

Techniques for Distributed Application Development

Cheryl Garner, SAS Institute Inc.

INTRODUCTION

Applications of all kinds have four basic elements. First there's the need to access data. Then there's a need to manage data — to get the data in the right format for further investigation. There's also a need to analyze data, applying logic in an effort to translate raw data into useful information. And finally, there's a need to present the information obtained from analysis.

The access to the data and the processing involved in managing and analyzing the data can be distributed across platforms in a variety of ways. The following sections describe the types of services provided by the SAS® System to enable optimal distribution of these activities. These services include compute services, data transfer services, and a new experimental capability offered by the SAS System - remote library services. Guidelines for choosing the right service or combination of services are also given along with further usage details.

SERVICES FOR IMPLEMENTING DISTRIBUTED APPLICATIONS

The SAS System provides two sets of services to enable the development of distributed applications. The first is compute services; these services give you access to all of the computing resources on your network. The second is remote data services; these services give you access to all of your data regardless of where they are stored. Most often, the best implementation of a distributed application incorporates a combination of compute and remote data services.

Compute services enable utilization of remote computing resources, including hardware, software, and data, to most efficiently execute an application and to maximize the use of all computing resources. The SAS System enables you to move any or all portions of an application's processing to a remote machine to take advantage of remote hardware resources, to utilize software available in the remote environment, to interface with existing legacy systems, and to execute against the remote copy of the data. The results of the remote processing are then returned to the local machine. This is useful when the remote machine has hardware and/or software resources available to more efficiently perform the task at hand. It can also be preferable if the amount of data to be processed is too large to be moved to the local machine or if it is updated too frequently for a local static copy to be useful.

Remote data services enable access to data stored in a remote environment. This access may be as a single user of the data or as multiple users simultaneously accessing the data. It may be to data stored in SAS data sets, external data bases, SAS catalogs, or external files. This capability can be further divided into *data transfer services* and *remote library services*. Data transfer services provide a method for moving a copy of the data from one machine to another machine where a physical copy is created. Subsequent local processing takes place against the local copy of the data without generating further network traffic until you decide to update the copy of the data with another transfer. Remote library services provide access to remote data libraries and move the data through the network as the local execution requests it. A copy of the data is not written to the local file system and the data must pass through the network on any subsequent use by the local processing.

These services are by no means mutually exclusive. The degree to which each is applied and the most beneficial combination of services is completely determined by the needs and goals of each individual application. Each set of services has a fairly defined set

of benefits and drawbacks. By matching these characteristics to the characteristics of your data and your application needs, you can determine how to best combine these services in order to write the most efficient distributed application.

IMPLEMENTATION CONSIDERATIONS

COMPUTE SERVICES

The *compute services* available in the SAS System can give you easy access to many of the remote resources on your network from a single local SAS session. This results in maximum utilization of all of the computing resources within your organization. The following paragraphs describe several common scenarios that illustrate the possible benefits of utilizing compute services to solve part or all of the requirements of a distributed application.

One or more remote machines on your network may have vector processors or other larger, faster hardware resources that would more efficiently accomplish the CPU intensive portions of your application. Compute services allow you to move any or all segments of an application to a remote machine for execution. The results of the remote execution are returned to the local SAS session to be used with further local processing and/or managed by the local GUI.

In addition, a remote machine may have peripherals attached that would otherwise be inaccessible to you on a standalone local system. A PC in another part of your organization may have a plotter or printer attached that you need to use. By simply directing your local SAS session to move the graphics processing to the remote PC and pointing to its output device, you can get a hard copy of your graph from the remote plotter.

Data center rules or data characteristics may mandate a single centralized copy of the data needed by your application. By moving the processing to the remote system where the data reside, there is no need to transfer or create additional copies of the data. This can address security needs as well as enable access to data sources that are too large or dynamic to be transferred.

Using compute services may introduce some negative side effects that conflict with the goals of your distributed application. The following paragraphs address these side effects and offer alternatives to eliminate or minimize them.

Compute services, by definition, require remote CPU cycles. Because the processing takes place on the remote machine, it does require CPU cycles from that machine. This could be a drawback if the goal is to offload work from the remote system. The remote CPU impact could be minimized by utilizing data transfer services to transfer a copy of the data to the local machine and maintaining the processing on the local CPU or by combining compute services and data transfer services. Refer to the section below, "Combining Compute Services and Data Transfer Services", for more information.

Also, compute services create network traffic in order to return the results of the processing back to the local machine. If the processing needs to be repeated frequently, the best solution would be to utilize data transfer services to simply transfer the necessary data to the local system and maintain the processing in the local environment.

Another side effect is that remote production data systems will still be impacted as the data is subsetted or pre-processed. This can be minimized by careful implementation of your application to eliminate or reduce multiple passes of the data.

REMOTE DATA SERVICES

There are many applications, or parts of applications, that require processing to be done in the local SAS session but the data required by that processing reside on a remote file system. In this case, it is necessary to move the data to the local processor for execution. There are two types of services available with the SAS System that can remove the boundaries of your organization's heterogeneous data sources and enable remote data access: data transfer services and remote library services.

