Paper 282-2010

Staying Highly Available using SAS® Grid Manager:

The Performance Management Application at Commerzbank AG

Thomas Braun and Miriam Schipper, Commerzbank AG, Frankfurt, Germany
Cheryl Doninger, SAS Institute Inc., Cary, NC
Jochen Kirsten, SAS Institute Inc., Heidelberg, Germany

ABSTRACT

All key performance metrics from the bank's main systems are aggregated by Commerzbank's sales management solution for retail banking, dubbed "Performance Manager." This has resulted in high-speed reporting for sales management that gives Commerzbank a competitive advantage.

While the application has proved to provide excellent scalability based on SAS Scalable Performance Data Server® in the past, new business requirements drove a doubling of the data volumes to be processed and loaded into the application. Facing this requirement, sustaining the scalability of the application while at the same time not extending the load times was the challenge. The solution was to migrate the application to a new architecture, which included the SAS Grid Manager solution.

This presentation outlines some of the challenges and achievements of the project. The main software components used and discussed here are the SAS Grid Manager 9.2 and the SAS Scalable Performance Data Server 4.5.

OVERVIEW

This paper has two parts. The first part describes the business background concentrating on the Performance Manager application. This part has been made available by Commerzbank AG, Frankfurt, Germany. The second part, provided by SAS, details some aspects of the technical implementation of deploying the SAS Scalable Performance Data Server in a grid environment.

INTRODUCTION

Commerzbank uses business intelligence (BI) solutions from SAS, the leader in business intelligence, for its consumer banking sales management. The reporting system for retail banking business provides approximately 6,000 users across the organization with consistent and valuable reports daily, weekly, and monthly on sales performance. At the moment, the application is being prepared for the integration of Dresdner Bank, which involves increasing the user base to over 10,000 users.

REQUIREMENTS FOR SALES MANAGEMENT APPLICATION PERFORMANCE MANAGER

An integrated depiction of Commerzbank's branch network and customer management model, the management of the whole product range, and an instant availability of information are the requirements to be fulfilled. Performance Manager meets all of these sales management challenges, as described in the following passage.

FIRST CHALLENGE: BRANCH NETWORK AND CUSTOMER MANAGEMENT MODEL

The structure of Commerzbank's private and business customers segment distinguishes between main branches, regional branches, and branches, which means a plus of added complexity.

Even more, for a complete view of the branch network, Commerzbank's different branch types (including Branch of the Future, Branch of the Future Plus, and b&p advisory centres) and the integration of Dresdner Bank with its own distribution structure must be accounted for. Constant changes in the network call for a dynamic reorganization of all business data.

Demand for completeness means also including Commerzbank's separate customer market segments and different relationship management models (generalists & specialists).

In a nutshell, identical sales performance information must be provided at all levels.

SECOND CHALLENGE: INTEGRATED PERFORMANCE MANAGEMENT OF COMPLETE PRODUCT OFFERING

A high breadth and depth of available information with simultaneous focus on key performance indicators is one of the main challenges. Complete and uniform depiction of product information, meaning revenues for new and existing business, volume and profitability, is also highly important for sales management.

The following strategic product groups are displayed in the application:

- wealth management
- insurance
- investments
- credit & loan products
- payment instruments (including current accounts)

In today's dynamic market, a short time to market is a crucial success factor. Sales figures of new products must be available as soon as the products are launched for effective sales management. This calls for a high flexibility in the application, particularly when it comes to new products.

THIRD CHALLENGE: IMMEDIATE TRANSLATION OF INFORMATION INTO PERFORMANCE MANAGEMENT

Users must be able to use information intuitively, which means that the provided information must be customized to user's analysis needs. Thus, information has to suit heterogeneous addressees and satisfy differentiated performance management focuses.

On the other hand, data must be available as soon as possible in order to manage effectively. This means that the previous week's numbers must be available at 8 a.m. every Monday morning in order to optimize reaction time.

Online performance must be high to ensure optimal convenience and user acceptance.

This requires an extremely high performance level, especially considering more than 10,000 users that will soon be using Performance Manager, and the 4 terabytes of data that needs to be processed.

IMPROVED PERFORMANCE MANAGER IS THE SOLUTION FOR VERSATILE REQUIREMENTS

The solution for these challenges is a highly customized user interface, a flexible presentation of published KPI's, and innovative application architecture.

FRONTEND

Performance Manager supplies standard reporting for all scopes of Commerzbank's private and business customers.

An authorization system ensures that each user sees precisely the information that is important to him. Thus, customer consultants, for example, receive information about their main achievements and activities: revenues, growth of customers and volume, appointments, and deals. In contrast, branch managers can check their branches as well as the results of their associates. Higher management levels are provided with further aggregated information that shows a top-view of sales performance.

Screens display everything from detailed results for individual consultants to stoplight-style performance graphics and a drill-down management view. Stoplight graphics display whether areas are doing well or need action. For cause studies, drill-down views help to analyze particular positions and to identify factors that constrain performance. Reaction impulses can thus be found more easily.

Additionally, the sales management process is further supported by the management commentary function.

Performance Manager's frontend can be operated intuitively by customer consultants as well as executives without significant training.

KPIS

All KPIs from the bank's operational systems are aggregated by Performance Manager. This allows various interactions to be identified and enables Commerzbank to assess consumer banking activities from different perspectives.

Depending on the analysis focus, the application supplies daily, weekly, or monthly reports. Analysis criteria include consultant type, region, and consumer segment as well as ratios of customers to full-time staff. So Commerzbank's branch sales staff is able to use a standard reporting that implements the same logic for all scopes and users, and thus provides uniform and consistent results.

