ABSTRACT
One of the four fundamental barriers to successful enterprise-wide information delivery is the incompatibility of various hardware and operating system environments. Customers with enterprise-wide information needs must be able to leverage their investment in technology, both old and new. Many terms are used in the field to describe the strategies available to customers including client/server, distributed processing, and downsizing. This paper will examine models of cooperative processing, review the use of SAS System tools—in particular SAS/CONNECT® and SAS/ACCESS® software—to accomplish the integration of resources, and suggest some guidelines for the distribution of application tasks.

INTRODUCTION
Cooperative processing, as a style of computing, commands a considerable amount of attention in MIS installations around the world. There is no lack of material written to describe the hardware and software options available to the intrepid MIS professional as he or she launches their enterprise into this brave new world. Despite this plethora of available tools, many cooperative processing or client/server implementations fail. The beginning of this paper will be devoted to examining a scheme for establishing, ahead of time, some important logical definitions. Using these definitions, then, some strategies for dividing the work of applications across available computing platforms will be examined.

APPLICATION SEGMENTATION
To sensibly begin the task of implementing a client/server application, it is critical to review the three major tasks of data driven applications (See Figure 1).

The task of data access and management involves all of the routines necessary to locate and acquire data for use by an application (See Figure 2).

Figure 2

Logical Tasks in Data Driven Applications

- Data Management
  - Locate and acquire data
  - Apply business rules
  - Statistical analysis
- Application Logic
- Presentation

These routines can involve formatting routines designed to gain access to legacy data stored in system formats as well as SQL routines to get data from relational database systems. It is also likely that some amount of re-shaping of data will be necessary to actually prepare the raw data elements for use in applications. This is also an opportunity to combine files from different systems and to create new, more meaningful data sources.

Application logic tasks involve applying routines reflective of business needs to detail data, to provide additional meaning (See Figure 3).

Figure 3

Logical Tasks in Data Driven Applications

Tasks Include:
- Apply business rules
- Statistical analysis

Usually, that means taking transaction level data and ‘rolling it up’ or summarizing it to higher levels of aggregation. These summaries might include adding totals for geographic areas or time periods (e.g., totals for regions or months). This task would also include applying well-known statistical routines to data to uncover relationships or exceptions.

Without first being clear about the structure of applications, a cooperative processing scheme will be made more difficult to implement. The reason for this is that cooperative processing, by definition, will include multiple processors sharing the job of executing an application. The best use of all available computing platforms in such an environment, each being used for their particular strengths, is a key element in successful implementation of a cooperative processing scheme. These data driven elements include the data access and management tasks, the application logic tasks, and the presentation tasks.
Finally, presentation tasks are those routines which take the results of the application logic routines and offer them to final clients (See Figure 4).

Figure 4

Logical Tasks in Data Driven Applications

This includes all the standard forms of presentation like tabular reports, business graphs, and spreadsheets. Recently, this layer of routines has included the more interactive presentation techniques associated with EIS packages, offering exploration through data summaries in a graphical environment.

GARTNER GROUP COOPERATIVE PROCESSING STYLES

The Gartner Group, a well respected analysis firm, uses a similar logical outline in its description of cooperative processing. There are some differences in the definition of Data Access and Management layer (i.e. Gartner locates access to DBMS data in the Application Logic layer). However, fundamentally the major division points of a data driven application are consistent with those listed above.

In an effort to describe how such applications can be implemented in a multiple platform environment, Gartner defines five styles of cooperative processing (See Figure 5).

Figure 5

These five styles show differing strategies for dividing the work of the three application segments across multiple computing platforms. The styles are characterized by where the division of function is made. That is, application programmers have some choice as to which layers of an application they choose to 'split up' across environments. In most cases, whole segments (e.g. Data Management) are targeted for a single platform. Thus assigning whole application segments to a single platform is the starting point for describing the different styles. In some cases, the divisions are better made within a layer (i.e. splitting the application logic across platforms). Also, as one moves from left to right in the diagram (or follow the network division from lower left to upper right), more of the application processing is being located out on the remote, or user workstation platform.

Further information can be obtained about the specifics of these strategies from the Gartner Group. This paper will examine just three of these levels, and concentrate on one, as an example of the type of implementation most SAS community users will find useful in their work. The three styles reviewed here include Remote Presentation, Distributed Function, and Remote Data Management (See Figure 6).

Remote Presentation

This style of cooperative processing keeps the Data Management and Application Logic layers of an application together on one platform, with the Presentation layer split off to a remote platform. According to Gartner, this division of labor is best suited for single message, non-conversational style transactions. In this scheme, the end-user entry point is located on the workstation, with single requests sent back to a server processor. Any application logic is executed on the server processor, with any subsequent data management (e.g. updating a database) accomplished here as well. An application employing a local editing session, sending messages or transactions back to a server processor, could be constructed using SAS FSEDIT on a local workstation, with SAS/CONNECT software to move the input data to a server for further validation and/or combination with other data.

While this style of client/server processing makes good use of the GUI capabilities of a workstation, it does little to relieve the server processor of work.

