SAS Client-Server Development: Taking it to the Web

Eric Brinsfield, Meridian Software, Inc.

Henrietta Cummings, Meridian Software, Inc.

ABSTRACT

As a sequel to the presentation "SAS Client-Server Development: Tricks of the Trade", we will expand on the discussion of client-server development as it relates to the Internet. Most Web-based applications currently utilize a thin-client model. Although this architecture is significantly different from most client-server applications, it is still client-server.

We will briefly discuss the pros and cons of thin-client application design. While clarifying how SAS® software fits into the puzzle, we will explain and demonstrate thin-client characteristics and speculate on future directions.

INTRODUCTION

Over the past few years, the increasing use and awareness of the Internet has stimulated many new ideas, technologies, and businesses. As a result, competition in the software industry has exploded, creating a world of constantly changing technologies. Client-server technology has been swept along with the wave.

In this paper, we will explain thin-client technology as it relates to client-server development and SAS software. We will also discuss how it has evolved recently and where we think it will go. Our presentation will include demonstrations that are not discussed in this paper.

DEFINITIONS

Client-server application

Generally, a client-server application provides a graphical user interface that runs on a PC or workstation that accesses and processes data on a separate server running the same or different operating system. Using this architecture, the user gets a nice user interface, but shares a centralized and sharable image of the data and benefits from extra power on the server or servers.

The Web, itself, is a simple two-tier application that is considered the world's largest client-server application.

Fat client

The fat client represents the traditional client-server architecture. With this design, the client workstation houses ample processing power and disk space so that much of the processing and most of the application code actually reside on the client side. The server provides access to shared data.

The fat client often contains more functionality than is actually needed by each member of the user base. It also has problems scaling with growing user numbers and data volume. If a small part of the business logic needs to change in the fat client, you often need to test it in its entirety and redistribute it to the entire user base.

Most IS departments circumvent this problem by placing the executable code on a file server, such as a Netware server. For example, we frequently see SAS client-server systems that use a Unix application server with Windows 95 workstation clients, but the SAS/AF catalogs and additional source code are stored on a Novell file server.

This centralizes the application code, thereby reducing maintenance headaches, but performance can suffer. Unless your throughput on the LAN is extremely fast, SAS executables and SAS/AF catalog entries typically load more slowly across the LAN than most users would like.

As a result, if your user base is small and maintenance is not an issue, store the application code on the individual workstation for better performance. You can, however, place logic in your application to synchronize the user image with a central copy, thereby solving the maintenance problem. This does complicate the logic somewhat, but it is not extremely different from the caching you experience in Web applications.

Thin client

In a thin-client design, the client workstation requires minimal disk storage capacity. In addition, most of the processing is pushed over to the server side. The thin client resembles a dumb terminal with a graphical user interface.

Browsers, such as Internet Explorer© and Netscape Navigator™, are examples of applications that support a thin-client design. The only software that must be installed on the workstation, besides the operating system, is the browser itself.

ADVANTAGES OF THIN-CLIENT ARCHITECTURE

- Cheaper workstations

  With reduced memory, disk storage, and processing requirements, the cost of the individual workstations is obviously lower. This led to the invention of the poorly accepted NetPC, which was designed to be cheap and rely totally on the Internet or an intranet.

  Alternatively, we will probably just find most users operating on Windows with less power, especially now that Microsoft has released the Windows Terminal Server and soon Citrix System, Inc. will release MeteFrame. These products remove the workload from the client side.

- Reduced cost of client-side software

  If all of the application software resides on the server side, you may often find the cost of software per person much more affordable. Realistically, however, software vendors will probably recover their losses by increasing the prices of server-side software.

- Easier maintenance of client workstations

  One of the most appealing aspects of the thin-client design is the centralized nature of the application software. Updates and corrections need only be applied to the server image and not on every individual workstation, a process which is extremely time consuming and expensive. This also ensures all users stay in synch and always have the latest updates and corrections.

- Simplicity

  Because we can shift most of the processing to one side of the client-server equation, the net effect is simplification. Although proper design is still essential, the result is usually simpler.

- Common, friendly graphical user interface

  Almost everyone is now familiar with at least one flavor of browser. As a result, if you can deliver an application that runs in a browser, the training time is significantly reduced. In addition, everyone is happier with the new application sooner.

DISADVANTAGES OF THIN CLIENT

- Limited functionality (temporary)

  Originally thin clients provided rather limited functionality, but the power of browser applications is increasing daily. Much of the improvement results from the advent of middleware, such as Microsoft's Transaction Server and SAS Institute's SAS/IntrNet software.