The following picture gives an insight into the report structure.

	Daily Sales Support	**Weekly** Activities	**Monthly** Revenues
Standard Reporting	**Sales support** - Investments	**High-level reports** - Sales performance (weekly) - Activities - Appointments **Detail Reports** - Product details & drivers **Ranking** - Overview branch network **Management-commentary**	**High-level reports** - Sales performance (monthly) - Asset-levels & customer numbers - Revenue **Detail Reports** - Product details & drivers **Ranking** - Overview branch network **Management-commentary**
Advanced Analysis Options		**Deal overview** - Itemization of weekly sales performance for consultants	**Basic reports** - High breadth and depth - Time series analysis
Emphasis	*Activity management*	*Sales performance management*	*Revenue management*

Figure 1. Reporting in "Performance Manager"

Each reported time interval has a different focus on sales performance. In this way, the application publishes very specific, yet diverse information for performance management.

ARCHITECTURE

Commerzbank decided to go with SAS predominantly for performance, flexibility and scalability. SAS can easily be expanded and modified to continue to meet Commerzbank's needs in the future. The following image summarizes the three stages of architecture extension.

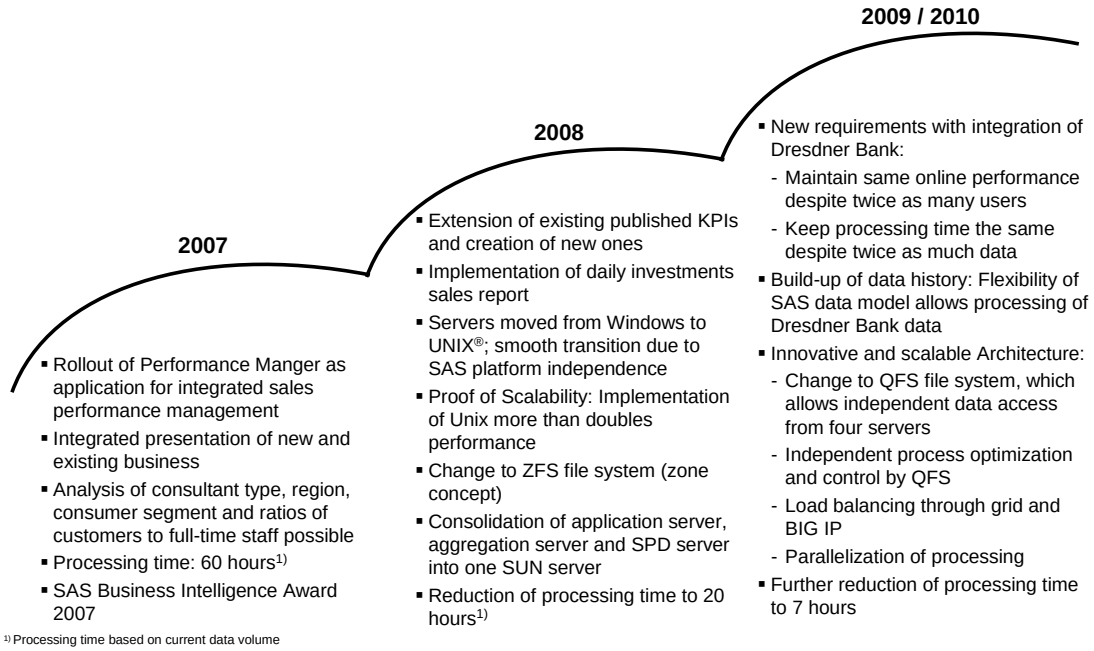

Figure 2. The Third Stage of Architecture Extension Prepares for Integration of Dresdner Bank

Performance Manager stands out due to its extremely high performance. This would not be possible without SAS Scalable Performance Data Server, which especially excels in the rapid processing of large amounts of data. Performance Manager currently processes 4 terabytes of data. Performance Manager thus sets a real industry standard by guaranteeing the speed necessary for top-quality analysis.

The quality and innovation of reporting that is supplied over the weekend can be only achieved by the combination of the load-balancing provided by SAS Grid Manager and SPD Server including BIG IP, and scalable QFS.

