The SAS® System in a Client/Server Environment
Bengt Bengtsson
SAS Institute European Head Office, Heidelberg, Germany

Abstract
Client/server computing is understood in many different ways, ranging from being the revolutionary new model for computing in the 90's, to simply a buzzword to which we all must pay lip-service. In this paper we will attempt to describe different kinds of client/server applications, both by examining the current market offerings, and by taking a more abstract view of the way in which clients and servers could interact. This abstract view looks at five styles of client/server applications, as defined by the Gartner Group, following their research into the use of client/server technology by a number of user organisations. The five styles are:

- distributed data access
- remote data access
- distributed processing
- remote presentation
- distributed presentation

A part of the paper will be a practical demonstration of applications fitting all five styles, with a discussion of situations where each of the styles is appropriate. The demonstration will feature the SAS System running under Windows 3.1 and HP-UX 9.01.

Introduction
Each larger organisation of today has invested in hardware, software and human resources to achieve an efficient computing environment. In order to maximise the return of these investments the industry trend is to go to distributed computing environments providing user-friendly interfaces on client computers and central resources such as data storage on central servers.

Hardware Resources
Traditionally, when discussing performance, we think of hardware resources. Hardware resources include CPU, external storage and network resources. Usually the primary CPU resource of concern is processor cycles, but often memory consumption is critical too. The network itself is often the bottleneck in many client/server applications. Besides the bandwidth there is the capacity of network equipment such as communication controllers, gateways, bridges, routers and other network components to take into consideration. Access to special hardware such as vector processors or special peripheral units, e.g. plotters, are often critical for business applications. We can see how important it is that our client/server solution supports various kinds of hardware so we can maximise the returns of all our hardware investments. It is also important that our client/server solution is open for the future, so that we can take advantage of new achievements in the field of computer and network technology.

Software Resources
There are three main kinds of software: systems software, business applications systems and data. By systems software we usually mean the operating system itself, but tools used to develop the second kind of software, the business applications, must also be considered. A third thing to remember, and that never should be underestimated, is the data itself. Often the value of the data by far exceeds the value of all the hardware and other software together - if it can be turned into useful information. It is therefore very important that we have access to all the information that we have collected over the years, as well as to new data coming in.

Human Resources
Time is money, hence our client/server solution must make it possible for our staff to make the best use of their time. By giving the end-user an easy-to-use working environment he or she can concentrate on the business work instead of worrying about how to use the computer. It is also important that tools used to develop client/server solutions provide the same interface for our developers on all platform involved, so that they have to learn only one tool, instead of one tool for each platform. And, of course, the applications developed must be easy to maintain.

Client/Server Market Today
The expression client/server can often be seen in the press and in adverts today. We can easily identify three types of client/server products on the market.

Distributed Databases
The client application issues SQL statements and the client/server product materialises data from elsewhere in the network. This is the most
common type of client/server solutions on the market and the kind of client/server computing that is provided by most modern relational database vendors. The advantage of this approach is obvious. Data access is seamless. The application on the client side is unconcerned with the mechanics of getting the data elsewhere. The disadvantage however, is that the solution almost only deals with data. This can lead to an unacceptable load on the network and the client computer, since, in almost all cases, all the data has to be moved across the network to the client computer where the application runs. And of course, we still need a tool for turning all that data into information.

Client/Server Extension Toolkits

There are programming tools available on the market which make it easier to write applications where the client can talk to an application running on a server. Usually these are 3GL APIs which allow the client to ask for services. The advantage is the control the programmer has over the application. Basically any service can be requested from the server. Unfortunately the drawbacks can't be neglected. Since this solution involves programming using sockets, RPCs (Remote Programming Calls) or one of a variety of very low-level networking protocols, it means that the high cost of low-level programming remains, with a long development cycle and high maintenance costs. It also means that the applications written can't be easily moved to new platforms as these arise in the organisation. For each new system, a new way of programming, and a new set of tools, need to be learned. All this leads to an ever growing backlog of projects to be implemented.

Packaged Applications

The third sort of client/server solutions we can find on the market today are the "special solutions", the packaged applications. Here all the client/server work is hidden from the end-user. The advantage of this is of course that it is easy to use. Unfortunately, the user has almost no control of what his client/server application is doing and how it is done. The lack of flexibility is striking. Often these solutions are very partial solutions, e.g. some of the reports specifications available in the server system can be transferred to the client platform, but some can not. Usually there is one client system talking to another server system - it is not the same system running on two or more different platforms. This makes it hard, or impossible, to move parts of the application to the server or to the client when needed.

DIFFERENT MODELS OF CLIENT/SERVER COMPUTING

Let us take a closer look at different ways of implementing client/server applications.

The Gartner Group

The Gartner Group surveyed different types of client/server applications and came up with five different categories. Basically any application can be divided into three functions: data access; application logic; and presentation. The five models defined by the Gartner Group differ in the way these functions are distributed between the client and the server. All of these models have their advantages and disadvantages.

Distributed Data Access

In this case data is distributed in the network and handled by data servers. All logic and presentation take place on the client side. This means that all data involved in the application has to move to the client side before being processed (limited processing e.g. SQL queries can still take part in the data server). This is the type of client/server computing provided by most modern RDBM systems today.

Remote Data Access

All data is stored on a data server. All logic and presentation take place on the client side.

Distributed Function

Data is stored on the server side. A part of the application logic is handled by the server, another part by the client. This allows us to off-load the server CPU, while not moving unnecessary data across the network. The benefit of this approach is that we can take advantage of the hardware and software available on the server platform by using it as a compute server, hereby only having to move the results of the server processing for further processing on the client...
side. This is particularly useful when the resources on the server platform are better suited to handle the task than the client, or when the volume of the data to be processed is too large to be moved across the network.

Remote Presentation
All data is stored on the server and all application logic is handled by the server. Only the presentation of the results are handled by the client computer. This lets us use desktop GUIs for presenting the information to the user, but does not help us offloading the CPU significantly since most of the work still is done on the server.

Distributed Presentation
This model differs from the "remote presentation" in that the presentation is split between the server and the client.

A Demonstration
An umbrella application, containing five business applications, each using one of the five models of client/server computing, is demonstrated. All parts of the application are developed using the SAS System. This implementation uses Windows 3.1 as the client platform and HP-UX 9.01 as the server platform. Communication protocol used is TCP/IP. (Other platforms include e.g. MVS, CMS, VSE, OpenVMS VAX, OpenVMS AXP, OS/2, Windows NT and a variety of UNIX platforms. Other communication protocols supported include APPC, DECnet and NETBEIOS)

The Complete Solution
While a number of vendors offer client/server tools on the market today, these solutions are only partial solutions. As they only support one sort of client/server computing, the risk of being locked into a single strategy, without flexibility to meet more complex needs, is large. The SAS System offers a complete solution for client/server computing. Not only are all models of client/server computing supported; data servers, compute servers and presentation servers can be exploited so that each part of the computing environment can be used for what it is best at. Since the SAS System uses exactly the same syntax on all supported platforms, the boundary between application segments run on the client and segments run on the server can be moved by just letting the code execute on the another CPU. The ability to implement intelligent client/server applications where one part of the application can run on one platform today and on another platform tomorrow is critical for the success of applications such as decision support systems and EIS applications which often include analysing large amounts of data in an inhomogenous environment.