CHAPTER 7 Examples of Combining Compute Services and Data Transfer Services

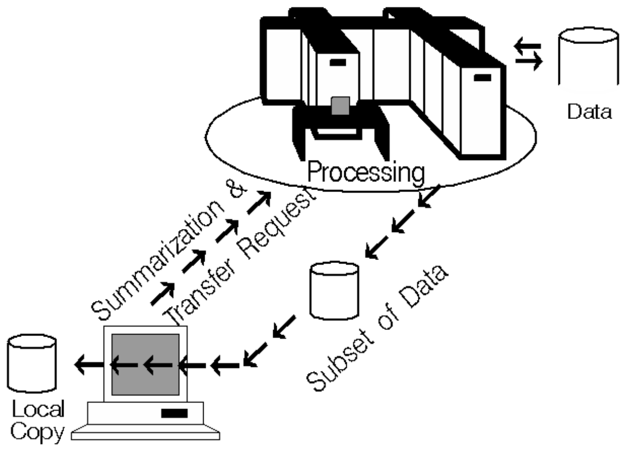
Introduction

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, then you can benefit from combining compute services and data transfer services. Reasons for not moving a copy of the data could include:

- □ the amount of data is too large.
- □ the data is frequently updated.
- □ 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. As Figure 7.1 on page 56 illustrates, compute services enable you to format and pre-process data into a subset or a 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.

Figure 7.1 Combined Compute and Data Transfer Services Processing Model



Example 1. Compute Services and Data Transfer Services Combined: Local and Remote Processing

Purpose

The SAS/CONNECT statements SIGNON, SIGNOFF, RSUBMIT, and ENDRSUBMIT enable you to submit statements to a remote host from a session on the local host. You can include these statements in a SAS program and do both local and remote processing within a single SAS program. This program can be run in an interactive line-mode SAS session, in a non-interactive SAS session, or by including the program in a session that is running on your local host. In each case, the program executes statements on both the local host and the remote host.

Program

For example, suppose that you want to perform some processing on a remote host, download the resulting SAS data set, create a permanent data set on the local host, and print a report on the local host.

The following example illustrates how to put all of these tasks into a single program.

```
    /************************************/
    /* prepare to sign on               */
    /************************************/
❶ options
    comamid=netbios
    remote=netpc;
❷ libname lhost 'c:\sales\reg1';

    /************************************/
    /* sign on and download data set    */
    /************************************/
❸ signon;
❹ rsubmit;
```

```
❺    libname rhost 'd:\dept12';
      proc sort data=rhost.master
         out=rhost.sales;
         where gross > 5000;
         by lastname dept;
      run;

❻    proc download data=rhost.sales
         out=lhost.sales;
      run;
❼ endrsubmit;

❽
      /************************************/
      /* print data set in local session   */
      /************************************/
 proc print data=lhost.sales;
 run;
```

❶ Specify the COMAMID= and the REMOTE= system options in an OPTIONS statement. These two system options define to the local session what type of link you want to establish to which remote host.

❷ Define a libref for the SAS data library on the local session where the downloaded data set should be stored.

❸ Sign on to the remote host. It is not necessary to include *remote-session-id* when you have defined the REMOTE= system option in a previous OPTIONS statement.

Note: This connection is not using a script file. △

❹ Send statements to the remote session for processing with the RSUBMIT statement. All statements are sent to the remote session until an ENDRSUBMIT statement is encountered. Although it is not necessary to include *remote-session-id*, using *remote-session-id* in the RSUBMIT statement clarifies which remote session should process a group of statements when more than one link is active. If you omit *remote-session-id*, the RSUBMIT statement submits the statements to the remote session that is most recently identified in a SIGNON or an RSUBMIT statement or a REMOTE= system option.

❺ Define the libref for the SAS data library on the remote host.

❻ The PROC DOWNLOAD step transfers the data from the library on the remote host (RHOST) to the library on the local host (LHOST).

❼ The ENDRSUBMIT statement signals the end of the block of statements that is to be submitted to the remote session. Statements following the ENDRSUBMIT statement are processed by the local session.

❽ The PROC PRINT step executes in the local session and reads the SAS data set that was downloaded in the PROC DOWNLOAD step.

Running the Program

You have several options for running this program:

- Type and submit each line in a line-mode SAS session. All of the statements between the RSUBMIT and ENDRSUBMIT statements are sent to the remote host for processing. All other statements are processed on the local host.

Note: When statements are submitted to the remote host, several statements may be grouped into a single packet of data sent to the remote host. Therefore, a line that is remote-submitted is not necessarily processed immediately after you enter it on the local host. △

- □ Build a file containing all of these statements and use a %INCLUDE statement to include the file in a line-mode session. The file is processed immediately.
- □ Build a file containing all of these statements and run a non-interactive SAS job to process the statements as follows:

  ```
  sas file-containing-program
  ```

  Note: Refer to Chapter 23, "Starting and Stopping SAS/CONNECT Software," on page 193 for more information about automatic logon scripts. △
- □ Build a file containing all of these statements and use an INCLUDE command to include the file. You must submit the included statements from the windowing environment.
- □ Build a file and issue the SUBMIT command from the Explorer window.