Remote Data Management

This style of cooperative processing locates the Presentation and Application Logic layers on a single platform, and keeps the Data Management layer on the server platform. This is the classic division suggested when designers ask to have, 'all of my data on the mainframe, but do all of my processing on the PC'. This particular model is also the one which, at first glance, offers the best opportunity for the savings in processor cost so coveted by MIS organizations. By moving processing out to cheaper machines, enterprises hope to gain significant economies.

This style of cooperative processing is best suited for 'client/server' database access. That is, applications which can either capture data at the workstation, pre-process it, then send it to a server data base, or retrieve data from a server database and apply analytic and presentation logic at the workstation level. These could be characterized as, in the first case, transaction oriented, and in the latter case, decision support. Using the SAS System's tools, an application can easily be built which allows for the capture and validation of data at the workstation level before shipping it off to a server data base. Using SAS/ACCESS software, data stored in popular data base systems could then be updated. A decision support application which takes data from a server and applies analytic and presentation logic at the workstation side can also be
easily built with SAS software facilities. It is in exploring this second feature of the remote data access style, however, that we encounter a potential roadblock. Because, while it is certainly reasonable to build a decision support application which locates all analytic and presentation processing on a workstation, the scale of the data which might need to be delivered to the workstation will introduce a new, unanticipated cost. The movement of large amounts of data can both overtax available network resources, and overwhelm local computing resources. Thus, a model which more finely separates some of the application logic is of interest.

Distributed function
This style of cooperative processing makes the difficult choice of splitting the application tasks across platforms, by actually going inside one of the three application segments. In this case, the application logic segment is distributed across platforms. This means that some of the work necessary to accomplish an application should be executed on the server platform, before data are delivered to the workstation. Garthner indicates that this is 'the most difficult style of cooperative processing (applications) to design and develop, since there are two separately compiled application programs'. This difficulty is certainly validated by the small amount of coverage in the trade press of vendors which supply tools to accomplish this 'most difficult' of cooperative processing styles.

In the remainder of this paper, the capabilities needed to implement applications which distribute functions across platforms will be reviewed, with particular emphasis on the available SAS capabilities.

TOOLS NEEDED FOR DISTRIBUTED FUNCTION
To build and implement applications which can take advantage of the computing capabilities of all platforms in a distributed environment, there are four basic tool sets necessary. These four tools include data access capabilities, connectivity links, relocatable executions, and an integrated GUI/application development environment.

Access to data
All applications need access to data. In the majority of tool sets offered for client/server application development, this is made possible through relational DBMS's. Of course, not all enterprise data have been moved into relational databases yet, so a comprehensive client/server software tool set should offer access to data in legacy forms like VSAM, IMS or sequential file formats. The SAS System tools meet both of these needs.

Connectivity links
No distributed application development can be accomplished without good communication links to all platforms. In the program-to-program world, this means the ability to not only transfer data, but establish conversations with all remote resources. All available client/server tool sets offer communication software to enable messages to be passed amongst the cooperating platforms. The SAS support for popular program-to-program protocols like APPC for IBM platforms, DECnet from Digital Equipment, NetBIOS for connectivity in LANs, and TCP/IP, enables nodes in the distributed configuration to be treated as peers, rather than pre-determining the client/server relationship.

Relocatable executions
This is the key capability which enables distributed function cooperative processing to be implemented. Relocatable executions means the ability to develop application logic portions, independent of the prior knowledge of the final target execution platform. So, an application developer should be able to choose to modularize the application function tasks, then install them on the platforms of choice. This ability should be dynamic, because, despite best intentions, the final configuration of application execution is usually not known until some amount of actual testing and experience has been gained.

The reasons for needing the capability to execute some application logic on server machines include 1) to reduce the amount of data flowing from the server to the workstation, 2) to take advantage of specialized features available on the server platform, and 3) to better match the scale of the computing task to the available platform.

In the first case, decision support applications are, by nature, involved with the summarization of large amounts of data. A report which shows totals for sales across many locations, time periods, and lines of business has passed a very large amount of data at some point in its generation. When these production applications were executed in central processors, the fact that large quantities of data were being passed to the 'processor' was masked. However, as this function is distributed out to workstations, it becomes immediately clear that much of the work of the decision support application was strictly data summarization. If some of that work cannot be accomplished closer to the data, reducing the amount which must flow out to the 'computing' processor, a serious contention for network resources will arise, not to mention possible overloading of the workstation platform. In essence, there is a need to treat the server platform as not just a data server, but also as a compute server.

(Much of this work can be accomplished through the creation of an information database, where standard summarizations of data are accomplished as a central maintenance task, and the results made available to workstation applications. This paper will not go in to a discussion of information databases.)

In the market today, there are some tools offered to enable this 'pre-processing' data reduction. They can be summarized as

1. SQL summarization features of client/server databases
2. stored procedures
3. remote procedure calls
4. remote submit

The ability to use SQL summarization functions (e.g. GROUP BY, ORDER BY) offers application programmers the capability of summarizing data on the server platform before moving it to the workstation. Where the client/server database is not serving as the target database of a production on-line transaction system, this additional SQL processing within the database engine may be acceptable. Of course, this presumes that all of the data of interest are stored in a relational database to begin with.

