Paper 391-2008

Balancing the Load - SAS® Server Technologies for Scalability

Cheryl Doninger, SAS Institute Inc., Cary, NC
Jason Spruill, SAS Institute Inc., Cary, NC

ABSTRACT

This paper will address a variety of SAS® servers and how they can be used to balance workload and work together to provide scalability in a SAS Enterprise deployment. We will discuss a variety of servers including stored process servers, workspace servers, data step batch servers, and grid servers. We also will discuss the options for using these servers to balance load and provide solutions that can leverage a scale-out architecture.

INTRODUCTION

There are many different ways in which you can interact with SAS applications. Traditional SAS users are familiar with SAS Display Manager Session (DMS) that provides a user interface for interactive job submission. In addition, a command line interface allows for the submission of SAS batch jobs. However, these are not the only methods of accessing SAS. SAS® Web Report Studio enables you to access SAS via a Web-based interface. Applications such as SAS® Management Console and SAS® Data Integration Studio provide a portable Java client for access to SAS applications. For Windows users, SAS® Enterprise Guide® provides an interface to SAS based on the popular .NET framework. Additionally, the SAS® Add-In for Microsoft Office provides an interface to SAS tools directly in a third-party product.

Regardless of the interface or interfaces you use to run your SAS applications, a SAS program is eventually executed by some type of back-end SAS server. Because of the need for many different interfaces to SAS and the wide range of SAS products and solutions, multiple SAS server technologies have been developed over the years to play different roles and serve different needs. In the context of this paper, the term *server* refers to a SAS process. There might be one or more SAS servers per machine. The term *load balancing* refers to spreading SAS workload across multiple SAS servers and also multiple machines for the purpose of increasing performance, resource utilization, or response time, for example.

The purpose of this paper is to define the evolution of SAS server technologies that play a role in scalability and how they can be load balanced to meet the needs of multiple users and increased workload in an enterprise SAS deployment. Initially, we will look at SAS/CONNECT®, the benefit of running multiple SAS/CONNECT servers, and the opportunity to programmatically implement load balancing. Next, we will focus on the multi-user capabilities of a SAS grid environment and how SAS® Grid Manager can provide the infrastructure to balance SAS workload across various SAS servers in a grid using third-party algorithms. Next, we will focus on the stored process and workspace servers that are provided with SAS® Integration Technologies, how these servers are used by SAS applications, and details about the internal load balancing algorithms that are used by these servers. Finally, we will discuss a new SAS 9.2 feature that enables the workspace server to take advantage of the load balancing that is provided by SAS Grid Manager.

SAS applications, solutions and users generate many SAS work requests. The value of having these requests managed by SAS Grid Manager is the ability to create a shared SAS grid environment that is managed and load balanced to meet the varied needs of all your SAS users across your enterprise. You also have an enterprise infrastructure that has the flexibility to grow as your organization grows.

SAS/CONNECT

SAS/CONNECT is a long-standing SAS product built on a client/server architecture. SAS/CONNECT was originally developed primarily to enable SAS processes to communicate with each other for the purpose of sharing data and processing. The client and server can reside on the same physical machine or on separate distributed machines on a network. In SAS 8, parallel capabilities were added to SAS/CONNECT to enable spawning *n* SAS/CONNECT server sessions to simultaneously execute *n* tasks as independent processes and coordinate the execution and results of all *n* tasks in the client or parent SAS session. The SAS/CONNECT server sessions or processes can all execute on the same SMP machine with each session or process running on a separate processor. SAS/CONNECT also has the flexibility to spawn multiple SAS sessions to run on any number of remote machines across a network. The remote

machines can have either single processor or multiprocessor capabilities. The interface to the parallel capabilities of SAS/CONNECT is syntactic and therefore requires a SAS application programmer to develop an application to leverage these capabilities.

The parallel capabilities of SAS/CONNECT enable an application developer to divide long-running SAS applications into sub-tasks, and use any number of SAS/CONNECT server sessions to execute the sub-tasks in parallel and substantially speed up the execution of the entire application. The possibility of this type of performance gain applies to applications that include either multiple independent units of work or replicate runs of the same fundamental task. By dividing time-consuming tasks into multiple units of work and executing these units of work in parallel, a job can be performed in substantially less time than if each task is performed sequentially.

SAS/CONNECT LOAD BALANCING

Just as the parallel processing capabilities of SAS/CONNECT are surfaced through the SAS 4GL syntax, so are the load balancing capabilities of the multiple SAS/CONNECT server sessions. It is the responsibility of the application developer to incorporate load balancing logic into an application that is developed with SAS/CONNECT. SAS/CONNECT does provide statement syntax and macro variables to facilitate development of load balancing. In addition, SAS has written a macro called %DISTRIBUTE which provides a template for deploying a SAS program in a distributed environment by using the parallel capabilities of SAS/CONNECT. The %DISTRIBUTE macro also provides load balancing of the multiple SAS/CONNECT server sessions. The %DISTRIBUTE macro and a paper describing its use can be downloaded from the papers section of the Scalability and Performance Community at http://support.sas.com/rnd/scalability/papers/index.html under the heading "**The %Distribute System for Large-Scale Parallel Computation in the SAS System.**"