Data Transfer Services

For many applications, *data transfer services* offer the maximum benefit. The following paragraphs describe scenarios that lend themselves toward a data transfer solution.

A major benefit of data transfer services is to offload work from a remote system to one or more local machines and boost response time for production systems running in the remote environment. Once the data has been downloaded to the local machine, all subsequent data access and processing is done by the local processor. By moving a copy or subset of the original data to the local machine, production applications can be run on the local machine without further remote CPU consumption or impact on remote production data systems.

Moving a copy of the data to your local system adds robustness to your decision support environment. In the case of a network failure that would temporarily eliminate access to the remote data, you can continue working with your local copy of the data.

You can transfer only the data that you need by using WHERE processing to dynamically subset the data as it is being transferred to the local machine. This reduces network traffic and gives you exactly the data you need on the local system.

Data transfer services not only facilitate moving data from a larger source to a requesting local workstation, but also support the viewpoint of a centralized control point, such as a mainframe, initiating communication to a network of workstations. This enables centralized distribution of data and/or applications. Automated jobs can be set up to allow for distribution of data and/or applications during non-peak hours to multiple machines that have need of the data and/or applications for the next day's work. Likewise, jobs can be set up to query a network of workstations for the purpose of gathering data and pulling it into a centralized repository.

Yet another reason to utilize data transfer services is for backup purposes. Data and/or applications can be copied from one system to another system with more memory resources to provide a backup in case of loss on the local system.

Also, by moving a copy or subset of the data to the local system, applications can be developed and tested without further dependence or impact on remote system response time.

Depending on the goals of your application, data transfer services may have side effects that conflict with your needs. The following paragraphs address these side effects and offer alternatives to eliminate or minimize them.

Transferring a copy of the data to another file system creates multiple copies of the data. The remote data may be updated so frequently that it is not feasible to keep a local copy of the data current enough to be useful. Also, security restrictions at your site may prohibit the possibility of multiple copies of the data. In this case, consider compute services if the amount of data involved is large. If the amount of data is small to medium, remote library services allow the processing to take place on the local system and the data to come from a remote source as the execution requests it.

If the volume of the data that would need to be moved is too large,

then data transfer services alone are not sufficient to meet your needs. The increased network traffic as well as the time it would take to transfer the data may be unacceptable for your application. This scenario needs the functionality of compute services to either move all of the processing to the remote system to execute against the remote data source or a combination of compute services and data transfer services. Refer to the section below, "Combining Compute Services and Data Transfer Services", for more information.

Data transfer services are less transparent. Transferring the data is an explicit step that must be done prior to initiating any local processing. If the volume of data to be accessed is not large and your application requires more transparent access, remote library services should be used.

Remote Library Services

A new capability available in experimental status with the SAS System, *remote library services*, means that the SAS System now provides a comprehensive set of tools for implementing any distributed application with the greatest flexibility and efficiency. This capability is made up of a *remote engine* that executes in the local or client session and a *server* that executes in the remote or server session. The SAS procedures and DATA steps that execute in the local session pass requests to access remote SAS files to the remote engine. The remote engine then communicates the requests for data to the server. The server then administers the requests to SAS files on behalf of the local/client SAS session.

There are some scenarios for which neither compute services or data transfer services offer the ideal solution. The following paragraphs describe some of these scenarios and how remote library services can fit the needs of your application.

Remote library services enable you to access remote data in a multi-user environment through the use of the multi-user server. This gives a big boost to productivity by allowing multiple users to simultaneously access and update the same data source while guaranteeing the integrity of the data.

Now it is possible to perform updates to data in external data bases. Remote library services enable a single user accessing data stored in an external data base to perform updates to that data through the use of the single user server. This permits updates of remote data base data as a result of local processing.

If you need to maintain a single copy of the data on the remote machine and keep the processing on the local system, then you need remote library services. This is the superior choice if the amount of data needed by the local processing is small, the remote data is very frequently updated, or your data center rules prohibit multiple copies of data.

Remote library services also provide more transparent data access. This eliminates an explicit step of coding an upload or download of the data before processing it and permits the graphical user interface of an application to reside on the local system while the data remains in the remote environment (for example, a local FSEDIT of a remote data set). It also allows applications to be built that provide seemingly identical access to local and remote data without requiring any knowledge on the part of the enduser as to where the data reside.

Careful thought must be given to the following: the amount of data that will be accessed by your application, whether you need multi-user or single user data access, whether your application will be making multiple passes of the data, and the resulting effects on your network load. Analyzing these criteria will help you determine when to utilize remote library services and when to utilize data transfer or compute services. The following paragraphs illustrate some situations where remote library services may not be the ideal way to access your data.

Accessing data through remote library services is not efficient for large amounts of data. The data must still move from the remote machine to the local machine, however, it is not written to the local disk. If the local processing is against large amounts of data, it is usually better to use compute services to move the processing to the remote system rather than transfer any of the data.