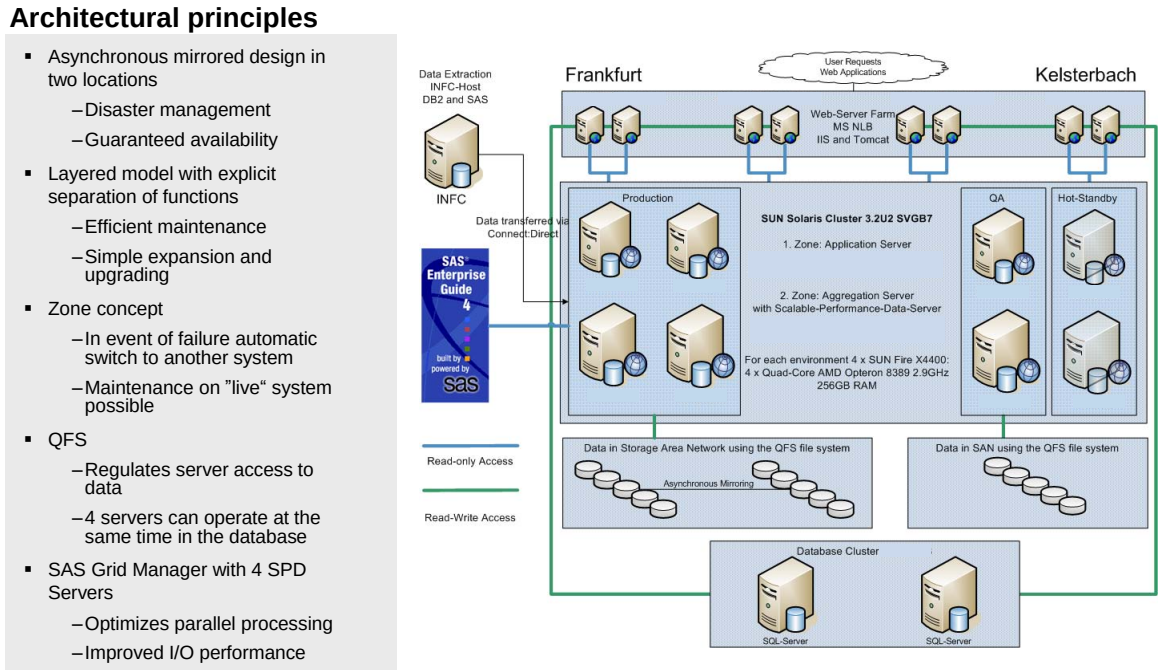

Figure 3. Overview of the New SAS Platform

Business intelligence specialists Anaxima and SAS supported Commerzbank in implementing the solution.
In spite of the demanding requirements, reports for the previous week are ready every Monday morning for every user and online performance is at a high level.

CONCLUSION

Altogether, there are four key factors that contribute to Performance Manager's success:

- The set of key performance indicators and key figures are shaped according to business strategy and sales structure.
- An intuitive frontend with diverse analysis options, a short period for processing data, and a specific range of information for every user group ensures high user acceptance.
- A uniform data basis secures integrated information and the flexible metadata model can react instantly, if changes are required.
- The architecture and infrastructure are scalable, innovative, and flexible.

SAS SOFTWARE COMPONENTS USED FOR THE IMPLEMENTATION

The following sections explain the SAS software components used and the fine points of some technical information around such a grid installation.

SAS GRID MANAGER

SAS Grid Manager was introduced in SAS 9.1.3. It builds upon the parallel capabilities of SAS/CONNECT and adds many other capabilities required by enterprise grid deployments. SAS Grid Manager provides load balancing, policy enforcement, efficient resource allocation, and prioritization for SAS products and solutions running in a shared grid environment. In addition, it de-couples the SAS applications and the infrastructure used to execute the applications. This allows 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 allows you to create a managed, flexible, and shared environment to efficiently process the following to meet your business needs:

- multiple users
- parallel workloads
- enterprise scheduling

Multiple Users

Many customers have a number of ad hoc SAS users that develop models, 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 a particular share of the computing resources, and the prevention of mass submission of work requests that result in frequent server crashes. SAS Grid Manager will map the SAS work requests to the available 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 needed to create an organized and managed SAS analytic environment to ensure that the appropriate resources are allocated to the appropriate users and that the workload meets the objectives of the organization.

The total SAS user community is likely using many different interfaces for running SAS applications. Some portion of the users might be running SAS in interactive Display Manager System (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 applications and SAS Enterprise Guide tasks to a SAS grid for execution. SAS Grid Manager provides the infrastructure to balance all SAS applications and workload across a shared grid infrastructure.

Parallel Workloads

A subset of SAS applications consist of sub-tasks 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 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 will spawn every iteration of the loop to a SAS grid via SAS Grid Manager.

SAS Risk Dimensions has a similar iterative workflow of executing the same analysis over different subsets of data. In the case of SAS Risk Dimensions, the data is subset based on market states or by instruments, and every 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 component 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. Finally, a

programmer can 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. In this case, the programmer can use the grid syntax available in the SAS 4GL programming language to create a grid-enabled application. In all of these scenarios, SAS Grid Manager will distribute and load balance 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 the SAS Management Console to enable you to create SAS workflows and to schedule them based on time and file events, or both. SAS Grid Manager will then distribute the jobs within one or more workflows 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 as well as 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 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.

The use of SAS Grid Manager Software enables workload stemming from data integration, reporting, and solution clients to be dynamically and transparently routed to the least used node at this time. That can be any host participating in the grid, regardless of whether the host is located in a local or remote site as long as it can be reached over the network. This ensures that a client's request will be executed in the shortest possible time. In fact, it guarantees that the request will be run, even when one or more grid nodes are not available due to ongoing maintenance or even a hardware failure.

Controlling Resources with Queues

In order to have fine-grained control over the resources on the grid, the grid can be partitioned into sub-grids. There can, for example, be a series of hosts forming a sub-grid for data integration (ETL) and another bunch of computers dedicated for reporting (BI) tasks.