SAS GRID MANAGER

SAS Grid Manager was introduced in SAS 9.1.3; it builds on the parallel capabilities of SAS/CONNECT and adds many other capabilities that are required by enterprise grid deployments. SAS Grid Manager provides load balancing, policy enforcement, efficient resource allocation, and prioritization for SAS products and solutions that run in a shared grid environment. In addition, it de-couples the SAS applications and the infrastructure that is used to execute the applications, which enables hardware resources to transparently grow or contract as needed, and provides tolerance of hardware failures within the grid infrastructure. SAS Grid Manager integrates the resource management and scheduling capabilities of the Platform Suite for SAS with the SAS 4GL syntax and subsequently with several SAS products and solutions. This integration enables load balancing of SAS workload in a variety of different ways, including the following:

multiple users
parallel workloads
enterprise scheduling

MULTIPLE USERS

Many customers have a number of ad hoc SAS users who develop models and do queries and other sorts of ad hoc development, discovery, and analysis. SAS Grid Manager provides management of this ad hoc environment from the perspective of controlling which jobs get priority over others, which jobs get which share of the computing resources, and prevention of mass submission of work requests, which results in frequent server crashes. SAS Grid Manager will map the SAS work requests to the appropriate resources, and if necessary, run just a subset of the work and queue the remaining work requests for execution as soon as resources become available. High-priority jobs can even preempt lower-priority work so that the most critical business processes execute first. SAS Grid Manager provides the structure that is needed to create an organized and managed SAS analytic environment to ensure that the appropriate resources are allocated to the appropriate users and workload to meet the objectives of the organization.

The total SAS user community is likely using many different interfaces for running SAS applications. Some portion of users might be running SAS in interactive DMS mode. In this case, jobs can be interactively submitted to a grid environment by defining a new key sequence to submit the appropriate statements to send the job to the grid rather than executing on the local workstation. Similarly, for those users who prefer batch job submissions, a wrapper script file can be used to submit batch jobs to the grid with no change to the application and no change in the way your users interact with SAS. Additionally, SAS Enterprise Guide users can submit their SAS programs and Enterprise Guide tasks to a SAS grid for execution by inserting the appropriate SAS syntax in the Insert custom SAS code dialog boxes within SAS Enterprise Guide. SAS Grid Manager provides the infrastructure to balance all SAS applications

and workload across a shared grid infrastructure. Detailed instructions for the key definitions, grid batch wrapper scripts, and steps for grid-enabling SAS Enterprise Guide can be found at http://support.sas.com/rnd/scalability/grid/download.html and http://support.sas.com/rnd/scalability/grid/EG_on_grid.pdf, respectively.

PARALLEL WORKLOADS

A subset of SAS applications consists of subtasks that are independent units of work and can be distributed across a grid and executed in parallel. The benefit of distributed parallel execution is potentially substantial acceleration of the entire application. A common workflow in applications that are created by SAS Data Integration Studio is to repeatedly execute the same analysis against different subsets of data. For this workflow, the Loop and Loop-End transformation nodes can be used in SAS Data Integration Studio to automatically generate a SAS application that spawns each iteration of the loop to a SAS grid via SAS Grid Manager. SAS® Risk Dimensions® has a similar iterative workflow that executes the same analysis over different subsets of data. The data is subset based on market states or by instruments. Each iteration of the analysis can be submitted to the grid using SAS Grid Manager to provide load balancing and efficient resource allocation. In contrast, the workflow for SAS® Enterprise Miner™ during the model training phase is to execute a series of different models against a common set of data. Because the models are independent of each other, the SAS Enterprise Miner engine that generates the SAS program to execute the user-created flow will automatically insert the necessary syntax to spawn each model execution to the grid for parallel and ultimately accelerated execution. A programmer might modify an existing application or develop a new application that consists of replicate runs of the same fundamental task or multiple distinct independent units of work. The programmer can use the grid syntax in the SAS 4GL programming language to create a grid-enabled application. In all of these scenarios, SAS Grid Manager distributes and load balances the parallel work requests to a shared pool of SAS grid resources.

ENTERPRISE SCHEDULING

Scheduling production jobs is an important and necessary function in every enterprise. SAS provides the Schedule Manager plug-in as part of SAS Management Console to enable you to create SAS workflows and schedule them based on time and file events. SAS Grid Manager distributes multiple workflows or the jobs within a single workflow to a SAS grid environment for load balancing and sharing of resources. The SAS jobs can be created by a variety of SAS applications and solutions or written by SAS programmers. Various levels of scheduling capabilities have been incorporated directly into many SAS products and solutions, including SAS Data Integration Studio, SAS® Marketing Automation, SAS® Marketing Optimization, and SAS Web Report Studio. The jobs that are created by these products as well as any other SAS products, including user-written SAS programs, can be used to create simple or very complex workflows and scheduled to a SAS grid environment.

SAS GRID MANAGER COMPONENTS

Several types of machines and software components make up a SAS grid environment. These machines have been defined to clarify the role each plays, the software components that must be installed on each, and the SAS metadata that must be configured. Diagram 1 illustrates the machines and components that make up a SAS grid architecture. The SAS Metadata Server is shown on a separate machine in this sample architecture. It is often common to dedicate a machine to running the SAS Metadata Server. However, you can choose to run the SAS Metadata Server on the grid control machine.

