Data warehousing and application development using client/server computing

Veronika Horníková
Miroslav Orlovský
Investment and Development Bank, Inc.

1. Introduction

The fundamental characteristic of client/server computing is distribution of computing resources (e.g. data, compute power) across different computers. The idea is to divide applications into logical segments (tasks) so that they are then performed on platforms most appropriate.

Simple definition:
"One process (client) requesting services or data from another process (server) executing on a different machine."

2. Problems:

Barriers to information access (those result in information islands):

- heterogeneous hardware environment
- different data sources
- specialised software (hardware)

3. Reasons for using client/server:

- exploitation of centralised computing power/data capacity
  - scalability
  - performance
  - flexibility (in order to adjust to changing demands)
- GUI on desktop
- protection of
  - investment
  - strategic software
  - strategic data

Client/server provides an integrated solution.

4.
Client/server models

Most applications within information processing can be divided into three parts:

Data access: includes an access to master data or transaction data stored in different data structures

Application logic: presents a program part, which modifies and manipulates the data, executes required operations (analysis) over it

Presentation: displays results of data analysis in an easy-to-understand way for the user

According to the platform, that separate parts are realised on, we recognise following models:
⇒ distributed data access
⇒ remote data access
⇒ distributed function
⇒ remote presentation
⇒ distributed presentation

(See Figure 1)

5. SAS services for implementing client/server solution

SAS system offers wide palette of services for effective implementation of all client/server models described above:

• Remote computing services - enable to distribute application logic between local and remote system, enable computing performance of systems available in existing network to a local system - enable utilisation of remote computing resources, including hardware, software, and data, to most efficiently execute an application

• Remote Data Services - enable access to data stored in a remote environment

  Data Transfer Services (DTS) - provide a method for moving a copy of the data from one machine to another machine where a physical copy of data is created

  Remote Library Services (RLS) - enable transparent access to SAS libraries stored on remote system

• Dynamic Data Exchange (DDE) - support of data exchange with other DDE supporting applications, e.g. Lotus, Microsoft Excel ...

• Object Linking and Embedding Support - support of OLE technology.

6.
6.1. **General system characteristics**

6.1.1. **Client**

platform is a personal computer (PC - Pentium) running under MS Windows 95. Installed: SAS Base, SAS/Connect, …

All SAS code on the client executes in an interactive session.

6.1.2. **Server-I.**

platform is a personal computer (PC - Pentium) running under MS Windows NT4.0. It is used as a file server for storing application codes and some source data files, that are available to client through remote disk connecting (File and Printer Sharing for MS Networks / TCP/IP)

6.1.3. **Server-II.**

platform is DIGITAL Alpha 4100 using Open VMS operating system. Installed: SAS Base, SAS/Connect, … .

It is used as:

- Data server - by services of SAS/Connect and RLS enables access to data (SAS Dataset) stored on its disc space (Remote Data Services) for clients
- Computing server - provides client with its computing capacity (Remote Computing Services). All SAS code on the server runs also in an interactive session.

6.1.4. **Connection**

is realised by means of LAN network, (used protocol: TCP/IP). Client/server-II. connection enabling to share data and computing performance in SAS environment on different platforms is realised by SAS/Connect module.

6.2. **Pseudocode structure example:**

1. do (client executes following code)

2. do until (data is available on server platform)
   
   client instructs server to check if data is available
   
   if (data is available)
      
      then client instructions server to process the data
   
   end

3A. do until (client has available data from server)

   client requests for accessing data from server (RLS)

   end

3B. do until (client receives data from server)
client instructs server to transfer data to client
end

4 do until (client completes its process)
    client processes data
end
end

6.3. **Code description:**

1 main program loop of client application (program controls the entire application)
2 this section of code (cyclically) checks data availability on server, until data is available. SIGNON command creates connection between client and server and establishes "REMOTE SAS session" on server, that means, that there will be two SAS sessions active: one on the desktop (client or local) and one on the mainframe (server or remote). Client manages REMOTE SAS session (sends SAS statements) interactively by RSUBMIT and ENDRSUBMIT statements.
3 data processing on server was terminated and client is ready to receive data. There are two ways to access data, those are equivalent from an aspect of next processing:
   3A RLS - data remains physically on server and it is made available for client through REMOTE LIBRARY SERVICES.
   3B PROC DOWNLOAD - a copy of the data from the server is transferred to the client platform
4 data is locally processed on client

6.4.
Now I would like to present you a short description of the client/server configuration used in the Investment and Development Bank (IRB) as well as our experience with developing client/server applications and comparison of our suppositions and real results of using this way of data processing.