Similarly, remote library access is not efficient for multiple passes of the data. If the procedures being run make multiple passes of the data or the entire procedure must be run more than once against the data, it would be better to transfer a copy of the data to the local machine. Then you would incur the network traffic cost once rather than paying the cost on each pass of the data.

Also, response time may be degraded by accessing remote data that is being simultaneously updated by other users. If this impact cannot be tolerated, you may need to transfer a copy of the data to the local system and remove the live data from your application.

COMBINING COMPUTE SERVICES AND DATA TRANSFER SERVICES

Several of the scenarios described above suggested a combination of compute services and data transfer services as the optimal choice for portions of a distributed application. The combination of these services provides tremendous flexibility and efficiency to any distributed application.

Your application will benefit from combining these services if you need information from data that is stored on a remote system and you do not want to move a copy of the data to the local system. Reasons for not moving a copy of the data could include the following: the amount of data is too large, the data is frequently updated, or you want to avoid data duplication. Regardless of the motivation for reducing the amount of data that is transferred, incorporating compute services will achieve your goal. Compute services enable you to format and pre-process data into a subsetted or summarized form on the remote system prior to transferring the subsequent smaller amount of data to the local platform. This balances the use of CPU cycles between the local and remote systems and minimizes the amount of data contributing to network traffic.

THE PRODUCTS THAT PROVIDE THE SERVICES

There are two products within the SAS System that provide the services described in this paper: SAS/CONNECT® Software and SAS/SHARE® Software. The following sections describe which product provides which service(s) along with a brief description on how each service is provided.

SAS/CONNECT Software

SAS/CONNECT Software is defined as our primary cooperative processing product for multiple CPU environments. It allows the SAS application developer to optimize computer resources by choosing the appropriate CPU on which to execute the various components of an application. This may mean accessing remote data from local processes or moving the processing to the remote machine where the hardware or data resources reside.

The SAS/CONNECT product provides compute services, data transfer services, and remote library services.

Compute services are provided through the use of the RSUBMIT facility. The RSUBMIT facility causes SAS program statements to be sent from a local SAS session to a SAS session on a remote system for execution. All results of that execution are then returned to the local SAS session for display and management. The RSUBMIT facility is implemented in several ways. The RSUBMIT command provides an interactive means for moving processing to a remote session. The RSUBMIT statement allows you to alternate

between local and remote pieces of execution within a single SAS job submission. There is also a SUBMIT REMOTE implementation within SCL to allow the applications programmer to imbed remote functionality within an application.

Data transfer services are provided by the UPLOAD and DOWNLOAD procedures. These procedures can be used to transfer SAS data sets, catalogs, graphics catalogs, entire data libraries, as well as external files between local and remote systems. These procedures also support WHERE statement processing for dynamic data subsetting.

Remote library services are the newest feature of SAS/CONNECT Software. This experimental feature includes a remote engine and a single user SAS/CONNECT server. The remote engine provided with SAS/CONNECT Software is used to communicate with a SAS/CONNECT server or a SAS/SHARE server. A libname statement is executed in the local session to define a remote library. Then, whenever that library is used in the local execution, the remote data is transmitted to the local execution for processing. The data is not written to disk, so you do not have a local copy of the data. However, it must be brought to the local session in order to be processed, so it does result in network traffic. This capability is ideal for low volume, transaction type processing.

The SAS/CONNECT Server

In order to use a SAS/CONNECT server, you must be signed on to the platform where the remote data reside. (Note: If you have a SAS/SHARE server running on the remote machine, you do not need to signon in order to access a library through the remote SAS/SHARE server. In fact, you would not want to incur the overhead of signing on if you only need to access the SAS/SHARE server in the remote environment.) You then specify a libname statement specifying a remote SAS data library and give the remote session id as the value of the server= option as shown in the following example:

```
SIGNON <rsessid>;  
  
LIBNAME <libref> SASE9 <'SAS-data-library'>  
        SERVER=<rsessid> <engine/host-options>;
```

The SAS/CONNECT server is currently available through the experimental cross-host version of the remote engine. Therefore, SASE9 must be specified as the engine name on the LIBNAME statement. When this version of the remote engine goes production, the production name of REMOTE will be used. (The REMOTE engine is implicit when a SERVER= option is specified on the libname statement.)

The SAS/CONNECT server is initialized in the specified remote SAS session upon execution of a libname statement with a valid remote session id as the value of the server= option. Each time a libname is submitted, the server= value is first checked to see if it matches any currently signed on remote session ids. If there are no remote sessions that match the value of the server= option, it is assumed to be a libname to a SAS/SHARE server. Therefore, SAS/CONNECT and SAS/SHARE servers must have unique names because the use of the SAS/CONNECT server will always override the use of the SAS/SHARE server if the names are the same.