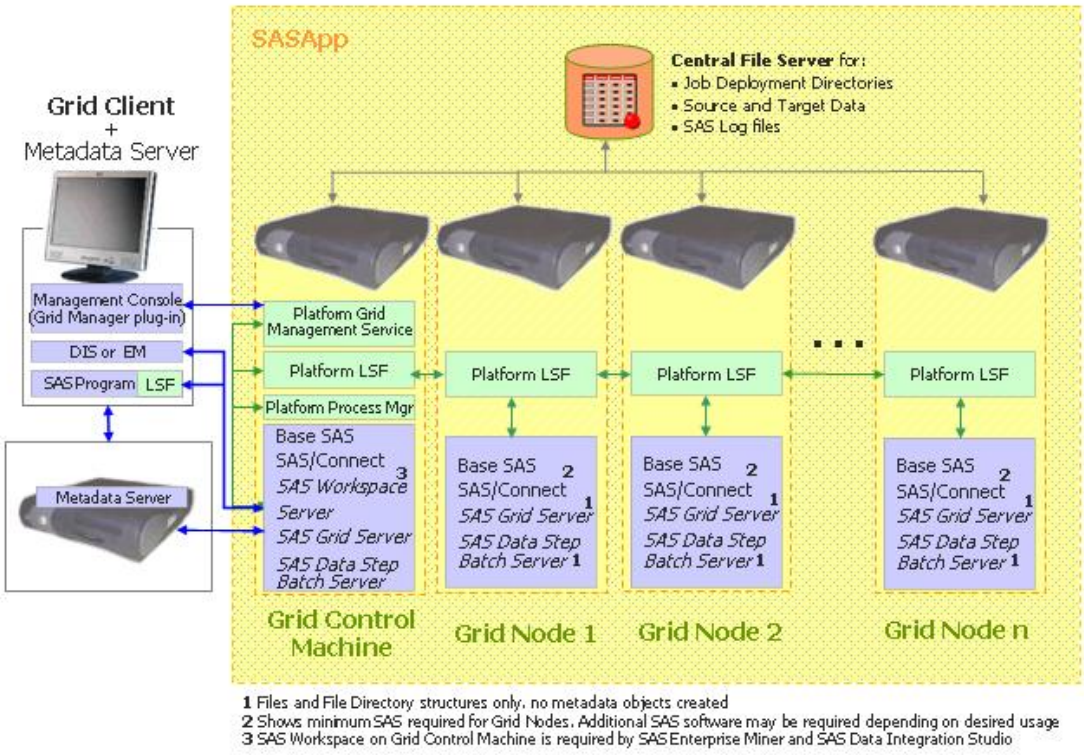

Diagram 1: SAS Grid Architecture

The three machines that are specific to a grid installation are defined as follows:

Grid Client – a grid client submits work to the grid but is not part of the grid resources that are available to execute work. Here are examples of a grid client:

a SAS Data Integration Studio client (Platform LSF is not installed on this client machine).

a SAS Enterprise Miner client (Platform LSF is not installed on this client machine).

a SAS Management Console client using the Schedule Manager plug-in or any other applications that schedule SAS workflows (Platform LSF is not installed on this client machine).

a SAS® Foundation installation (minimum Base SAS®, SAS/CONNECT, and Platform LSF) used to run a program that submits work, both whole programs or programs broken into parallel chunks, to the grid. Installation of the Platform LSF component is required in this case in order for SAS/CONNECT to submit the work to the grid.

Grid Control Machine – any machine in the grid can be designated as the grid control machine. More software is installed on the grid control machine and more SAS metadata configuration takes place on this machine. In a SAS Data Integration Studio and SAS Enterprise Miner scenario, the grid control machine runs a workspace server that executes programs that use SAS Grid Manager to distribute work to the grid nodes. The grid control machine can be configured as a grid resource that is capable of receiving work to execute or not, depending on the needs of your environment.

Grid Node – a grid node is a grid-computing resource that is capable of receiving the work that is being distributed. Each grid node must be running a minimum of Base SAS, SAS/CONNECT, and Platform LSF.

The following SAS metadata definitions are created for the different servers that can be used in a grid:

SAS Grid Server – a SAS metadata definition that provides the information that is needed by a grid-enabled SAS application to execute on the grid. A grid-enabled application is an application that uses the GRDSVC_xxxx grid function calls. One of the required pieces of information in the SAS Grid Server definition is a script file or command line to be used to invoke a SAS session on a grid node.

SAS Batch Server – a SAS metadata definition that provides the command line that is used to execute a scheduled SAS job when it has been triggered to run. There are two types of batch servers:

SAS Data Step Batch Server – a command line for executing SAS programs that are written in the SAS 4GL syntax

SAS Java Batch Server – a command line for executing Java programs that are created by SAS solutions and products

SAS Workspace Server – a metadata definition that provides the information that is necessary for the SAS Object Spawner to invoke a workspace server, including port, command line, and so on. A SAS Workspace Server is not a requirement of SAS Grid Manager but is needed if you will be running any SAS products in the grid that require a workspace server such as SAS Data Integration Studio and SAS Enterprise Miner.

SAS GRID MANAGER LOAD BALANCING

SAS Grid Manager provides an integration of the SAS application environment and the grid middleware available from Platform Computing known as the Platform Suite for SAS. The Platform Suite for SAS has three components:

Process Manager (previously called Platform Job Scheduler) – provides the interface that is used by the SAS scheduling capability. This is the interface that is used by the SAS scheduling framework to control the submission of scheduled jobs to LSF and manage any dependencies between the jobs.

Grid Management Services – communicates with the Grid Manager plug-in to provide run-time monitoring and management capability within SAS Management Console for the jobs, hosts, and queues in the grid environment.

LSF – (Load Sharing Facility) dispatches all jobs that are submitted to it, either by Process Manager or directly by SAS, and returns the status of each job. LSF also manages any resource requirements and performs load balancing across machines in a grid environment.

As the name implies, the Load Sharing Facility (LSF) is the component that is responsible for performing the load balancing and resource allocation of all work requests that are being submitted to the grid. LSF does not execute jobs immediately, but it puts the jobs in a specified queue or in the default queue if no queue is specified. Multiple queues can be defined with different priorities, resource allocations, time policies, and preemptive capabilities that are associated with each queue in order to meet the needs of your organization. LSF matches the requirements of the job (if defined) with the resources of the host as defined by the administrator, and decides when jobs should be picked up from the queue and executed on a host. Several factors affect this decision, such as the load on the host and the requirements of the job.