After first phase of the Management Information System development, its pilot run started. That brought regular loading of the large amounts of data to the data warehouse with it, that represents increase of 200 MB per decade (approximately). To store the data warehouse, ALPHA 4100 server is used. Concerning a structure of the transaction database (MUMPS), requirements on computing sources for process of loading data to data warehouse became quite extensive. It seemed to be useful to use the client/server methodology both:

- to build the data warehouse (it means to load raw data into the data warehouse while using ALPHA 4100 as a data server)

and also

- to develop applications over this data warehouse.

In IRB a.s. we chose model of client/server with distributed function (See Figure 1). It means, that:

- extensive computing processes working with big volume of data are executed on server. Those are processes related with regular processing of data from the transaction database and updating the data warehouse. Client/server processing meant in this case a utilisation of powerful processor of the server and at the same time an elimination of data transfer on network. In practice, it means reduction of time requirements of this processing and also respecting limited storage capabilities of the local machines (clients).

- (on the other side) client manages this processing and provides with comfortable graphic user interface (GUI).

### 6.4.1. Building data warehouse

As mentioned above, regular loading of data files from transaction database to the data warehouse represents increase of approximately 200 MB per decade (Concerning our storage capacities we decided to update our data warehouse (to load new data) once in a decade). The need of processing such the large amounts of data brought us naturally to using of Remote Computing Services, because the most raw data (as well as the data warehouse) is stored on ALPHA server. So, from client machine the remote SAS session (and data processing in it) is started. In this phase we don’t need Data Transfer Services because the results of this processing (all datasets, and also base and summary tables necessary for applications) are also stored on ALPHA server. Using remote processing of data brought us two benefits: (1.) exclusion of network errors related with data transfer and (2.) remarkable reduction of time requirements of data processing (it changed from minutes to seconds).

By processing the raw data, those are not stored on ALPHA server but on NT (file) server (e.g. some marketing information; data, that are not part of the transaction system), we use Data Transfer Services for transfer the results of this processing (which is realised locally) from local machine to server to integrate them into the data warehouse.
6.4.2. Application development

Except the regular data processing at updating the data warehouse, there are, of course, also user applications within Management Information System built on client/server architecture. The purpose is to utilise the server’s compute power at extensive analytic calculations. User applications are constructed as a combination of Remote Computing Services and Data Transfer Services (See Figure 3):

◆ data preparation (summarisation) is realised on ALPHA server (RSUBMIT and ENDRSUBMIT statements delimit program blocks that are to be passed to the REMOTE SAS session for execution on the remote host)
  ⇒ working with large data tables - Remote Computing Services

◆ then prepared data (summary tables) are transferred to client (PROC DOWNLOAD statement)
  ⇒ size of these tables is considerably reduced unlike original data - Data Transfer Services

◆ next processing of summarised data and graphic presentation is realised on client
  ⇒ working with relatively small table

We have two types of applications within our Management Information System: SAS/AF applications and SAS/EIS Motore applications (SAS/EIS Motore module is the extension of SAS/EIS module). While in SAS/AF applications code, that includes Remote Computing Services and Data Transfer Services must be written by programmer, in SAS/EIS Motore applications client/server computing is supplied by Motore code customisation.

MOTOPEN:

method;
   /* original code of the method */
   /* Client-server setup */
   if not rlink('ALPHA') then do;
      submit continue;
         options remote=ALPHA comamid=tcp;
         filename rlink '!SASROOT\connect\saslink\tcpvms.scr';
         signon;
      endsubbmit;
   if not rlink('ALPHA') then do;
      put '** Error On Remote Session SignOn';
   end do; /* rlink ('ALPHA') exists */
   /* ** Assign data and format libraries. ** Assigning libraries for remote SAS session ** can be done in autoexec.sas instead ** Formats will be needed when performing dynamic summarisation. */
   submit remote continue;
      libname DIRDAT 'path_for_library_DIRDAT_on_server';
      libname LIBRARY 'path_for_library_LIBRARY_on_server';
   endsubbmit;
   end;
/* Client-server setup */
MOTCLOSE:
method;
/* original code of the method */

/* Client-server setup*/
if rlink ('ALPHA') then
   call execmd('signoff'); /* closing connection */
/***
put here code to update PC data set
with refresh date,
save logaccess data set
...
***/
/* Client-server setup*/
endmethod;