From a Platform Load Sharing Facility (LSF) standpoint, these sub-grids are controlled by so-called queues. Queues can be used to allocate hosts in a grid across multiple dimensions such as time, users, priority, resources, and so on. Depending on the configuration, certain hosts can participate in their sub-grid only for a limited time slot each day and can be part of other execution units during the rest of the day. Certain user IDs can be assigned to a dedicated queue, assuring that these users can use only resources from the hosts defined in this queue. Queues can be assigned priority classes, too, with the capability of preempting lower priority jobs in favor of higher priority jobs coming in.

```
Begin Queue
QUEUE_NAME = High
PRIORITY = 80
QJOB_LIMIT=100
PREEMPTION = PREEMPTABLE[Med Low]
End Queue

Begin Queue
QUEUE_NAME = Med
PRIORITY = 90
QJOB_LIMIT=25
PREEMPTION = PREEMPTABLE[Low] PREEMPTIVE[High]
End Queue

Begin Queue
QUEUE_NAME = Low
PRIORITY = 100
QJOB_LIMIT=15
PREEMPTION = PREEMPTIVE[Med+1 High+2]
End Queue
```

The script above shows an excerpt from a queue definition with three queues defined, High, Med, and Low. The definition will allow the high priority queue to use all of the resources unless there is a medium or low priority job. It also tells LSF that high priority queue jobs can preempt jobs from medium or low priority queues. Resource allocation can also be made more flexible than shown above using the FAIRSHARE capability. This allows queues with queue job limits to use more resources than defined by the limit as long as no jobs in other queues are competing for resources. In the project discussed here, a skew of the server load balance was detected during certain times. The usage of queues corrected this issue.

Grid-Enabling Legacy SAS Code

From a SAS program perspective, any SAS code can be executed on a grid either in one piece or when being divided in its independent units of work. A perfect example of an independent unit of work would be an SQL query sent by a reporting client. Another good example of this is the loading and index creation of an SPD Server table. Multiple tables can then be combined to form an SPD Server dynamic cluster table. This will be discussed below.
In order to help enable legacy SAS code for execution on a grid, the SAS 9.2 Code Analyzer, the SCAPROC procedure, can be used. This procedure will run the code to identify the units of work that can be run independently from each other and insert grid statements in the code such that it can be easily deployed on a grid.

The example below features two independent units of work (SUMMARY procedure invocations). It records the Code Analyzer information into *c:\comments_out.txt* (not discussed here) and also runs the Grid Job Generator outputting the grid-enabled version of the source into *c:\grid_out.txt*:

```
proc scaproc;
   record   "c:\temp\comments_out.txt"    /* Code Analyzer comments */
   grid     "c:\temp\grid_out.txt"        /* Grid exploiting code */
   resource "SASApp";                     /* SAS server*/
run;

proc summary data=wk_data.a;
   var x; output out=new1 mean=mx;
run;
proc summary data=wk_data.a;
   var y; output out=new2 mean=my;
run;

proc scaproc; write; run;                        /* Closes the results files */
```

The grid-enabled version stored in *c:\grid_out.sas* will look like this:

```
/*----------------------------------------------------------------------------*/
/* There are 2 tasks in this job.                                             */
/* 2 of these tasks can be RSUBMITed.                                         */
/* These 2 tasks used 15 units of time.                                       */
/* The longest task took 15 units of time,  100% of total time.               */
/*----------------------------------------------------------------------------*/
%let SCAGRID_SESSIONS=3;
/*----------------------------------------------------------------------------*/
/* Enable grid service                                                        */
/*----------------------------------------------------------------------------*/
%let rc=%sysfunc(grdsvc_enable(_all_, resource=SASApp));

/*----------------------------------------------------------------------------*/
/* This macro starts up the connect sessions                                  */
/*----------------------------------------------------------------------------*/
%macro scagrid_sessions(count);
  %do i = 1 %to &count;
    signon sess&i connectwait=no;
    %put Session started on grid node %sysfunc(grdsvc_getname(sess&i));
  %end;
%mend scagrid_sessions;

/*----------------------------------------------------------------------------*/
/* Start up our connect sessions.                                             */
/*----------------------------------------------------------------------------*/
%scagrid_sessions(&SCAGRID_SESSIONS);
/*----------------------------------------------------------------------------*/
/* This function call initializes data structures for our SCAGRID functions. */
/* We pass in the number of sessions and the number tasks in this job.       */
/*----------------------------------------------------------------------------*/
proc scaproc; startup 2 &SCAGRID_SESSIONS; run;

/*----------------------------------------------------------------------------*/
/* TASK 1 run in rsubmit                                                      */
/*----------------------------------------------------------------------------*/
/* I/O Activity                                                               */
```

```
/*---------------------------------------------------------------------------*/
/* DATASET    OUTPUT SEQ   created in task 1 WORK.NEW1.DATA                   */
/*---------------------------------------------------------------------------*/
/* Symbol activity                                                           */
/*---------------------------------------------------------------------------*/
/* SYMBOL GET task: 0 used by subsequent task:no  Name:SYS_IOUSEEE           */
/*---------------------------------------------------------------------------*/
/* ELAPSED 0                                                                 */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                     */
/*---------------------------------------------------------------------------*/
proc scaproc; getsession 1 "sess"; run;
%put sess=&sess;
/*---------------------------------------------------------------------------*/
/* rsubmit for task 1                                                        */
/*---------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_1;
proc summary data=wk_data.a;
       var x; output out=new1 mean=mx;
run;
endrsubmit;

/*---------------------------------------------------------------------------*/
/* TASK 2 run in rsubmit                                                     */
/*---------------------------------------------------------------------------*/
/* I/O Activity                                                              */
/*---------------------------------------------------------------------------*/
/* DATASET    OUTPUT SEQ   created in task 2 WORK.NEW2.DATA                   */
/* FILE       OUTPUT       created in task 2 c:\temp\comments_out.txt         */
/*---------------------------------------------------------------------------*/
/* Symbol activity                                                           */
/*---------------------------------------------------------------------------*/
/* SYMBOL GET task:0 used by subsequent task:no  Name:SYS_IOUSEEE            */
/*---------------------------------------------------------------------------*/
/* ELAPSED 15                                                                */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Dependancies                                                              */
/* Highest task depended on: 0                                               */
/*---------------------------------------------------------------------------*/
/*---------------------------------------------------------------------------*/
/* Get an available session with the scagrid_gs function                     */
/*---------------------------------------------------------------------------*/
proc scaproc; getsession 2 "sess"; run;
%put sess=&sess;
/*---------------------------------------------------------------------------*/
/* rsubmit for task 2                                                        */
/*---------------------------------------------------------------------------*/
rsubmit &sess sysrputsync=yes cmacvar=scagrid_task_2;
proc summary data=wk_data.a;
       var y; output out=new2 mean=my;
run;
endrsubmit;
/*---------------------------------------------------------------------------*/
/* This macro issues waitfors and signoffs for our sessions.                 */
/*---------------------------------------------------------------------------*/
%macro scagrid_waitfors(count);
  %do i = 1 %to &count;
    waitfor sess&i;
    signoff sess&i;
  %end;
%mend scagrid_waitfors;
/*---------------------------------------------------------------------------*/
/* Wait for and sign off all sessions.                                       */
/*---------------------------------------------------------------------------*/
```

```
%scagrid_waitfors(&SCAGRID_SESSIONS);
/*-----------------------------------------------------------------------------*/
/* Termination for our SCAGRID functions.                                      */
/*-----------------------------------------------------------------------------*/
proc scaproc; shutdown; run;
/*-----------------------------------------------------------------------------*/
/* All done.                                                                   */
/*-----------------------------------------------------------------------------*/
```