LSF waits for a configured number of dispatching intervals before sending another job to the same host. The waiting time is configured by the parameter **JOB_ACCEPT_INTERVAL** in the lsb.params file (as defined in **$LSB_CONFDIR**/<cluster_name>/configdir1)[1]. The amount of time between dispatching intervals is specified by the parameter **MBD_SLEEP_TIME**, which is defined in the lsb.params file. In a default installation of Platform LSF, the **JOB_ACCEPT_INTERVAL** is 1 and the **MBD_SLEEP_TIME** is 20 (seconds). Therefore, Platform LSF dispatches one job to a particular machine and waits 20 seconds before dispatching another job to the same machine.

Typically, the default configuration is sufficient for most workload types. However, you can tune Platform LSF to optimize your machine configurations and workload types to obtain better overall performance. Complete details about the options that are available to tune the load balancing that is provided by SAS Grid Manager and the Platform Suite for SAS can be found in the paper “SAS Goes Grid – Managing the Workload Across Your Enterprise” at http://www2.sas.com/proceedings/sugi31/211-31.pdf and in the Platform Suite for SAS documentation available on

[1] **LSB_CONFDIR** is the directory that contains all batch configuration files (such as details about the hosts, queues, and batch parameters in your cluster). Typically, this directory is located in ***<LSF_TOP>*/conf/lsbatch**, where ***<LSF_TOP>*** is the location of your Platform LSF installation.

the Scalability and Performance Web site at http://support.sas.com/rnd/scalability/platform/index.html.

SAS® INTEGRATION TECHNOLOGIES

SAS Integration Technologies provides the foundation to enable you to make information delivery and decision support a part of your enterprise's information technology architecture. Integration Technologies provides you with the software to build a secure client/server infrastructure to connect clients (for example, Java and Web-based clients) to SAS servers to implement distributed processing solutions using SAS. Applications such as SAS Enterprise Guide and SAS Data Integration Studio use SAS Integration Technologies to build stored procedures and workflows to control the flow of data throughout your enterprise. SAS Web Report Studio uses the components that are provided by SAS Integration Technologies to produce and distribute reports from this data. SAS® BI Web Services uses SAS Integration Technologies to provide a standard Web service interface to access analytical functions that are provided by SAS.

With the many consumers of the components that are provided by SAS Integration Technologies, scalability becomes an important characteristic of the architecture. SAS Integration Technologies provides its own load balancing capabilities to manage the loads that are incurred by these consumers. SAS Integration Technologies load balancing enables you to scale out by distributing the workload that is generated by your users. Scaling out becomes as simple as adding new machines to handle your growing workload. In addition, no changes are required to your applications in order to take advantage of your additional processing capabilities.

SAS INTEGRATION TECHNOLOGIES LOAD BALANCING

The Integrated Object Model (IOM) is the underlying technology that implements the components that are described in this section of the paper. IOM load balancing is a feature developed by SAS that enables the distribution of client workloads across servers. The following sections document the specific IOM components and algorithms that provide load balancing of the IOM servers.

Load balancing manages the handling of these workloads by monitoring the load of each server that is a member of the logical server. A logical server is a homogenous grouping of server components. The term *cluster* refers to logical servers that are used in conjunction with load balancing. A load balancing algorithm assigned to each cluster determines which server in the cluster will be used. The following IOM components work together to provide load balancing capabilities to clients:

SAS Object Spawner – The SAS Object Spawner is a stand-alone application that handles client requests for the SAS Workspace Server and SAS Stored Process Server. The SAS Object Spawner typically runs as a system service or daemon. It listens for new client connection requests on behalf of the SAS Workspace Server and SAS Stored Process Server. This application launches new server processes of either type as needed. Load balancing runs as part of the SAS Object Spawner. When the SAS Workspace Server or SAS Stored Process Server is set up with load balancing activated, the SAS Object Spawner passes the client connection request to the load balancer in order for it to determine the server in the cluster that should receive the client. If load balancing determines that a new server should handle the client connection request, the SAS Object Spawner will launch the new server for that client to use.

SAS Workspace Server – The SAS Workspace Server is a server that provides access to Foundation SAS software. This includes access to SAS language services, SAS libraries, the underlying server file system, and formatting services. SAS Workspace Servers are single-use servers, meaning that for each client that connects, a new server starts. When the client disconnects, the server shuts down. Load balancing monitors the number of SAS Workspace Server processes that are running at any given time, and determines the load of the server's system by monitoring this information. The SAS Workspace Server does not have load balancing activated by default, but an administrator through SAS Management Console can easily activate the feature. Without load balancing activated, only the operating system, through memory or process limits, controls the number of SAS Workspace Servers started.

SAS Stored Process Server – The SAS Stored Process Server that is provided by Integration Technologies runs SAS programs that are stored on a server called directly by client applications. The SAS Stored Process Server delivers the results of the SAS program to the client application. The SAS Stored Process Server is a multi-user server, which means that a single running instance of the server allows multiple clients to connect, either concurrently or consecutively. Unlike the SAS Workspace Server that shuts down after its client disconnects, the SAS Stored Process Server continues to run and handle other

clients. Load balancing determines the load for SAS Stored Process Servers by monitoring the number of clients that are connected to each running instance of the server. In SAS 9.2, by default, the SAS Stored Process Server always uses load balancing.

Administrators can configure clusters in different ways, depending on the needs of the organization. A cluster can contain a single server definition that runs on a single host. Distributed clusters might include multiple servers that are defined to run on the same machine, or might involve several machines to balance client workloads.