In the following example, a SIGNON is executed to a remote UNIX node named REMNODE. Then, a libname statement pointing to a directory on REMNODE is submitted with the remote session id of

REMNODE specified for the server= value. This results in the initiation of a SAS/CONNECT server on REMNODE and the libref A being defined to this server running on REMNODE.

```
options comamid=tcp;
filename rlink 'tcpunix.scr';
signon remnode;

libname a sase9 '/usr/local/prod'
        server=remnode;
proc datasets lib=a; run;
quit;
```

The libref A can then be used in subsequent local execution to access data in the remote library without first moving a physical copy of the data to the local session. Note that the data must still move through the network to the local session in order to be processed. However, it is not written to disk to produce a local copy of the data.

The access method specified to establish the SAS/CONNECT conversation through signon is also used for communicating with the SAS/CONNECT server. Use of the SAS/CONNECT server is enabled by the following peer-to-peer access methods:

- APPC (MVS and OS/2)
- TCP (all platforms)

Enablement for the SAS/CONNECT server by other peer-to-peer access methods is under development. Support for the terminal-based access methods such as ASYNC, HLLAPI, and TELNET is not planned.

SAS/SHARE Software

SAS/SHARE Software is defined as a set of advanced data base features necessary for multiuser concurrent data update applications utilizing SAS storage formats. In the future, we intend to add database features such as audit trail and integrated integrity constraints. For more information about these features, refer to the paper "Database Features Extend the Scope of SAS/SHARE Software" listed in the section "Where to Find More Information".

The SAS/SHARE product includes the remote engine and a multi-user SAS/SHARE server. Together, the remote engine and SAS/SHARE server provide the remote library services for libraries being accessed by multiple users. A libname statement is executed in the local session to define a remote library. Then, whenever that library is used in the local execution, the remote data is transmitted to the local execution for processing. The distinguishing characteristic is that the SAS/SHARE server allows the remote library and data to be accessed and updated by multiple users and manages those users to ensure the integrity of the data.

The SAS/SHARE Server

If you need multi-user update access to a remote data library then you must have a SAS/SHARE server running in the remote environment. You then define the remote library with a libname statement that specifies the remote server name for the server= option as in the following example:

```
options comamid=tcp;
libname remlib sase9 'sasdemo.edu.screens'
        server=server1;
```

Use of the SAS/SHARE server across machine architectures is enabled by the following peer-to-peer access methods:

- APPC (MVS and OS/2)
- TCP (all platforms)

Enablement for the SAS/SHARE server by other peer-to-peer access methods is under development.

Do I Need SAS/CONNECT Software or SAS/SHARE Software?

There are well-defined differences between the functionality of SAS/CONNECT Software and SAS/SHARE Software. Mapping these characteristics with the requirements of an application should enable you to determine which product you need to get the job done. In many cases, you might find that you want to combine functionality from both products.

The most obvious distinguishing characteristic of SAS/SHARE Software is that it is the only piece of the SAS System that provides multiuser update access to SAS data libraries or SAS files. On the other hand, SAS/CONNECT Software is the only piece of the SAS System that gives you compute services, data transfer services, and single user remote library services.

The following enumerated list defines which product is required on a machine based on the role of that machine.

1. If a machine needs to provide multi-user access to SAS data either for concurrent update access or to serve data to multiple remote users from one SAS session, then you need SAS/SHARE Software on this machine.
2. If you need the advanced database features planned for SAS/SHARE Software, then you need SAS/SHARE Software on this machine.
3. If you need remote data access from a machine and you do not fit the first category, then you need SAS/CONNECT Software on this machine because it includes the remote engine required to access a remote SAS/SHARE server as well as the capabilities in items 3 and 4.
4. If you want a machine to be used as a compute server by remote clients, execute remote SAS sessions and/or provide a secure environment to update DBMS data through the single-user SAS/CONNECT server, then you need SAS/CONNECT Software on this machine.
5. If you want a machine to be able to connect to compute server machines, create and control remote SAS sessions from a local SAS session, then you need SAS/CONNECT Software on this machine.

RESTRICTIONS OF REMOTE LIBRARY SERVICES

There are some restrictions on how and what remote library services can provide. Most of these restrictions exist when the local/user session and remote/server session are running on machines that have differing internal representations of data and the remote library access crosses machine architectures. The following paragraphs detail the current limits and give alternate solutions when possible.

Currently, with respect to remote library services, access to data in remote SAS data libraries is limited to SAS data sets when the local system and the remote system have different data representations. Therefore, if your local application requires remote SAS catalogs or utility files, you must use compute services or data transfer services to implement your application. We hope to enable remote library access for SAS catalogs in a future version of the system.

The peer-to-peer access methods are the only access methods with the ability to support remote library services. The APPC access

method on MVS and OS/2 as well as the TCP/IP access method on all platforms have been modified to support remote library services. If you must use an access method or platform that does not yet provide this support, you should use compute services or data transfer services to implement your application. Development efforts are in progress to add other peer-to-peer access methods to this list.