Note that the automatically grid-enabled code can be submitted to the grid as is.

The grid-enablement of legacy SAS code, which has already been using MPCONNECT capabilities for remote submission of tasks is almost trivial, as most of the statements are already present.

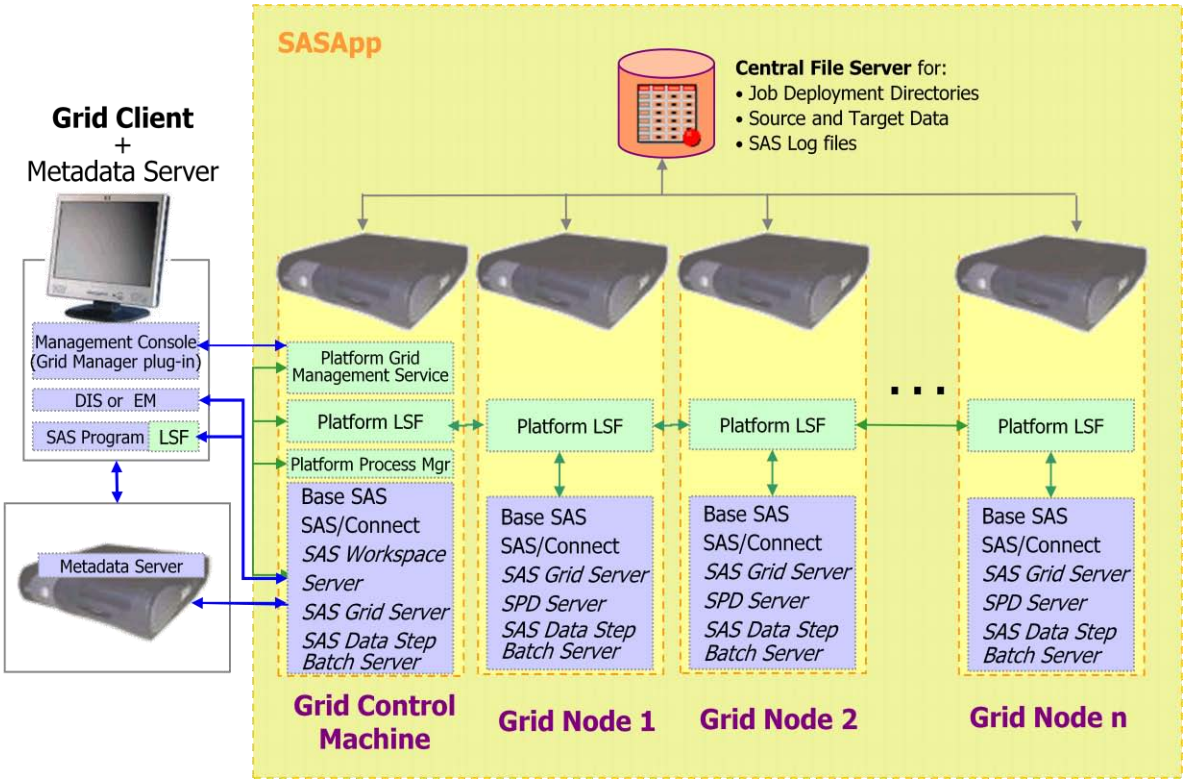

Figure 4. SAS Grid Architecture Topology

Given a grid topology like the above, one of the key availability challenges is with the SAS Metadata Server. The Metadata Server must appear to exist at a constant host name, which can be resolved to the IP-address of the new failover host.

SAS Metadata Server High Availability

As a central component of the grid and overall architecture, the SAS metadata server needs to be running and accessible at any time. This would require failover capabilities for the metadata server instance to be in place in case there is a hardware problem with the node that the metadata server runs on. While there are high availability solutions available from 3rd party vendors offering this capability, these would add complexity to the infrastructure as well as extra cost in terms of purchase, deployment and staff training.