Stored procedures are another example of enabling the server platform to execute some application logic on data before moving it to the workstation platform. With a stored procedure, application programmers are able to build calls to separate tasks, using either SQL or another 3rd generation language, and store these inside of the database. These stored procedures can then be called from within SCL queries, and can make requests for compute services from the server, as well as data services.

Again, this strategy of building executions which will pre-process data on server platforms makes better use of all available resources. However, as with other systems, stored procedures usually reside in relational database systems. To the extent that these systems are where the data are stored, then the facility is
A third tool for use in enabling server platforms to execute application logic is the remote procedure call, or RPC. Remote procedure calls are best known in the context of the Distributed Computing Environment (DCE) standards, sponsored by the Open Software Foundation (OSF). DCE is a collection of services intended to standardize some of the services all applications might use as they are developed to run in a distributed environment. Some of these services include integrated file system standards, security and authentication services, universal naming conventions for resources, as well as remote procedure call mechanisms. Many vendors are beginning to make these services available within their operating systems (UNIX) or as additional services to be licensed.

Specifically, remote procedure call technology allows application logic to be built on a local platform (for testing purposes), then, once it appears to run as designed, it can be separated into the parts which will run locally and those which will run remotely. These pieces are then moved to the target platforms and recompiled. When finally installed, they allow a local program to make calls to functions, as if they were actually available in the local environment. However, the RPC services transport the call instructions to the remote platform, and the 'called' function executes, returning results familiar with writing 3rd generation style procedures.

The RPC technology is new and, as a consequence, somewhat immature. And it still depends on programmers knowing 3rd generation language techniques, as well as having to compile the separate parts in each of the target platforms. It. taken with the other DCE services, however, offers great promise for aiding the distribution of application executions in the future.

Another capability in this environment is, of course, the remote submit capability enabled through SAS/CONNECT software. Remote submission of instructions for execution on server platforms has been a part of SAS connectivity capabilities since their inception in the original micro-to-host link. Remote submit capability is that part of SAS/CONNECT software which enables a server platform to perform any or all of the functions of the local platform. This capability has two distinct advantages not enjoyed by the previously named strategies.

The first of these, common syntax for application development across all supported computing environments, is based on the MultiVendor Architecture® (MVA) of the SAS System. MVA assures that application syntax will be consistent across all supported platforms. Thus, applications developed on a local workstation platform can be moved with confidence to any supported platform, with the assurance that they will work. There is no need to code differently for each target platform.

The second advantage SAS users enjoy is the full feature set available on both ends of a connection. Because SAS/CONNECT software enables communications between two or more SAS sessions, applications have available the full range of SAS capabilities. There are no substrate versions of SAS operating via SAS/CONNECT software. As a result, applications designed for local execution can be segmented at any logical point, with the knowledge that the companion capability will be in the remote environment, accessible through the same syntax.

This also means that a distributed function application does not need to be recompiled in the server environment. The parts of the application destined for execution on the server platform are simply sent to that platform, and execute in a known SAS environment. This has the additional advantage of allowing application programmers to quickly change the segmentation level of an application by simply adjusting local SAS statements.

GUI and Application Development tools

The availability of end-user friendly graphical user interfaces has been the key advance in making computer technology accessible to a broad range of users. What used to be a mysterious environment, understood by few, is now the familiar, enjoyed by many. The explosion in the use of workstation computers is directly related to the advances in user interfaces, as well as the tremendous lowering of price. This use of workstations to manage the user interface segment of distributed applications is a given. Workstations already are equipped to manage screen environments and can save server platforms from having to do this memory expensive work.

Application development tools are also keeping pace with this graphical interface style of presentation. Tools now make use of object-oriented concepts to allow user presentations to be built. As distributed function applications become more in demand, these tools will be required to not only build attractive data capture and presentation screens, but integrate critical analytic capabilities as well as support for computing in remote environments.

With advances in our own SAS/AF software and SAS/EIS software, the SAS System offers just such a set of integrated tools. Now, SAS application developers can have access to object-oriented strategies for building local applications, as well as being integrated with the full range of SAS System capabilities for analysis and presentation of data.

SUMMARY

As enterprises approach the implementation of client/server technology, a logical task awaits them. The exercise of identifying the logical components of an application is a key first step. Any decisions about subsequent physical location of these logical segments will be based on how well application developers understand the ramifications of splitting applications.

A variety of styles of cooperative processing is available for implementation. Care should be taken to appropriately apply the style which will best match up with the demands of an application and take full advantage of the integrated enterprise computing environment.

The SAS System for Information Delivery offers the most comprehensive set of capabilities for building enterprise strength client/server applications.

SAS, SAS/ACCESS, and SAS/CONNECT are registered trademarks, and MultiVendor Architecture, a trademark, of SAS Institute Inc., Cary, NC, USA.

Other products mentioned are trademarks or registered trademarks of their respective owners.