If you want your local SAS session to access both a SAS/SHARE server and a SAS/CONNECT server, choose a name for your remote SAS/CONNECT session that is different from the name of the SAS/SHARE server you intend to access. As described above in "The SAS/CONNECT Server" section, the server= value specified on the libname statement is first matched against any active SAS/CONNECT conversations. If a match is found, the SAS/CONNECT server will be used. Therefore, if the SAS/SHARE server and the SAS/CONNECT server names were the same, the SAS/SHARE server would never be used.

Character variables are assumed to contain printable character data and numeric variables are assumed to contain numeric data in float decimal format. Storing binary data in character variables or any other unusual use of variables must be avoided when accessing data sets across machine architectures. Character conversion is performed on character variables and numeric conversion is performed on numeric variables. Any unusual data will produce unpredictable results from the translation process.

Especially with numeric variables, translation from one representation to another may alter the value of the variable. A common type of alteration is loss of precision. This occurs when the source representation has more bits of mantissa than the target representation. No warning is produced due to loss of precision during translation. Another type of value distortion is loss of magnitude. This occurs when the source representation has a greater exponent range than the target representation and a value with a magnitude lying in the excess range of the source representation is converted. A warning is produced when this type of alteration occurs.

SAS data sets, typically on IBM mainframe machines, that may be accessed cross-architecture should not include two-byte numeric variables. Due to their underlying floating point representation scheme, most hosts that SAS supports have a minimum numeric variable length of three. A data set containing a length two variable cannot be accessed with remote library services from such a host. You should use compute services or data transfer services in order to access a data set with two-byte numeric variables.

A CASE STUDY - STAR AUTOS DEMO

A prototype has been developed of the systems that would be needed to conduct business in a car rental agency. The hypothetical car rental agency, Star Autos, is a franchised business. The Headquarters for the franchisor, Star Autos, Inc., are in Chicago, IL. Star Autos, Inc. provides vehicle purchasing, marketing, and computer services to franchisees throughout the country. The franchisee owns and maintains cars and local offices, purchasing the franchise name and services from Star Autos, Inc. Both the Headquarters and each of the franchise offices have data and distributed applications that must be implemented and integrated in order to meet their business goals.

Data and Application Requirements of Headquarters Office

Among the application systems provided by the Headquarters are a central reservation data base, a corporate reporting system, and a distribution system for distributing reservations and reports to each of the franchise offices on a nightly basis.

The reservation data base contains reservations for all of the

franchise offices. It is simultaneously updated by multiple travel agents as they add new reservations. It can also be asynchronously accessed by each of the franchise offices for transaction type requests for a specific reservation as well as for adding a new reservation being made at the franchise counter. The Headquarters' reservation data base is also utilized by the franchise offices for adhoc report generation based on total reservations and also rental type.

The corporate reporting system generates production reports in order to compare the performance of the franchise offices. These reports give information on predicted revenue based on the number of reservations, total reservations, and on rental types that are currently reserved. The reports output by this system are stored in external text files.

The distribution system implements an automated method for sending pertinent reservations as well as selected reports to each franchise office on a nightly basis when system load is light and rates are lower. The reservations for each franchise location are copied to the local office each night so that each office will have a current copy of its reservations at the beginning of the next day's business. Each franchise maintains a data set that specifies a set of reports that the franchise is interested in receiving. Part of the distribution system is to query this data set and distribute only the requested reports to each individual franchise.

Data and Application Requirements of Franchise Offices

Several functions have been distributed to the local franchise offices in order to ease the system load on the mainframe at Headquarters and to improve response time for the customer by locating the data at the point where it is used - at each franchise office counter. The systems that are in place at the individual franchises include a contracts system as well as a reporting system.

The contracts system includes a reservation data base and a contract data base that exist on the workstation at the franchise counter. When a customer needs to check out a car, the local reservation data base is queried for the customer's reservation. If it is found in the local data base, the information is completed and the record is stored in the contract data base. If the reservation is not found in the local data base, a query is made to the Headquarters' master reservation data base and, if found, the information is entered and the record is stored in the contract data base. If no reservation exists, a blank entry screen is presented, the information is completed, and the record is stored in the contract data base. In addition, if the customer wishes to return the rental car in a city other than the origin of rental, the contracts system transfers the completed contract to the destination city so that the information is available when the customer returns the car to the final destination.

The reporting system includes methods for accessing production reports that are produced on a regular basis by the Headquarters office as well as a method for generating adhoc reports which require company-wide data maintained on the Headquarters' mainframe. Each franchise maintains a control data set that contains, among other things, the specific production reports that are important to that office. The reporting system first checks for the existence of a requested production report. If the previous night's run was successful, the report will be on the local system, and it will be displayed. If the report is not found, processing will be invoked on the mainframe to generate the report, the report will be moved to the local workstation, and then displayed. The adhoc portion of the reporting system produces graphic output to compare the performance of this franchise to that of the other franchises within the company with respect to reservations as well as rental types. The graphs must be displayed locally but must be run against the master reservation data base because it is the only source of information on all of the franchise offices.

