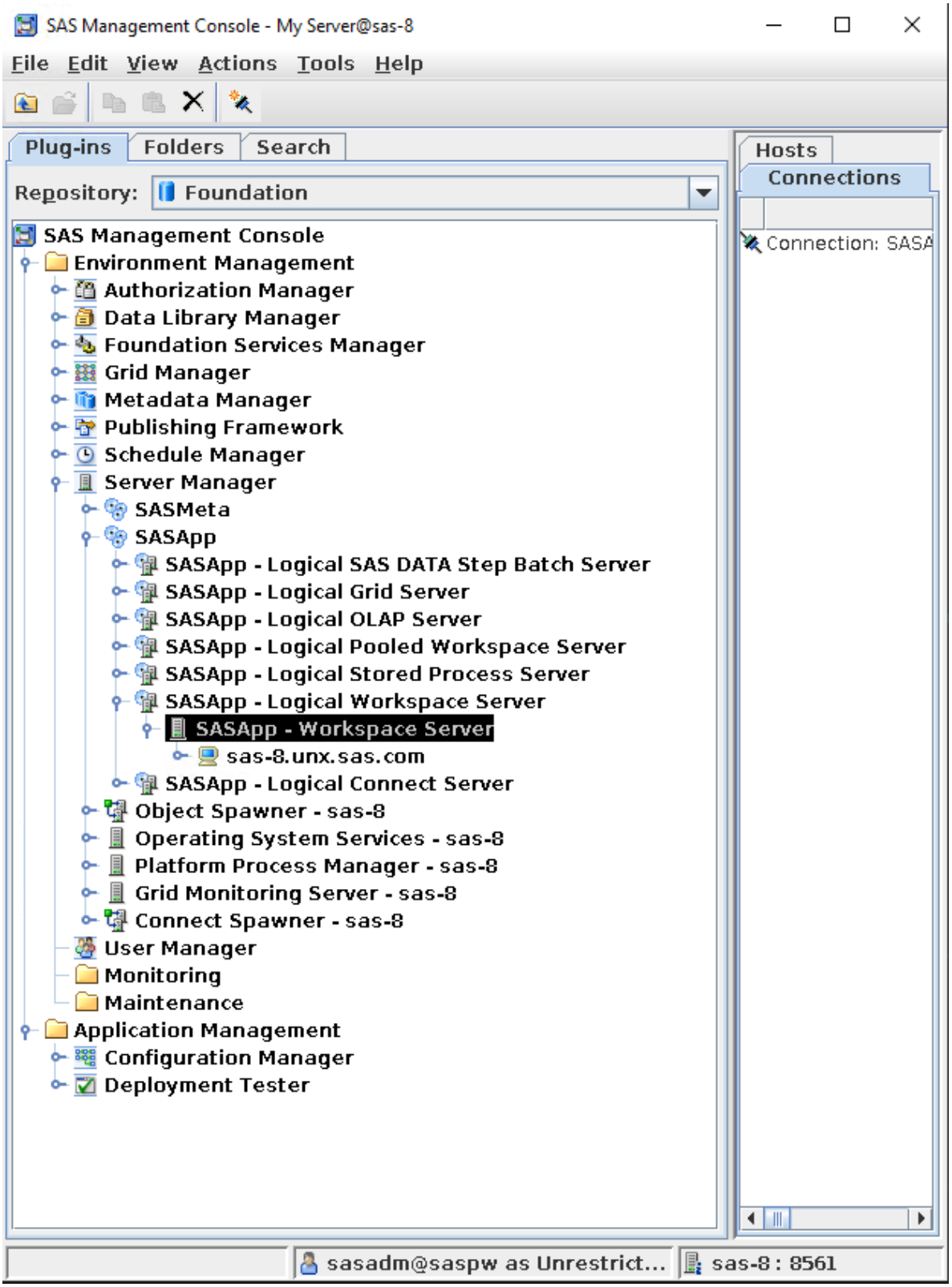

Figure 2: SAS Management Console Before an AWS Instance is Added

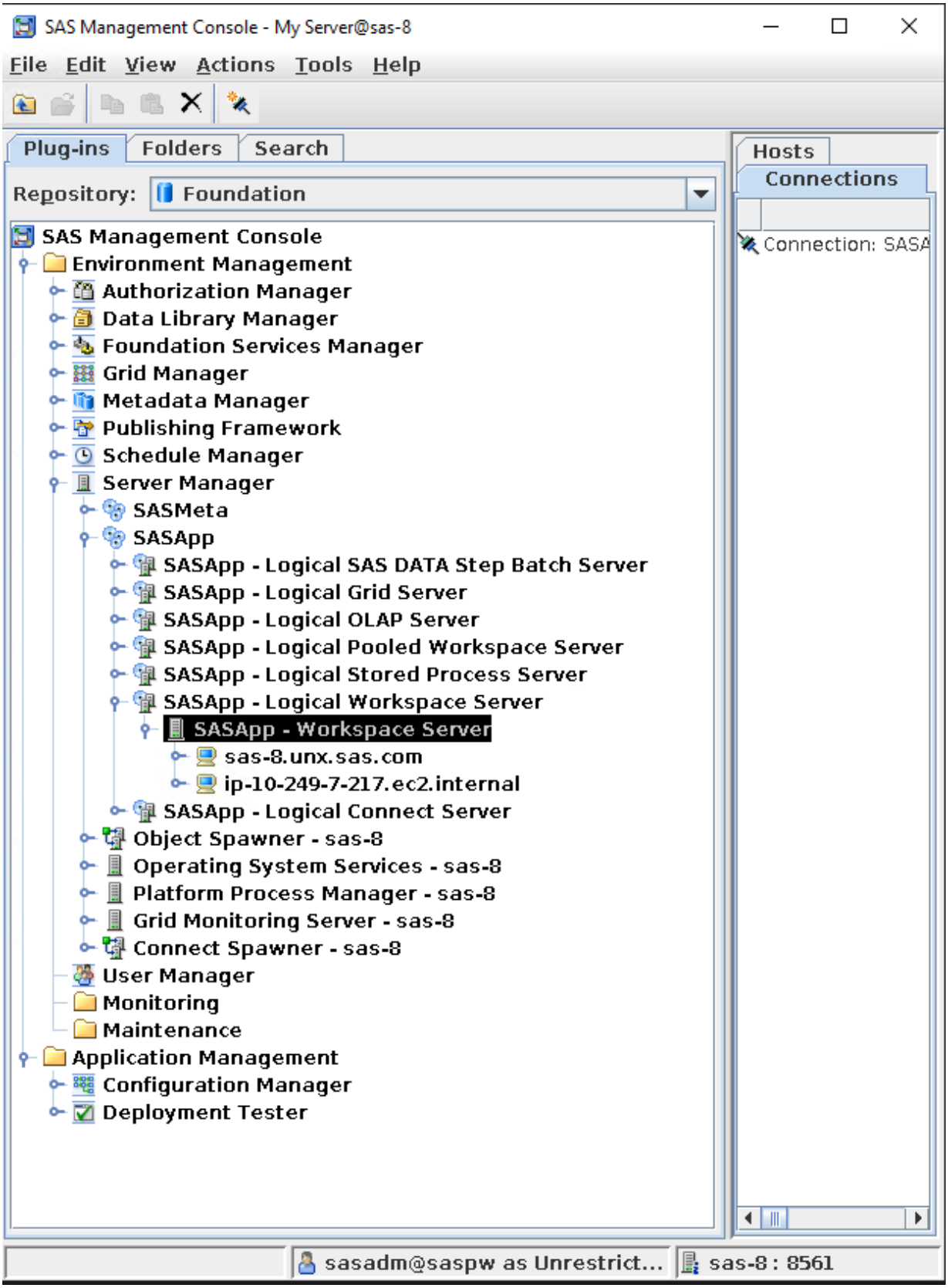

Figure 3: SAS Management Console After an AWS Instance is Added

[DB1]DOES THE SYSTEM HAVE TO BEHAVE THE SAME WAY FOR ALL USERS AND WORKLOAD?

It may be that you want the system to behave differently depending on the workload that is submitted. We can define some LSF queues to create demand for extra instances while other queues do not create demand. Batch or high priority workload can create demand for additional resources while other workload does not, for example.

WHAT STEPS ARE REQUIRED TO ENABLE AUTOMATED SCALING?

BUILD AN AMI LSF CAN USE TO INSTANTIATE INSTANCES

You have to specify the AWS attributes LSF resource connector will use when it asks AWS to create an instance. Some of those attributes are the identifier of the AWS AMI, the subnet, and the instance type. The instance used to create the AMI must be prepared to run SAS. That means the instance must be configured as follows:

- Configure network and DNS to allow communication with LSF cluster
- Authenticate users in a fashion that is compatible with the LSF master machine
- Meet all requirements in the SAS Pre-Install Requirements Document
- Mount all required file systems
- Install LSF as a slave host

CONFIGURE LSF RESOURCE CONNECTOR TO LEVERAGE AWS

LSF resource connector has to know several things to enable it to make requests to AWS. These are some of the questions you must answer:

- What are the conditions that must be met before instances are added?
- How is authentication to AWS performed?
- What AMI, instance type, and subnet should be used when instantiating instances?
- When are instances terminated?
- What is the maximum instance growth rate and upper limit on instances?

CONCLUSION

If you have, or will have, a SAS Grid Manager for Platform environment in AWS and you would like to contain costs by having the system grow and shrink based on demand, that's now possible. This might mean your existing environment can shrink to a size smaller than it is now. This might mean your existing environment can grow to a size larger than it is now. This also might mean both. Enabling this technology requires you to have AWS skills that enable you to perform tasks such as building AMIs and setting up authentication.

ACKNOWLEDGMENTS

I would like to thank the following people for their assistance in the development and testing of the system configuration on which this paper is based.

Kinson Chik, Advisory Software Developer, IBM

Qingda Wang, Senior Development Manager, IBM

Brian Graham, Senior Technical Architect, SAS Institute

Eric Davis, Technical Consultant, SAS Institute

Paula Kavanagh, Senior Software Developer, SAS Institute

RECOMMENDED READING

- *Using the IBM Spectrum LSF Resource Connector*

CONTACT INFORMATION

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

Glenn Horton
SAS Institute
Glenn.Horton@sas.com