Platform Enterprise Grid Orchestrator (EGO) is part of the Platform Suite for SAS Version 4.1 and is provided as part of SAS Grid Manager for 9.2. It can also be obtained through SAS Technical Support for use with SAS 9.1.3. EGO is a collection of cluster orchestration software components that, among other things, provide high availability to critical services. Besides other functionality, EGO can ensure high availability of services running on the grid through built-in disaster recovery scenarios.

EGO can be configured to fail over the SAS Metadata Server to another node in the grid in case its current node becomes unavailable. Hence, the grid will transparently re-start the SAS Metadata Server on a new node, rendering additional hardware, which might be unnecessarily sitting idle until used. After being re-started on the new node, the

SAS Metadata Server is transparently accessible with the same connection credentials, providing business continuity to all consumers of the grid.

The following steps describe a desirable SAS Metadata Server failover scenario using EGO[1]:

- The SAS Metadata Server is configured to be started by EGO on one of the hosts within the grid.
- SAS grid clients can look up the host address for the SAS Metadata Server. For example: It is important that clients must be configured to use the host name instead of the IP-address for accessing the server since the IP-address will change during failover.
- The SAS Metadata Server goes down unexpectedly as its host crashes.
- EGO detects that the SAS Metadata Server is no longer available, and tries to restart it on the next available host, based on the control policy and resource requirements.
- The SAS Metadata Server is successfully restarted. Its host name resolves to a new IP address.
- The recovery is completed. Clients can still access the SAS service by connecting to the host name they know, even though the SAS service is restarted on a different host.

In brief, the benefits of using EGO for high availability of the SAS Metadata Server are:

- no additional software license required
- no extra stand-by hardware needed to run the metadata server as any grid node can take over
- less complex environment as EGO is part of LSF

SAS Grid Manager Client Utility

The SAS Grid Manager Client Utility (SASGSUB) is a stand-alone command-line utility that enables users to submit SAS programs to a grid for processing. It is available with SAS 9.2 Maintenance Release 2. This utility allows a grid client to submit SAS programs to a grid without having SAS installed on the machine performing the submission. It also enables jobs to be processed on the grid without requiring that the client remain active (submit and forget). You can use the command to submit jobs to the grid, view job status, retrieve results, and terminate jobs. The SASGSUB utility can enable the SAS checkpoint and restart execution modes for fail-safe job execution on the grid. This process can be automated using the LSF re-queue capability.

SAS 9.2 Checkpoint and Restart Execution

Used together, checkpoint mode and restart mode enable batch programs that terminate before completing to be resubmitted, resuming execution with the DATA or PROC step that was executing when the failure occurred. DATA and PROC steps that already completed will not be re-executed unless this is desired.
When checkpoint mode is enabled, SAS records information about DATA and PROC steps in a checkpoint library. When a batch program terminates prematurely, it can be resubmitted in restart mode to complete execution. In restart mode, global statements and macros are re-executed and SAS reads the data in the checkpoint library to determine which steps completed. Program execution resumes with the step that was executing when the failure occurred.
The checkpoint-restart data contains only information about the DATA and PROC steps that completed and the step that did not complete. It does not contain information about macro variables, macro definitions, SAS data sets, or any other information that might have been processed in the step that did not complete.

Data Storage is the Key

Unlike some grid environments, for example, those that are used for scaling out Monte Carlo simulations and use relatively small data sets, most grid deployments in the commercial space rely on a central storage for shared data access. The I/O bandwidth of the shared storage depends on the number of grid nodes that need concurrent access. In order to satisfy loading and reading performance, the shared storage bandwidth of an installation that is similar to what is discussed here would be around 2GB/s. Each SPD Server instance would get its share according to the load produced by all instances. The actual installation uses the QFS 5.0[2] file system from Sun Microsystems. It should be pointed out that although shared storage most often works well out-of-the-box, this can require tuning that would involve system engineers from the vendor to optimize this central component of the grid infrastructure. The tuning parameters that are used will depend on the individual mix of loading (ETL) and reporting tasks executed and can be different for each deployment. For the project discussed in this paper a close relationship was maintained with system engineers from Sun Microsystems to optimize the performance of this central piece of the installation.

[1] Detailed configuration steps of SAS Metadata Server high availability are discussed in the SFG 2009 paper **Achieving High Availability in a SAS Grid Environment** of Daniel Wong, Platform Computing Inc., Markham, Canada
[2] Shared file systems like GPFS from IBM and other vendors are also supported.

SAS SCALABLE PERFORMANCE DATA SERVER

SAS Scalable Performance Data Server (SPD Server) software is designed for high-performance data delivery. Its primary function is to provide user access to SAS data for intensive processing (queries and sorts) on the host server machine. When client workstations from varying operating platforms send processing requests to an SPD Server host, the host returns results in the format required by each client workstation. SPD Server uses parallel processing to exploit the threading capabilities of servers with multiple processors.

SPD Server executes threads, units of processing, in parallel on an SPD Server host. The software tasks are performed in conjunction with an operating system that enables threads to execute on any of the machine's available processors. A specialized machine and operating system are important processing partners, but SPD Server's power is derived from the software architecture that enables it to rapidly and efficiently process SAS data in concurrent parallel threads on multiple processors.

Scalable Performance Data Server Client/Server Model

The main architectural server components are the Name Server and the Data Server. When the Data Server starts, it registers the data domains that it serves with the Name Server. A client wishing to connect to a Data Server specifies, among other credentials, the domain name making a LIBNAME connection request to the Name Server. The Name Server routes the Data Server's connection credentials to the client, who will then connect to the data server that spawns a proxy server instance for the actual data access. It is possible to deploy single or multiple instances of the name and data servers in a network.