Analysis of the Distributed Application Implementation

The applications just described were implemented in a two phase process. In the first phase, the implementation was done using Version 6 production features of the SAS System, specifically compute services and data transfer services. The second phase made use of the experimental remote library services where appropriate. The following sections focus on several small pieces of the distributed application and discuss how they were implemented with the Version 6 production features and why each piece was or was not converted to make use of remote library services in the second phase of the implementation.

• The Best Solution - Remote Library Services

First, we'll take a look at the contract system running at the local franchise office. Specifically, we'll look at the situation of a reservation not found in the local reservation data base and the process of querying the Headquarters' master reservation data base and finding the reservation there.

The following fragment is SCL program statements that were coded in the first phase of the implementation in order to extract a single reservation from the master reservation data base.

```
submit continue;
  rsubmit &rsessid;
  /* define remote shared library to */
  /* remote SAS session */
  options comamid=xms;
  libname master "sasogg.sug118.library"
    server=prd6srv;
  /* transfer only requested reservation */
  /* to local temp ds */
  proc download data=master.reserv
    out=finlcall status=no;
    where resnum='&resnum'; run;
  endrsubmit;
  proc append base=s18.reserv data=finlcall;
  run;
  endsubmit;

  /* open local reservation data base */
  wherecls = "resnum=" || "" || resnum || "";
  dsid = open("s18.reserv");
  call set(dsid);
  rc = where(dsid, wherecls);
  rc = fetchobs(dsid, 1);
```

Data transfer services are used in this implementation via the PROC DOWNLOAD statement. The download procedure, in this case, makes use of a where statement to copy only the requested reservation to the local workstation where it is then appended to the local reservation data base. The record is copied to the local system because it needs to be displayed in a local data entry window and then saved into the local contract data base. WHERE processing is used to minimize the amount of data being transferred through the network and to provide the optimal response time to the customer waiting at the counter. Compute services could not address the needs of this part of the application because the remote data needs to be displayed and edited by the local GUI therefore the processing must remain on the local system.

Now, let's re-examine this implementation with respect to remote library services. The following SCL fragment represents the second implementation of this piece of the application which takes advantage of remote library services.

```
/* define remote shared library to local */
/* SAS session */
libname master sase9 "sasogg.sug118.library"
  server=mvs.c02xhsrv;

/* open remote Headquarters data base and */
/* search for resnum */
rdsid = open("master.reserv", 'u');
/* build and apply where clause to speed */
/* up retrieval */
wherecls = "resnum=" || "" || resnum || "";
rc = where(rdsid, wherecls);
call set(rdsid);
rc = fetchobs(rdsid, 1);
```

In this implementation, a libname statement is issued in the local SAS session to define the remote library containing the Headquarters' reservation data base. Note that the remote library is being accessed through a SAS/SHARE server. The reservation is then moved to the local session (but not written to disk) when the SCL FETCHOBS function is executed. Note the WHERE and FETCHOBS function calls after the open of the master reservation data base. The use of these functions is very important because it positions the remote file on the desired record. If a LOCATEC function were used in place of the WHERE and FETCHOBS functions, each reservation would have to be transferred to the local session and compared against the specified reservation number during the execution of the LOCATEC function, until a match is found. This would result in very poor performance and a misuse of remote library services. This is a good example of why it is important to remember that the data must still move through the network with remote library services and that the capability should only be used on small to medium amounts of data. It is preferable to use remote library services in this case rather than data transfer services because we are moving a small amount of data and it can be locally edited and saved into the local contract data base as needed.

• The Best Solution - Data Transfer Services

The next part of the application to be examined is the distribution of selected reservations from the master reservation data base to each of the local franchise offices. The implementation involves the mainframe initiating a conversation with each of the franchise workstations and moving selected reservations to each workstation.

The following program enables distribution of reservations to any number of franchise offices. This is accomplished by reading a control data set that provides the network location of each of the franchises and then uploading only those reservations with an origin value that matches the franchise location.

```
/*-----*/
/* Name: DISTRSV.PROGRAM */
/* */
/* This program distributes the */
/* reservations data set to the */
/* franchise offices. */
/*-----*/
length rc 8;

INIT:
submit continue;
/* set up distribution macro */
%macro distrib;
```

```

/* set up loop based on number of */
/* franchise offices */
%if %eval(&n) > 0 %then %do;
    %do i = 1 %to %eval(&n);
        %global nextfran;
        %let nextfran = &&fran&i;
        %global nextrem;
        %let nextrem = &&rem&i;
        %let i = %eval(&i+1);

options remote=&nextrem;
signon;

rsubmit;
    libname mres "d:\counter";
    libname backup "d:\counter\backup";
endrsubmit;

/* upload all reservations for a */
/* particular location */
%let rstep1 = %str(
rsubmit;
    proc upload data=hq.reserv
        out=update status=no;
        where origin=%')&nextfran%str(%);
    run;
endrsubmit;
); &rstep1;

rsubmit;
    /* sort uploaded ds for merge */
    proc sort data=update;
        by resnum;
    run;

    /* backup existing data set */
    proc datasets;
        copy in=mres out=backup;
        select reserv;
    run;

    /* merge new & existing data sets */
    data mres.reserv;
        set mres.reserv update;
        by resnum;
    run;
endrsubmit;
signoff;
%end; /* do until */
%end;

%mend;

/* execute macro to distribute */
/* reservations data set */
%distrib;
endsubmit;

_status_ = 'H';

return;

MAIN:
return;
TERM:
return;