COMMUNICATION BETWEEN LOAD BALANCING COMPONENTS

Load balancing manages the load on clustered servers by redirecting each client connection request that is received by the cluster. Each server in the cluster can potentially receive a connection request from a client. The SAS Object Spawner listens for client connection requests on behalf of the SAS Workspace Server and SAS Stored Process Server. For the purposes of load balancing, the term *peer* refers to the IOM components that accept the client connection requests. Therefore, the SAS Object Spawner is a load balancing peer.

In order for a distributed cluster to maintain current run-time information, there must be constant communication between the members of the cluster. Peers are the backbone of communication in a distributed cluster. Therefore, each machine in a distributed cluster must have a peer defined for it. Each machine in a SAS Workspace Server cluster or SAS Stored Process Server cluster must have a SAS Object Spawner definition. In any configuration, peers are the components that communicate the load information that is used to determine the destination to redirect client requests. In a distributed cluster, peers classify either as the parent peer or as a child peer. The parent peer of a cluster acts as the head of the cluster. At any given time, there is a single parent peer in a distributed cluster. All other peers are child peers and take their direction from the parent peer. Any peer in a distributed cluster can accept the client request. However, only the parent peer can make the decision on where to redirect the request. Child peers that receive client requests must contact the parent peer before redirecting the request.

Peers are determined to be either parent peers or child peers when they initialize. During initialization, each peer processes its configuration and determines the other peers in the cluster, generating a list of peers. Once the configuration processing is complete, each peer attempts to establish a connection to the other peers in the cluster. The goal of these connection attempts is for the peer to connect to the current parent peer for the cluster. If a peer is the first peer of a cluster to initialize, the peer becomes the parent peer for the cluster. If the peer connects to another child peer in the cluster, the child peer will direct the peer to the parent. Once connected to the parent, the child peer begins sending messages to initialize communication between the two peers. Diagram 2 illustrates the process by which a cluster of load balancing peers initialize communication with each other.

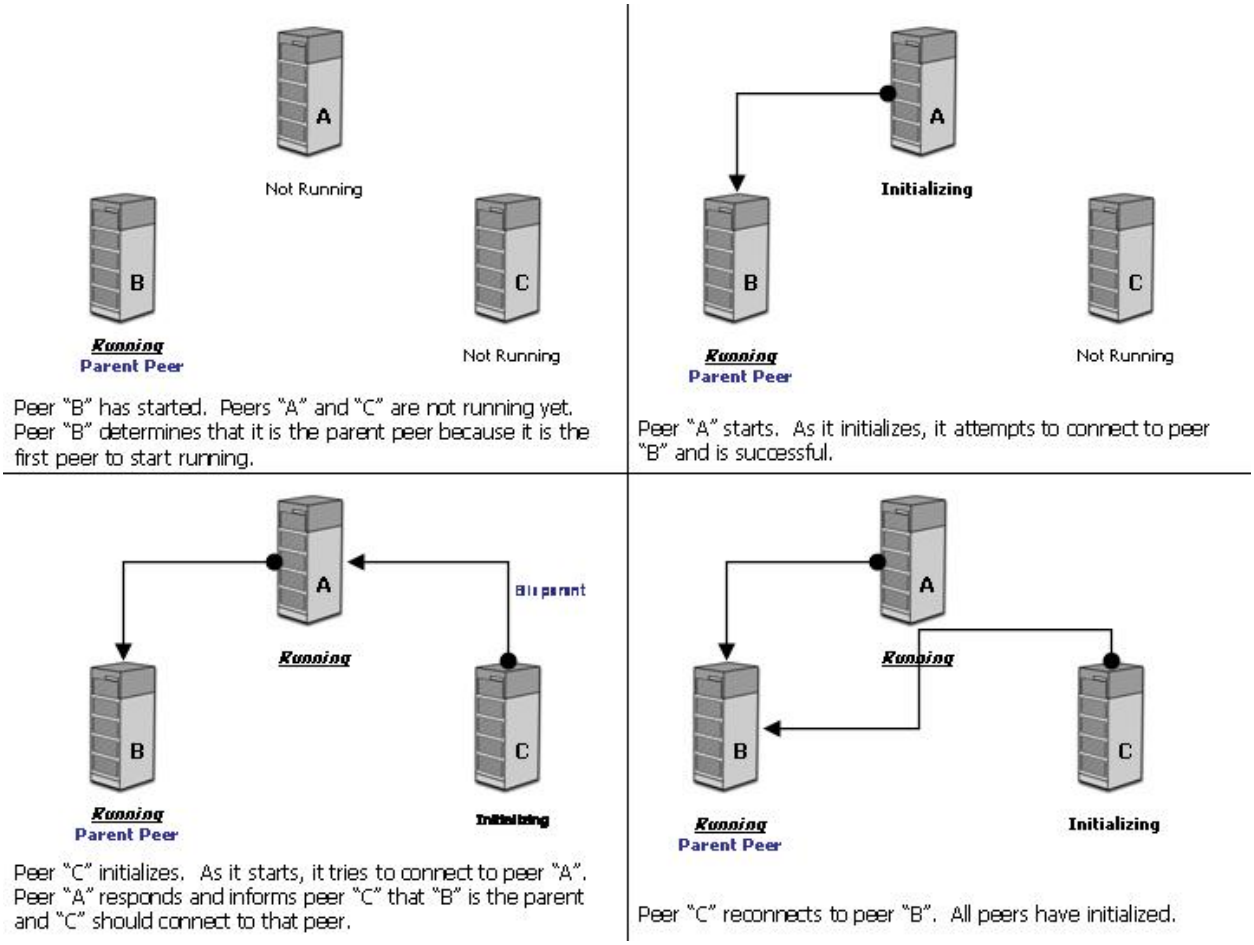

Diagram 2: Process by Which a Cluster of Load Balancing Peers Initialize Communication with Each Other

It is not required that all peers are always running in a distributed cluster. A subset of the machines that are configured for use can be used. It is also not necessary for all peers in a distributed cluster to remain running. Any peer in the cluster, including the parent, can be shut down at any time (either intentionally or unintentionally, as in the case of a hardware failure), and the cluster will continue to manage the workload with the remaining peers. If the parent peer does happen to be the peer that shuts down, the remaining peers determine which peer is the next parent. Each remaining peer then reconnects to this parent to continue load management.