  SAS programmers will soon, if they have not already, find themselves programming in JAVA on the client side and SAS on the server side. Whether JAVA becomes the standard or not, it is clear that, as more capability is bred into languages that support the thin-client design, more companies will want to implement systems using a common language that provides a standard look and feel.

- Slow transfer rates

  We have all experienced the long wait for a Web page to finish downloading from the Internet. We know from personal experience that the performance of analog modems, although much improved since the old 300 BAUD days, still imposes a frustrating bottleneck on Web applications.

  Currently, limited bandwidth is one of the biggest disadvantages of thin-client architecture, but two major efforts promise to alleviate the pain. First, is increase bandwidth, and second, is middleware.

Almost every large company in the world is hoping to offer unlimited bandwidth to both business and home users. Do not sign any long-term contracts for communication services now, because new communication pathways are just around the corner. Although options such as ISDN, T1, HDSL, ADSL, and cable modems exist today, prices and availability limit the use of the fastest technologies. This will change soon.

Middleware is evolving to perform much of the work that fat clients used to handle in classic client-server applications. Using this technique reduces the amount of data transferred across the network. So with middleware improvements, the negative impact of limited bandwidth is lessened.

- Reduced processing power on client workstation

Less powerful workstations limit your options for standalone processing. As a result, most advanced users will not accept the absolute thin-client solution. We do not believe this will ever be a problem, because processing power has become so affordable that reducing processing power, even on a thin client, does not make sense.

- Limited disk space on client workstation

The classic "fat" client equates to a big footprint. The application requires a significant amount of disk space. Users also want disk space for their own use. In every application we have designed and implemented, users always want to export, save, or archive something to their personal storage area, so they can go back to it later.

But, IT has already dealt with this by defining personal home directories on the network file server. As a result, a thin client may not appear to be thin when considering disk storage options.

- Competition for centralized resources

As thin-client applications spread, more and more of application processing will be pushed off to the server side. As a result, servers will become overloaded more quickly, acting much like the mainframes of our past. Users may start longing for the old fat-client days again, if budgets limit the upgrades of servers.

What really led to the decline of the mainframe central processing model? For many people, it was the lack of a reasonable graphical user interface. Why would anyone want to type a letter on a mainframe terminal when you have today's PC-based word processors?

The other reason was freedom. Many productive people felt constrained by waiting for MIS to develop new software systems. Users took control of their own life when they started moving to PCs. Are we heading back to bottlenecks again?

Possibly, but unlikely. The competition in the Web market is fierce. New applications are announced on a daily basis. If your needs are fairly standard, you should be able to find a solution that is already built. If you need a custom solution and you are willing to outsource, experts will gladly jump to your rescue. Times have changed.

- Connection required

Finally thin-client architecture relies on a connection to the server. For most corporate users on an intranet, this is not a problem, unless the network goes down. But, as more workers telecommute, a constant connection could be costly, if not prohibitive. Although the competition to provide Internet access is fierce, the future costs of connectivity are still uncertain.

For example, some telephone companies now charge by the minute for ISDN connect time in addition to the monthly service fee.

Also, looping back to the first problem of bandwidth, if thin client spreads and everyone is connected constantly, the load on the connection will increase and bandwidth becomes an issue again. Bandwidth is to the year 2000 as memory was to the year 1990. The more you give us, the more we will use, take advantage of, and want.

SAS SOFTWARE IN A FAT-CLIENT APPLICATION OVER THE INTERNET

So where does SAS fit into the picture? For many years, SAS/CONNECT software has provided developers with the capability to develop middleware to enhance client-server solutions. Using SAS/CONNECT, SAS developers can control where the processing occurs and where data resides. For more details refer to the Host Section presentation entitled *SAS Client-server Development: Tricks of the Trade*, which discusses and demonstrates a three-tier client-server design using SAS/CONNECT.

Because SAS/CONNECT not only supports, but also excels using the TCP/IP protocol, which is used for Internet communication, you can run a SAS/CONNECT fat client-server application over the Internet. As long as you can establish a link between the client and the server, SAS/CONNECT will operate as if the server was on your LAN. This functionality was available before the advent of SAS/IntrNet and is more a result of the support for TCP/IP than anything else.

The problem is, of course, that fat-client applications require the licensing of SAS software on both sides. Also, because of firewalls and dynamically assigned IP addresses, you typically cannot just connect to a remote server on the Internet. Clearly, the SAS-based fat client is not a viable solution for the typical Web user.

SAS SOFTWARE AS A THIN-CLIENT-SERVER ON THE INTERNET

SAS/IntrNet software provides developers with the capability to create truly thin-client applications that access the power of SAS. Among other facilities within SAS/IntrNet, JConnect and JTunnel enable Java applications and applets to connect to a remote SAS session and perform remote processing on the middle tier.