```

In the first phase of the implementation, moving a copy of each office's reservations to that office is clearly a better choice than maintaining the single centralized copy on the mainframe and requiring the franchises to access this copy of the data either through terminal emulation or by transferring single reservations many times throughout the day. The goals of reducing the mainframe workload and improving customer response time as well as the need to manipulate the reservation data in a local GUI, and the probability of little or no change to a reservation once it has been entered all support creating a full copy of each office's reservations on the local workstation.

When considering remote library services for this part of the application you need to consider the goals of offloading work from the mainframe and how often and how much data access is involved. Each franchise is expected to frequently query and update reservation data throughout the day. When you multiply this by the number of franchise offices, the network overhead, if remote library services are used, becomes too great to enable acceptable response time for the customer waiting to check out a car. Also, if the connection to the mainframe were temporarily lost, the franchise counter operations would be crippled. Clearly, it is still desirable to employ data transfer services to create local copies of the reservations designated for each office.

• The Best Solution - Compute Services

The next area to be considered is the adhoc reporting part of the local franchise reporting system. The adhoc reports require graphic output to highlight the performance of a specific office with respect to the rest of the franchise offices.

This segment of SCL code moves all of the processing, the PROC SORT and PROC SUMMARY preprocessing as well as the actual graph generation, to the remote mainframe for execution by the faster hardware and against the remote data. The graphic output is then displayed on the local workstation.

```

submit continue;
rsubmit &rssid;
    proc sort data=master.reserv
        out=tmp; by origin;

    run;
    proc summary data=tmp vardef=n
        noprint; by origin;
        output out=tmp2;
    run;

    title 'Total Reservations';
    pattern1 value=solid color=green;

    goptions dev=grlink ftitle=centx
        ftext=simplex htitle=3;
    proc gchart data=tmp2;
        block origin /sumvar=_freq_
        noheading; label origin='00'x;
    run; quit;
endrsubmit;
endsubmit;

```

The initial implementation of this portion of the application fully exploits compute services. The only data base containing information on all of the franchise offices resides on the Headquarters' mainframe. The mainframe can also provide the computing power necessary to process the large amount of data needed to produce the graphs. However, we want the graphs to be displayed on the local workstation and possibly saved for later reference. Compute services are the logical choice over data

transfer services because the amount of data needed to generate the graphs is too large to be transferred and you do not need or want a copy of the entire reservation data base on the local system.

Even for phase two of the implementation, when we consider the usefulness of remote library services for this portion of the application, compute services still have a clear advantage. The large amount of data involved is an obvious indication that remote library services is not the right choice. Therefore, compute services give the maximum benefit to this segment of the application.

• The Best Solution - Data Transfer Services Combined with Remote Library Services

The final portion of the application that will be discussed is the distribution of production reports from Headquarters to each of the franchise offices based on the information contained in the control data set maintained by each of the franchise offices. The implementation involves the mainframe initiating a conversation with each of the franchise workstations and transferring a set of reports to the franchise based on selection criteria.

The following program enables distribution of reports to any number of franchise offices. This is accomplished by reading a Headquarters' data set that provides the network location of a franchise, querying that office's control data set, generating a text file with the statements necessary to transfer the requested reports, and then repeating this process for each franchise in the Headquarters' data set.

```

/*-----*/
/*
/* Name: DISTRPT.PROGRAM
/*
/* This program distributes reports
/* to the franchise offices.
/*
/*-----*/

length rc 8;

INIT:

submit continue;
/* set up distribution macro */
%macro distrib;

/* set up loop based on number of
/* franchise offices
%if %eval(&n) > 0 %then %do;
  %do i = 1 %to %eval(&n);
    %global nextfran;
    %let nextfran = &&fran&i;
    %global nextrem;
    %let nextrem = &&rem&i;
    %let i = %eval(&i+1);

options remote=&nextrem;
signon;

x 'alloc fi(xferrpt)
da('sascgg.sugil8.xferrpt') shr';

rsubmit;
  filename xferjob 'xferjob.sas';
  libname rpt 'd:\counter\reports';
  filename frptlib 'd:\counter\reports\prod';

```

```

data _null_;
  set rpt.preport end=finish;
  file xferjob;
  if _n_ = 1 then
    put 'rsubmit;';

  /* transfer desired reports named by */
  /* name variable in preport data set */
  if (copy = 'Y') then do;
    put 'proc upload
        infile='sascgg.sugil8.'name'';
    put 'outfile=frptlib('name')
        status=no;run;';
  end;
  if finish then
    put 'endrsubmit;';
  run;

/* download file to %include at HQ */
proc download infile=xferjob
              outfile=xferrpt status=no;
run;
endrsubmit;

/* upload desired reports */
%include xferrpt;

signoff;
%end; /* do until */
%end;
%mend;

/* invoke macro to distribute reports */
%distrib;
endsubmit;

_status_ = 'H';

return;

MAIN:
return;
TERM:
return;