A client that connects to a cluster might be redirected to any of the servers in that cluster. Furthermore, the client cannot select the server in the cluster that will handle its request. Therefore, it is important that every server in the cluster have access to the same data, so that a client that connects to the cluster can access the data no matter which server handles its request. For this same reason, it is also important that the servers run in the same authentication domain. The same set of user credentials that a client request might send to the server must be valid on any server in the cluster. If not, the server will not be able to authenticate the client request, and the request will fail. Client applications can see multiple servers defined in metadata and they will attempt to connect to them, one at a time, until they find one that works, or they exhaust the list.

BALANCING CLIENT REQUESTS

Once a peer has initialized, it can begin accepting client requests. When a peer accepts a client request, it passes the request into the load-balancing processor. From the client connection request (which includes a port and server name), the load balancing processor is able to determine the cluster from which the client is requesting a server. The processor uses the information it obtained from the metadata about the cluster to generate a list of available servers in the cluster for the client redirection. The following factors affect the availability of a server destination:

Server Status – The status of each server is checked. Servers that are shutting down, or preparing to shut down, should not receive new client requests. For this reason, the available list does not include such

servers. In addition, if load balancing detects an incorrectly configured server or a server that is reporting failures when being launched, the load balancing processor excludes that server from the available list.

Server Accessibility – Not every peer in a distributed cluster might be running and initialized when the load balancing processor receives the client request. Load balancing redirects the client request only to peers that are currently connected to the cluster. Therefore, servers that are associated with these disconnected peers are not included in the available list.

Security Rules – Client requests should redirect only to a server destination they have permission to use. For this reason, the load-balancing processor ensures that the list of available server destinations contains only servers that the client has proper authorization for.

With the available server list created, the load balancing processor sends the list to the load balancing algorithm chosen for the cluster. The algorithm is responsible for applying its own rules to the available list and determining which server from the list should handle the client request. The selected server information returns to the processor, which in turn sends the information back to the peer that initially received the request. The peer then converts this information to instructions that enable the client to reconnect to the appropriate server in the cluster. Diagram 3 describes the flow of a client connection request as it goes through load balancing.

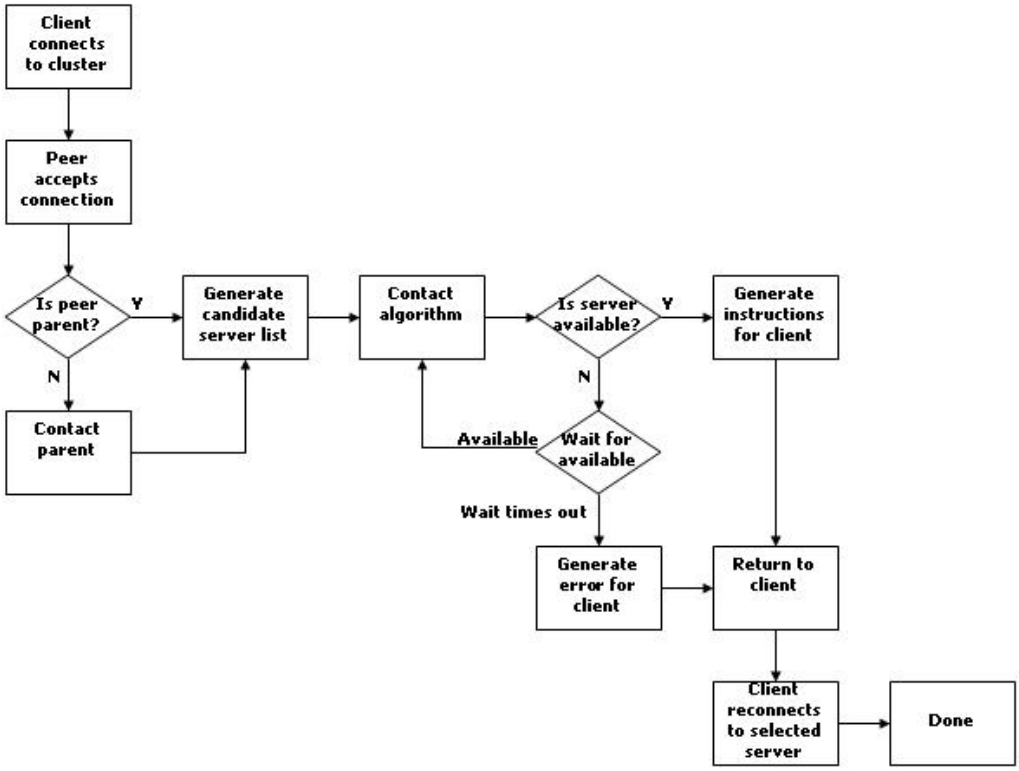

Diagram 3: The Flow of a Client Connection Request as It Goes through Load Balancing

Each cluster uses a single load balancing algorithm that independently determines the current load on each server in the cluster. These algorithms distribute the client connection requests by selecting the member server that should receive the next request. In addition to distributing the client requests, each algorithm might set limits on the number of clients that each server can accept. Therefore, there might be occasions where the algorithm determines that all available servers are at their maximum capacity. If this occurs, the algorithm will wait for an amount of time for a server to become available. Each cluster contains a property called *availability timeout* that controls the length of time

the algorithm will wait for available servers. If this amount of time passes before a server becomes available in the cluster, the client request will fail.