JConnect is a set of Java classes that can be called from within Java applets to communicate with SAS software on a server. JConnect provides functionality similar to that of SAS/CONNECT software to Java applets (and applications). A major advantage is that JConnect does not require SAS software to be installed on the client.

JConnect classes include functionality to start and shut down a remote SAS/CONNECT server. SAS statements can be sent to the server for execution via these classes. The results of the SAS statements can be text output, SAS data sets or views, graphics output, or ASCII text from the SAS log or SAS output. The graphics output includes both GIF images and VRML (Virtual Reality Markup Language) files that produce clickable graphs. Any of the results of the SAS statements can be retrieved and displayed by the Java client.

When SAS data sets are created as the result of the SAS statements executed, they can be accessed from the Java client via a JDBC Connection object. The JConnect classes include methods to retrieve the JDBC Connection object.

Access to the SAS server from the Web is through a socket connection. Most firewalls allow only HTTP traffic through the firewall. Because of this, users behind a firewall are not able to access the SAS server. A second problem in accessing the SAS server through SAS/IntrNet is caused by the Java security manager provided by the browser, which usually only permits unsigned applets to establish socket connections to the host from which they were downloaded. This requirement means the SAS server has to run on the same host as the Web server.

To solve the two problems described above, SAS Institute provides a JTunnel feature. JTunnel provides a CGI (Common Gateway Interface) installed on the Web server. This allows the Java client to send HTTP requests to the Web server. The Web server then forwards the requests to the SAS server via the JTunnel CGI.

A copy of the JTunnel CGI starts with every JTunnel request. It is important to monitor this, because an increasing number of JTunnel requests can degrade performance. Through JTunnel, however, the SAS server is not required to run on the same host as the Web server and Java clients can execute SAS statements even from behind firewalls.

Client-Server Java, RMI, CORBA, and the Web

We've been discussing interfacing with the JConnect classes from within a Java client. To keep the Java client truly thin and to enhance performance there are alternatives. One alternative is to distribute the application by placing only the objects necessary for the user interface in the Java client code. Place the objects that interface with JConnect in a Java application on the Web server and provide client to server communication.

Since both the client objects and the server objects in this case are Java objects, RMI (JavaSoft's Remote Method Invocation) can be used for the client to server communication. The distributed objects will need additional exception handling. RMI extends the Java exception classes to provide the additional exception handling that is needed.

Another alternative for the client-to-server communication is CORBA (Common Object Request Broker Architecture). CORBA is a widely accepted industry standard from the Object Management Group. There are now many vendors that sell Object Request Brokers (ORBs) that

follow the CORBA standard. These provide the functionality that enables distributed computing for the client objects and the server objects. For thin client development, CORBA is more difficult to use than RMI, but CORBA provides a more complete and scalable distributed computing environment than RMI.

SUMMARY

The Internet has become essential to business and to the lives of many individuals. As a result, software vendors who do not fit within the framework of the Internet will cease to exist. SAS Institute has adapted and now provides many tools that release the analytical strength of SAS software to all Internet users.

With the importance of the Internet and intranets, SAS client-server developers need to learn new skills and reshape their client-server designs to utilize SAS/IntrNet. Good object oriented design practices are more important than ever. As a client-server developer, you must evaluate where the processing should occur and reduce transfer of data across the network.

From the client-server perspective, the advent of the Internet led to the rise of the thin client and the three-tier architecture. Although they have been in use for some time, three-tier systems will make thin clients viable for more complex applications, especially when using SAS software.

We believe that thin-client features are beneficial and will persist, but we do not believe that hardware will necessarily remain thin forever. Innovations in client-server designs will offer more capability, which will perform best with more powerful clients. Although application software that is destined to run on the client will continue to originate on the server, bigger processors and more memory will always provide more satisfaction for the user.

REFERENCE INFORMATION

Eric Brinsfield, President
email: merecb@meridian-software.com

Henrietta Cummings, Senior Systems Designer
Email: merhhc@meridian-software.com

Meridian Software, Inc.
12204 Old Creedmoor Road
Raleigh, NC 27613

Phone: 919.518.1070
FAX: 919.518.1170
WWW: www.meridian-software.com

BIBLIOGRAPHY

"*Optimizing Client-Server Performance through Data Warehouse Design*", Eric C. Brinsfield, Proceedings of the SESUG97 Conference, 1997, Jacksonville, FL

ACKNOWLEDGEMENTS

Cathy Brinsfield of Meridian Software for editing and formatting.

TRADEMARKS