MOTDBRD:

/* original code of the method */

/* simple example of remote summary and downloading the result */
/* it is only a short fragment of code (there is much more customisation in Motore code */)*/
if rem_summary then do; /* rem_summary flag is set in previous code, it indicates remote data processing */
   /* remote summary */
   submit;
      rsubmit;
   endsubmit;
end;
submit; /* code shared between local and remote */
   proc means noprint nway data=&suggdsn;
      where &wclstype
   endsubmit;
do i=1 to listlen(lwhere);
   wcls = getitemc(lwhere,i);
   submit;
      &wcls /* additional where clause */
   endsubmit;
end;
submit;
   &wclscrty
   &wclxcept;
   class &classe;
   output out=&newfile sum=;
   run;
endsubmit;

/* If remote, copy result */
if rem_summary then do;
   submit continue;
      proc download data=&newfile out=&newfile status=n;
   run;
endrsubmit;
endsubmit;
end;
/*else local summary*/
else do;
submit continue;
endsubmit;
end;
end;
endmethod;

Then SAS/EIS Motore generates appropriate code. In both, SAS/EIS Motore applications and SAS/AF application, the code is very similar to following:

**example:**

/* Connection with ALPHA server as SAS user - at the beginning of processing */
options comamid=tcp remote=ALPHA;
filename rlink 'C:sas\connect\saslink\tcpvms.scr'; /* Windows --> Open VMS */
signon;

/* following code is generated by Motore */

rsubmit;
proc means noprint nway data=LIB.SUM1; /* creating special summary tables and selections */
where _type_ =9 and VNDATE=13777; /* on server */
class U5 VNDATE;
output out=WORK._0036499 sum=;
run;
proc download data=WORK._0036499 out=WORK._0036499 status=n;
run;
WORK library WORK library
on server on client

rsubmit;
proc means noprint nway data=LIB.SUM2;
where _type_ =57 and U4=110 and U5=1390 and VNDATE=13777;
class U3 U4 U5 VNDATE;
output out=WORK._0036512 sum=;
run;
proc download data=WORK._0036512 out=WORK._0036512 status=n;
run;

endrsubmit;
/* HERE A LOCAL PROCESSING OF SUMMARY TABLES FOLLOWS */
As you see, extensive computing processes working with source data are executed on server in remote SAS session and then the results are downloaded to the client machine that manages their next processing/presentation. This approach satisfied our suppositions: brought us network reliability (thanks to transferring relatively small data tables) as well as considerable acceleration of applications - now the response time is mainly depending on SAS capability to display different graphic objects in applications and consequently on power of the client machine, because graphic presentation is realised on the side of client.

7. Conclusion

That was a short presentation of implementing the client/server computing in practice using the SAS system services. Our suppositions concerned mainly the increasing performance for users and also the minimizing local disk space requirements. We tried to achieve that by using Remote Computing Services to process the data at the server and Data Transfer Services to download the summary tables or data subsets for local processing. As we exploited the remote computer, that is more powerful than local PC, not only as a file server but also as a SAS application server, we succeed to decrease the response time in most applications significantly.

References

[1] Client/server Computing with the SAS® System Tips & Techniques
[2] Building a Data Warehouse Using the SAS® System

Authors

Veronika Horníková
Investičná a rozvojová banka, a.s.
Štúrova 5
813 54 Bratislava
Slovak Republic
Phone: +421 7 389 5228 Fax: +421 7 323 048
E-mail: caudtova@.brc020.irb.sk

Miroslav Orlovský
Investičná a rozvojová banka, a.s.
Štúrova 5
813 54 Bratislava
Slovak Republic
Phone: +421 7 389 5228 Fax: +421 7 323 048
E-mail: orlovsky@irb.sk

1 The registered name of the IRB bank is: Investičná a rozvojová banka, a.s., Slovak Republic

© SAS, SAS/AF, SAS/EIS, SAS/EIS Motore are registered trademarks of SAS Institute Inc., Cary, NC, USA
Figure 2  Client / Server Architecture

PC
Windows 95
clients

Servers

PC
Windows NT4.0
server I.

Alpha 4100
Open VMS
server II.

disc space 12 GB
Figure 3

User applications
Combination of \textit{Remote Compute Services} and \textit{Data Transfer Services}

\begin{itemize}
  \item \textbf{Request} for data summarisation and transfer
  \item \textbf{Transfer of the results:} the summary data (subset of original data)
  \item \textbf{Local copy of the data}
\end{itemize}