BALANCING ALGORITHMS

The load balancing algorithms determine the servers in a cluster that are available to process client requests. Each algorithm controls this distribution by monitoring and calculating the load on each server in the cluster. The algorithm receives updates in real time related to clients that are connecting and disconnecting, the current state of servers being managed, as well as other pieces of information that enable the algorithm to have up-to-date calculations when attempting to determine the right server to receive a client request. Each algorithm might have its own set of properties that affect its behavior. These properties are configurable through SAS Management Console. The following are some of the different algorithms that are used by load balancing:

Cost algorithm – This algorithm calculates a weighted load value for each server in the cluster. This value is the current cost of the server, and is mostly determined from the number of clients that are currently connected to the server, but might include factors that are controlled by the server itself. These server factors might change based on the client usage of the server. Each server definition in the cluster contains a property called maximum cost, which denotes the maximum cost value that is attainable by that server. A server that has reached its maximum cost will not be available to receive client requests until the cost value decreases. The Cost algorithm selects the available server with the lowest current cost to process the client request. This is the default algorithm that is used by the workspace server.

Response Time algorithm – This algorithm can be used with the SAS Stored Process Server and monitors the response time of the servers in the cluster. The goal of this algorithm is to send client requests to the server with the best response time. Servers that use this algorithm specify the maximum number of clients that can connect to a server at any given time.

Grid algorithm – The SAS Workspace Server can use the Grid algorithm. This algorithm balances client requests by connecting to SAS Grid Manager and requesting load information. The integration of load balancing and grid services is new for SAS 9.2 and is discussed further in the last section of this paper.

ENSURING SECURITY

As discussed earlier, the load balancing processor considers security rules when generating the list of available servers that can process a client request. The processor makes these security decisions based on information in the SAS Metadata Repository, which is configurable through SAS Management Console. When the load balancing processor receives a client request, the processor contacts the SAS Metadata Server to determine if the client has authorization to access any of the servers in the requested cluster. If there are any servers in the cluster that the client does not have authorization to use, the load balancing processor removes these servers from the client's available servers list. If the SAS Metadata Repository determines that the client does not have authorization to use any of the servers in the cluster, the client receives an error. This prevents the client from using a server for which it does not have proper authorization.

The enforcement of these security rules can also allow for server prioritization. Suppose an administrator has multiple groups of users that are going to be accessing a system of SAS servers. Each user group represents a different level of user in the enterprise. An administrator might determine that one of the groups, such as executives, should have access to a greater number of servers than say, a different group. The administrator can use these security rules to enable the higher-priority group access to more servers than the lower-priority group. Load balancing will be able to determine, based on the client's identity, what priority the users have, and will generate an available server list accordingly.

ADMINISTERING IOM LOAD BALANCING

Administrators can use SAS Management Console to administer servers in a load balancing cluster. Administrators with the proper authorization can connect to each of the components listed earlier. Once connected, the administrator can perform tasks such as stopping, pausing, continuing, or refreshing the components. Also available to the administrator is server run-time information that might be useful in determining the status of a configuration. Included in this run-time information is load balancing data that tracks information such as the number of client requests processed, the number of clients currently connected, and the total wait time that clients have experienced. Administrators can also update server definitions and their properties as needed through SAS Management Console to respond to the information provided.

SAS Management Console is not the only mechanism through which to administer load balanced servers. Most of the functionality that is found in SAS Management Console is also available for use with application management tools from other providers.

INTEGRATING SAS GRID MANAGER AND SAS INTEGRATION TECHNOLOGIES

The Grid algorithm is a new load balancing algorithm now offered in conjunction with the SAS Workspace Server in SAS 9.2. The grid load balancing algorithm accesses the same information that SAS Grid Manager uses to process other SAS work requests. SAS Grid Manager provides the algorithm with the best machine that can handle the client request. For this reason, users will see the most benefit with the Grid algorithm when using a distributed cluster of SAS Workspace Servers. There are two parts to using the Grid algorithm — configuring the SAS Grid environment, and then configuring the SAS Workspace Server to communicate with SAS Grid Manager.

Configuration of a SAS Grid environment in SAS 9.2 is an automated process that is performed by the SAS Configuration Wizard and takes place during the configuration of the SAS installation process. There is nothing different or unique that needs to be configured in a SAS grid deployment in order for the SAS Workspace Server to communicate with SAS Grid Manager and run in the grid. The details of how to install and configure a SAS Grid environment are available from the Scalability and Performance Community at http://support.sas.com/rnd/scalability/grid/gridinstall.html.

CONFIGURING THE WORKSPACE SERVER TO COMMUNICATE WITH SAS GRID MANAGER

In order to configure the SAS Workspace Server to use SAS Grid Manager for load balancing, you must convert an existing Workspace Server to use load balancing. By default, load balancing is disabled for SAS Workspace Servers. Converting a SAS Workspace Server to use load balancing is done by using the Server Manager plug-in of SAS Management Console. Use the Server Manager plug-in to select the appropriate Logical Workspace Server definition, right-click the definition, select **Convert To**, and then select the **Load Balancing** option from the resulting menu. Diagram 4 is the dialog box that is displayed to configure the Grid algorithm for use with the SAS Workspace Server.