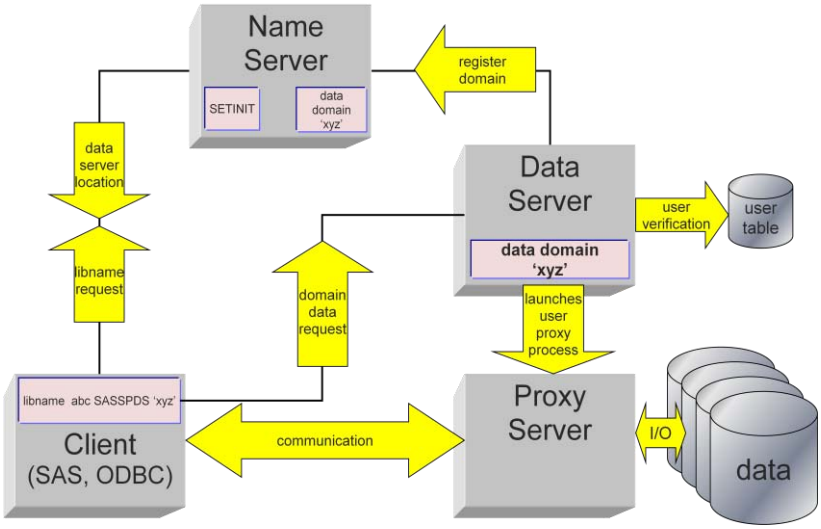

Figure 5. SPD Server Architectural Model

Modes of Deployment

SPD Server can be deployed in a variety of ways. Following the most used deployment forms for commercial use:

- Standard
  features one Name Server and one Data Server serving the data domains. The name and data servers can be running on the same host or different hosts.

- Multi-data server
  has a single Name Server gaining access to multiple Data Servers on different machines. Each Data Server serves its own set of data domains, either on locally-attached disks or in a private file system on SAN. This type of deployment scales well as long as the access pattern has a good distribution over the various data domains and not everybody uses the same data, hence host. This deployment is complex and requires some planning.

- Grid
  consists of a number of hosts, forming a grid, all with identical installations of one Name Server and one Data Server. Each host can be looked at as a standard deployment as described above. The load balancing between the hosts can be accomplished by the SAS Grid Manager and LSF. This ensures that there will always be an optimal distribution of the load across the hosts regardless which data is accessed. All hosts have access to the same data sitting on shared storage. This is the most scalable deployment and is discussed in this paper. The

baseline release for a grid deployment of SPD Server is 4.3 or higher. These have been tested to run on a grid. It is a requirement that all grid nodes use the same hardware platform and operating system type.

The SAS 9.2 software components, including the Scalable Performance Data Server 4.5, are installed locally on each of the grid nodes, minimizing start-up latency of the executables. Also, the SPD Server start-up script and the parameter files are kept on a per-local instance basis, ensuring that each server instance uses its own log and audit files as well as parameter settings, should these need to be different. The SPD Server user database can be kept in a central place on shared storage, gaining each server instance access to the same user information, which makes the maintenance of user accounts easier.

The SAS client and SPD Server execute on the same zone. This way they form a unit from a load distribution perspective. Whether a client request will be worked on by SAS or SPD Server, the zone will always show a resource consumption representing the request's unique load profile. This ensures that the load distribution will be optimal across all nodes. Furthermore, when the SAS client and SPD Server are executing in the same zone or host, a local protocol (UNIX domain sockets) will be used by default for the communication instead of using TCP-sockets. UNIX domain sockets are typically much faster, speeding up communication significantly.

Also, all code making an SPD Server LIBNAME connection submitted to the grid can use the same host reference to *localhost* as well as the same TCP-port, making it independent of the node that will actually be executed.

```
libname dm sasspds "datamart" user=sasdemo password=mypass server=localhost.5400
IP=YES DISCONNECT=YES COMPRESS=YES;
```

The code above illustrates a LIBNAME statement connecting to SPD Server on the local host at port 5400, which can be used on all nodes in the grid. The last three options have the following meaning:

- IP=YES
  specifies that if the ANSI standard SQL submitted to SPD Server would run faster in pass-through SQL, the query will be converted to (implicit) pass-through SQL on the fly.
- DISCONNECT=YES
  terminates the SPD Server proxy server when the LIBNAME is cleared, freeing up resources.
- COMPRESS=YES
  uses compression with all SPD Server tables in the domain. Data compression is discussed below.

Dynamic Cluster Tables

SPD Server is designed to meet the storage and performance demands that are associated with processing large amounts of data using SAS. As the size of the data grows, the demand to process that data increases, and storage architecture must change to keep up with business needs.

SPD Server offers dynamic cluster tables. Earlier releases of SPD Server provided a type of cluster table called the time-based partitioning table. To optimize the benefits of the clustering, the SPD Server administrator can use dynamic clusters to partition SPD Server data tables for speed and enhanced I/O-processing. Clustering is performed using metadata, that when combined with SPD Server functionality, provides parallel processing capabilities for loading and querying data tables. Parallel processing can accelerate performance and increase the manageability, flexibility, and scalability of very large data stores. A cluster table is read-only.