```

This program uses a combination of compute services and data transfer services. The DATA step to determine the reports that are being requested and to generate the PROC UPLOAD statements to transfer the reports is executed on the franchise system. This execution belongs on the franchise system because that is where the control data set used for report selection is maintained by each franchise and there is no desire to move these data sets to the mainframe system. The statements to transfer the reports are written to a text file which is then downloaded to the mainframe, invoked, and the appropriate reports are then distributed to the franchise office.

While the above implementation accomplishes the desired task, remote library services offers a simpler, more efficient solution. Consider the control data set on the local franchise workstation. The amount of information needed by Headquarters from this data set is small - the value of one variable for no more than a dozen records. The Headquarters does not need a physical copy of the data set, only some information to be used in subsequent processing. Also, the subsequent processing - creating a file with the statements for actually moving the reports to the franchise - is much better

appropriated to the mainframe to avoid having to transfer this file back to the mainframe for execution. For all of these reasons, remote library services are utilized in the second implementation of this segment of the application.

```

/*-----*/
/* Name: DISTRPT.PROGRAM */
/* */
/* This program distributes reports */
/* to the franchise offices. */
/* */
/*-----*/

length rc 8;

INIT:

submit continue;
/* set up distribution macro */
%macro distrib;

/* set up loop based on number of */
/* franchise offices */
%if %eval(&n) > 0 %then %do;
    %do i = 1 %to %eval(&n);
        %global nextfran;
        %let nextfran = &&fran&i;
        %global nextrem;
        %let nextrem = &&rem&i;
        %let i = %eval(&i+1);

options remote=&nextrem;
signon;

x 'alloc fi(xferrpt)
da('sasgg.sug18.xferrpt') shr';

rsubmit;
filename frptlib 'd:\counter\reports\prod';
endrsubmit;
/* use experimental SAS/CONNECT server */
libname rpt sase9 'd:\counter\reports'
server=&nextrem;
data _null_;
set rpt.preport end=finish;
file xferrpt;
if _n_ = 1 then
put 'rsubmit;';

/* transfer desired reports named by */
/* name variable in preport data set */
if (copy = 'Y') then do;
put 'proc upload
infile='sasgg.sug18.name'';
put 'outfile=frptlib('name')
status=no;run;';
end;
if finish then
put 'endrsubmit;';
run;
/* upload desired reports */
%include xferrpt;
signoff;
%end; /* do until */

```

```

%end;
%mend;

/* invoke macro to distribute reports */
%distrib;
endsubmit;

_status_ = 'H';

return;

MAIN:
return;
TERM:
return;

```

CONCLUSION

Distributed applications come in a variety of forms but they all have the need to access data, manage data, analyze data, and present the information obtained from the analysis. The successful implementation of a distributed application requires breaking the application down into specific segments, defining where the initial data sources reside and what information needs to be available to the user interface. Once these issues are understood, educated decisions must be made about where the data should reside and where the processing should take place for each segment of the application. The guidelines detailed in this paper will enable you to make educated decisions about implementing your application.

The SAS System provides a complete set of tools for providing you with the flexibility to implement the segments of your application to best meet your goals. Compute services, data transfer services, and now remote library services let you maximize resource usage, response time, and enduser productivity.

Remember that whether you use remote library services or data transfer services, the data must still move over the network. Be careful to program your application to optimize the balance between the cost of moving data via remote library services, the cost of moving data via data transfer services, the cost of computing on the remote host, and the cost of computing on the local host.

WHERE TO FIND MORE INFORMATION

For more information about SAS/CONNECT Software and SAS/SHARE Software:

- *SAS/CONNECT Software: Usage and Reference, Version 6, First Edition*
- *SAS Technical Report P224, SAS/CONNECT Software: Changes and Enhancements, Release 6.08*
- *SAS/SHARE Software: Usage and Reference, Version 6, First Edition*
- *Proceedings of the SAS User's Group International Seventeenth Annual Conference, "Maximizing Computer Resources with Cooperative Processing and Distributed Data Access,"* pages 641-650
- *Proceedings of the SAS User's Group International Seventeenth Annual Conference, "Database Features Extend the Scope of SAS/SHARE Software"* pages 335-340
- *Proceedings of the SAS User's Group International Eighteenth Annual Conference, "Tuning Tips for Applications That Use SAS/SHARE Software"*

SAS, SAS/CONNECT, and SAS/SHARE are registered trademarks
or trademarks of SAS Institute, Inc. in the USA and other countries.

CONTACT INFORMATION

Cheryl A. Garner
SAS Institute Inc.
SAS Campus Drive
Cary, NC 27513
Phone 919-677-8000
Fax 919-677-8123