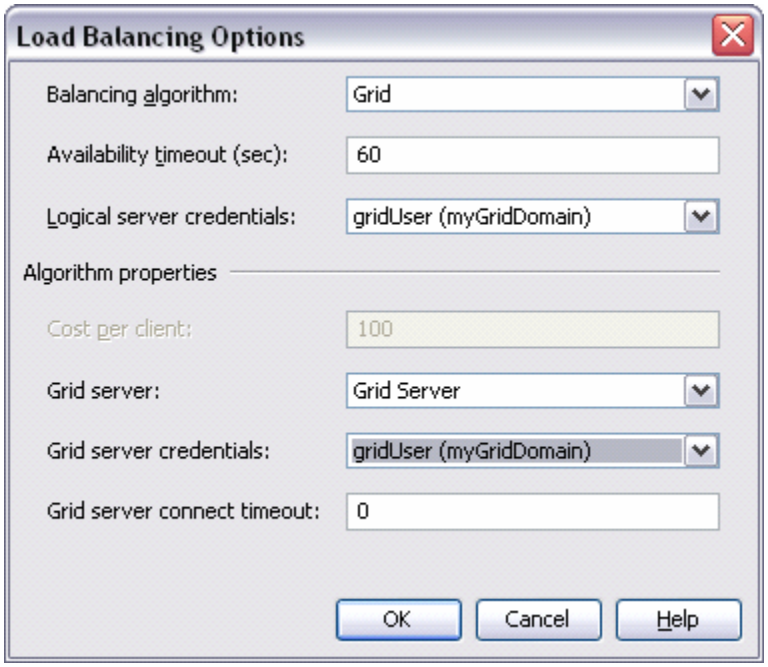

Diagram 4: Template to Specify Load Balancing Option for SAS Workspace Server

Select **Grid** from the **Balancing algorithm** drop-down list to tell the SAS Workspace Server to communicate with SAS Grid Manager for load balancing. In this example, "**Grid Server"** is specified in the **Grid server** field as the grid

server definition to connect to in order to communicate with the grid. The **Grid server credentials** field with **"gridUser (myGridDomain)"** shows the credentials that are used by load balancing to communicate to the specified grid server; in this case, **"Grid Server"**.

BENEFITS OF USING SAS GRID MANAGER WITH SAS WORKSPACE SERVERS

Before SAS 9.2, the SAS Workspace Server provided proprietary load balancing algorithms to facilitate load balancing multiple SAS Workspace Servers in a cluster. The limitation with the proprietary load balancing algorithms is that their view is limited to the SAS execution environment and even more specifically to the execution of SAS Workspace Servers only. Therefore, if there are other SAS programs and applications as well as other applications which are not SAS that are executing in a shared grid infrastructure, the resources that are being consumed by these other applications are outside of the scope of the proprietary Workspace Server load balancing algorithms and therefore, not taken into account. The benefit of integrating SAS Grid Manager and SAS Workspace Servers is the ability to leverage the load balancing capabilities of SAS Grid Manager, which are closely tied to the consumption of resources by all users of the grid, and as a result, much more capable of assigning work requests to the most appropriate resources.

CONCLUSION

There are many benefits to running SAS applications and solutions in a grid environment, including the following:

- creating an environment that can be shared by multiple users and multiple applications
- allocating resources dynamically to meet peak demands
- implementing policies and priorities to govern use of computing resources
- running larger or more complex analysis
- easing maintenance of applications by separating the applications from the infrastructure that is used to run them
- improving price and performance through the use of commodity hardware
- scaling out cost effectively as data volumes, computing needs, and number of users grow

As discussed earlier in this paper, many SAS products and solutions have been integrated with SAS Grid Manager to bring these benefits to you as users and consumers of these applications. Our goal is to continue to expand the integration of SAS Grid Manager with other SAS products and components where it makes sense. Because SAS Workspace Servers are at the core of many SAS products and solutions, it was the obvious next step for integration with SAS Grid Manager.

SAS applications, solutions and users generate many SAS work requests. The value of having these requests managed by SAS Grid Manager is the ability to create a shared SAS grid environment that is managed and load balanced to meet the varied needs of all your SAS users across your enterprise. You also have an enterprise infrastructure that has the flexibility to grow as your organization grows.

ACKNOWLEDGMENTS

The authors acknowledge Dan Jahn, Glenn Horton, Doug Haigh, and Randy Williams at SAS Institute Inc., for their contributions to this paper.

FOR MORE INFORMATION

For more information about SAS grid computing and SAS Integration Technologies, visit the following Web sites:

Introduction to Grid Computing. Scalability and Performance Community. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid.

SAS Grid Computing. Technologies/Enterprise Intelligence Platform. SAS Institute Inc. Available www.sas.com/grid.

Syntax for SAS/CONNECT Grid Functions. Syntax for Grid Enablement. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid/gridfunc.html.

Sample Code for Load Balancing SAS Jobs from Multiple Users. Syntax for Grid Enablement. SAS Institute Inc. Available support.sas.com/rnd/scalability/grid/gridfunc.html.

Tran, A., and R. Williams. 2004. “Implementing Site Policies for SAS Scheduling with Platform JobScheduler.” Available support.sas.com/documentation/whitepaper/technical/JobScheduler.pdf.

SAS Integration Technologies Product Documentation. Available http://support.sas.com/documentation/onlinedoc/inttech/index.html

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the authors:

Cheryl Doninger
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone: 919-531-7941
E-mail: cheryl.doninger@sas.com

Jason Spruill
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone: 919-531-3705
E-mail: jason.spruill@sas.com