A cluster table consists of multiple member tables. Each member table can be loaded or otherwise refreshed independently by its own ETL process. These processes can be scaled out on a grid in parallel. Hence the time manipulating the data in a cluster can be broken down greatly. The same is true for backing up the member tables. Tests have demonstrated that the parallel loading of member tables, including possible creation of indexes, scales virtually linearly with the limitation being the I/O bandwidth of the shared storage. The customer that is discussed in this paper uses this feature quite extensively to break down their load times, providing for scalability as the data volume is expected to double because of the merger.

After the member tables have been loaded in parallel there should be a dedicated SPD Server instance bringing them together in a cluster table. This can be easily controlled by a batch job.
Any manipulation of the tables' access control lists (ACL) that define access permissions cannot be shared by multiple server instances at this time and should be done by a dedicated master instance.

Aging out old data is easy and fast when the member tables are organized along a time dimension. For example, each table holding the data of a limited time period, such as a month or a quarter. Instead of running lengthy SQL DELETE statements, without actually re-gaining the space of the deleted rows, the oldest data can be aged out very fast by simply un-clustering the cluster table, deleting the table holding the oldest physical data and re-creating the cluster table again. Of course a backup of that table can be performed if needed before deleting it. In case a new time slot has arrived, this can be included in the new cluster at this time. Tables containing new time slots can easily be created on the side without interfering with any reporting from the cluster structure.

As the data in the member tables can be organized along multiple dimensions at the same time, diligent clustering of data in these member tables can speed up queries significantly. A query will selectively only have to touch the member tables, which would contribute to the result. Selecting the contributing member tables is achieved by a structure called MINMAXVARLIST without having to read the actual table. This is a pseudo index holding the smallest and greatest value of a table column. The table will be read only by the query when the pseudo index indicates that there is data in the table satisfying the query.

The following table above shows the run-time advantage of a cluster table when the clustering of the data matches the filter predicate of a query accessing the table. The data is clustered along two dimensions, a time dimension and a quantity dimension. This results in a total of 64 member tables. The maximum number of member tables is 64K in the current releases of SPD Server. Depending on how many member tables are accessed by the query, the run-time, compared to a non-clustered table, can be as much as 68 times faster. This is when only two out of the 64 data buckets are accessed. In a worst-case scenario, when all member tables are accessed, the run-time of the cluster table will compare to the non-clustered table.

Query Predicate (only the relevant filtering criterion is shown)	**Classic Table Query Run-time [s]**	**Cluster Table Query Run-time [s]**	**Run-time Advantage Factor**	**Buckets Possibly Hit**
`date between "01JUL04:00:00:00"dt and "31AUG04:00:00:00"dt and quantity > 4`	**270**	**4**	**68**	**2**
`date between "01JUN03:00:00:00"dt and "31AUG04:00:00:00"dt and quantity > 4`	**215**	**4**	**54**	**12**
`date between "01JAN03:00:00:00"dt and "31AUG03:00:00:00"dt and quantity > 0`	**134**	**24**	**5.6**	**12**
`date between "01JUN03:00:00:00"dt and "31AUG04:00:00:00"dt and quantity > 1`	**155**	**44**	**3.5**	**24**
`date between "01JAN03:00:00:00"dt and "31DEC06:00:00:00"dt and quantity > 0`	**120**	**106**	**1.1**	**64**

Figure 6. Runtime Comparisons of Queries Classic Versus Clustered Tables

Materialized Views
SPD Server allows users to create an SQL view as a materialized view. What makes a materialized view different from an SQL view? For a materialized view, the results of the view statement are computed and saved in a temporary SPD Server table at the time the view is created. For a standard SQL view, the results are computed each time the view is referenced in a subsequent SQL statement. As long as there are no changes to any of the input tables that the view consists of, the materialized view will return the results from the temporary table when the view is referenced in an SQL statement. If any of the input tables that comprise the view are modified, the materialized view will re-compute the results the next time that the view is referenced and it will refresh the temporary table with the new results. The materialized view temporary results table exists for as long as the view is in existence. When a view is dropped or deleted, then the temporary results table is also deleted.

In the reporting area where the number of concurrent users can easily be in the order of 5000 users or more, the use of materialized views can be considered at some stage. Storing these on solid state disks (SSD) for best performance is currently being looked at.

Data Compression
SPD Server, like Base SAS, offers compression of data sets using an algorithm called run length encoding (RLE). Compression can be turned on with the data set or LIBNAME option, COMPRESS=YES. Although the achieved compression rate is data-dependent, compression can reduce the file size down to 30%–40% of the uncompressed version in some cases. While compression is primarily used to save disk space, it can also be used to increase the bandwidth of the I/O-subsystem as the data is read in compressed form into memory. For example, a data set which would compress to half the size of the uncompressed version would approximately double the I/O transfer rate when being brought to or from the hard disks. It should be noted that additional CPU-cycles will be used to facilitate the compression or decompression of the data, hence data compression should be used only when the data set compresses well and there are enough CPU-cycles available to handle the additional overhead without interfering with other tasks' processing requirements. The project discussed in this paper uses data compression to speed up the I/O-transfer from the SAN to the hosts that are located in the remote, as seen from the SAN, IT-center in particular.

CONCLUSION

The project discussed in this paper demonstrates nicely how the SAS software and 3rd party components, used in conjunction with careful planning and an intelligent design, form an overall solution which not only helps solving an actual business problem, but also gains added value facilitating high availability and optimal resource utilization.

ACKNOWLEDGMENTS

Christoph Morgen, SAS Institute, Inc., Germany helped a lot in making this paper possible by coordinating between the various contributing parties.

CONTACT INFORMATION

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

Jochen Kirsten
SAS Institute
E-mail: Jochen.Kirsten@sas.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.