Example 2. Compute Services and Data Transfer Services Combined: Sorting and Merging Data

Purpose

In cases where the same remote data set needs to be manipulated by multiple local hosts, data transfer services can be used to distribute the subset of data that is needed by each local host. Each local host receives only the data it needs and uses its compute services to process that data in the local GUI. With this method, local hosts do not have to continually access the data set on the remote machine.

Program

The following SCL program fragment distributes a reservations data set from a remote host at a central office to local hosts at a number of franchise offices. The program enables distribution of reservations to a franchise office by using a WHERE statement to select the desired reservations.

```
   INIT:
   submit continue;
   signon atlanta;

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

       proc upload data=mres.reserv
          out=combine status=no;
          where origin="Atlanta";
       run;

❷      proc sort data=combine;
          by resnum;
```

```
      run;

❸     proc copy in=mres out=backup;
         select reserv;
      run;

❹     data mres.reserv;
         update mres.reserv combine;
         by resnum;
      run;
   endrsubmit;

   signoff;
   endsubmit;
```

❶ Upload all reservations for a particular location.
❷ Sort uploaded data sets for merging.
❸ Backup existing data set.
❹ Merge new and existing data sets.

Example 3. Compute Services and Data Transfer Services Combined: Macro Capabilities

Purpose

SAS/CONNECT is fully functional from within the macro facility. Both the UPLOAD and the DOWNLOAD procedures can update the macro variable SYSINFO and set it to a nonzero value when the procedure terminates due to errors.

You can also use the %SYSRPUT macro statement on the remote host to send the value of the SYSINFO macro variable back to the local SAS session. Thus, you can submit a job to the remote host and test whether a PROC UPLOAD or a PROC DOWNLOAD step has successfully completed before beginning another step on either the remote host or the local host.

Program

Suppose that you have a transaction file on your local host and you want to upload it to the remote host, and then use it to update a master file. You can test the results of the PROC UPLOAD step on the remote host by checking the value of the SYSINFO macro variable.

The SYSINFO macro variable can be used to determine if the transaction file was successfully uploaded. If successful, the master file is updated with the new information. If the upload was not successful, you receive a message that explains the problem.

You can use the %SYSRPUT macro statement to send the return code from the remote host back to the local session. Your SAS session on the local host can test the results of the upload and, if it is successful, use the DATASETS procedure to archive the transaction data set.

```
❶ libname trans 'local-SAS-data-library';
   libname backup 'local-SAS-data-library';
```

```
❷ rsubmit;
❸    proc upload data=trans.current out=current;
     run;

❹    %sysrput upload_rc=&sysinfo;
     %macro update_employee;

❺       %if &sysinfo=0 %then %do;
           libname perm 'remote-SAS-data-library';
           data perm.employee;
              update perm.employee current;
              by employee_id;
           run;
        %end;

❻       %else %put ERROR: UPLOAD of CURRENT
                  failed. Master file was
                  not updated.;
     %mend update_employee;
❼    %update_employee;
  endrsubmit;

❽ %macro check_upload;
❾    %if &upload_rc=0 %then %do;
❿       proc datasets lib=trans;
           copy out=backup;
        run;
     %end;
  %mend check_upload;
⓫ %check_upload;
```

❶ Associate a libref with the SAS data library that contains the transaction data set on the local host.

❷ Send the PROC UPLOAD statement and the UPDATE_EMPLOYEE macro to the remote host for execution.

❸ Because a single-level name for the OUT= argument is specified, the PROC UPLOAD step stores CURRENT in the default library (usually WORK) on the remote host.

❹ If the PROC UPLOAD step successfully completes, the SYSINFO macro variable is set to 0. The %SYSRPUT macro statement creates the UPLOAD_RC macro variable on the local host and puts the value that is stored in the SYSINFO macro variable into UPLOAD_RC. The UPLOAD_RC macro variable is passed to the local host and can be tested to determine if the PROC UPLOAD step was successful.

❺ Tests the SYSINFO macro variable on the remote host. If the PROC UPLOAD step is successful, the transaction data set is used to update the master data set.

❻ If the SYSINFO macro variable is not set to 0, the PROC UPLOAD step has failed, and the remote host sends messages to the SAS log (which appear in the local SAS session) notifying you that the step has failed.

❼ Executes the UPDATE_EMPLOYEE macro on the remote host.

❽ The CHECK_UPLOAD macro is executed on the local host because it follows the ENDRSUBMIT statement.

❾ Tests the value of the UPLOAD_RC macro variable created by the %SYSRPUT macro statement on the remote host to see if the PROC UPLOAD step was successful.

❿ When the transaction data set has been successfully uploaded and added to the master data set, the transaction file can be archived on the local host by using the COPY statement in the DATASETS procedure.

⓫ Executes the CHECK_UPLOAD macro on the local host.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS/ CONNECT User's Guide, Version 8*, Cary, NC: SAS Institute Inc., 1999. pp. 537.

SAS/CONNECT User's Guide, Version 8

ISBN 1–58025–477–2

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, September 1999