Distributed Application Processing and How to use it.
or
Stop Wasting those PC Mips!

Patrick Fioravanti
Infocentre Corporation
7420 Airport Road
Suite 201
Mississauga, Ontario
Canada L4T 4E5

As Fourth Generation software and data communications technology becomes more prevalent throughout the HP3000 community, the number of opportunities available for networking the mini with our PC's grows significantly.

With many organizations reaching and exceeding the computing capacity of their HP3000, the concept of redistributing the load, moving some of the application processing to the micro computer becomes very attractive. Distributed processing is a reality today, but for many shops it represents a new frontier that should be approached with caution, in a premeditated way.

This paper will take a close look at distributed application processing, involving the networking of the HP3000 with personal computers. We will first introduce some of the concepts at work, then describe the various data communication topologies that can be implemented. Having set the foundation for the discussion we can move on to application design possibilities - investigating how new applications might be designed in order to capitalize on the system resources made available through the networked configuration. Along the way, we will be providing some guidelines for effective use of this distributed processing concept based on the capabilities, strengths, and weaknesses of the various system components (both hardware and software) within the network.

Introduction

Historically, within the HP3000 environment we have seen the flow of corporate information managed almost exclusively by the central computing facility. Serious business applications have been developed and implemented on our HP3000s. With many organizations reaching and exceeding the computing capacity of their HP3000, the concept of redistributing the load, moving some of the application processing to the microcomputer becomes an attractive alternative. Pursuing this avenue has led a number of HP installations into the realm of distributed processing, where applications are developed that utilize PC hardware and software resources in addition to HP3000 based resources.

What are the perceived benefits of distributed processing?

The benefits associated with the integration of PCs with the current computing environment can be discussed under these headings:

* Offloading the central machine,
* Reducing costs,
* Reducing application downtime.

Offloading the central machine

Today the HP3000 is used typically by data processing professionals and end users alike. Programmers, analysts, database administrators define, develop, implement, and maintain production application systems. End users access these implemented systems, undertaking the application processing. In a lot of cases, the combination of system development and production activities strain the system resources, sometimes yielding unacceptable levels of performance during the peak processing hours of each business day. Something has to give.

Consider that each of the PCs sitting on desks in the user community as well as the DP department, has its own CPU coupled with significant storage and memory capacity. Aggregated, there is a lot of computing power in our PCs waiting to be harnessed effectively. Using PCs that are already cost justified, to potentially double the available computing power represents a significant opportunity to shift some of the current load away from the HP3000.

Reduced costs

A number of computing costs can be reduced by migrating some of the processing from the HP3000 to the PC. Hardware acquisition is one of these costs. It is not difficult to acquire a PC with a generous configuration for about the same price as a good quality display terminal. The same applies to software, as PC software is available for a fraction of the price of minicomputer software possessing similar functionality. Once the PC is established as an extension of the central facility, the incremental cost of hardware or software additions is substantially lower than on an HP3000.

Thirdly, we can look for decreases in communication costs. In cases where remote terminal workstations are replaced with PCs, the PC can execute the application locally. While doing so it may be possible to sever the link to the HP3000, thereby reducing data communication costs significantly.

Reduced application downtime.

With the PC comes the ability to collect local transactions on an attached disk system that can later be used to update the minicomputer. The PC then acts as a spare machine that can be pressed into service should the HP3000 be unavailable. Furthermore, a termporarily defective PC can be replaced far more easily than can an HP3000. With PCs, hardware redundancy becomes affordable.

Distributed processing - How?

If you accept that there are real benefits associated with PC integration, then we should examine the necessary tools, as well as some of the concepts behind them.

Ensuring success in distributed processing hinges largely on the compatiblity between the various machine environments. Application developers and users must be able to move easily between PC and HP3000 based processing. We can accomplish this by adopting an application development environment that operates on both machines. Specifically we require the same programming language, and the same database management system. A common application development environment provides two very important benefits:

* The application development staff can build PC applications with minimal retraining. All existing skills relative to the programming language and database design, creation, manipulation are transferable.

* Application development activities can be undertaken either on the PC or the HP3000 regardless of where the finished product will ultimately run.

There are a number of strategies for distributing application development and processing. A very simple strategy results in standalone applications being developed for processing on one or more PCs. This strategy is useful, but generally finds itself restricted to simple applications processing non critical data. More sophisticated strategies are required for applications that must be distributed across a number of machines. That is, the processing may be split between the HP3000 and PCs as well as the data. Implementing distributed applications in this fashion requires a communication facility - a means to transfer data between the two machine environments.

We need to set in place appropriate data security and access control procedures. In the past, with all application processing taking place on the HP3000, we have not felt the need to be concerned with these issues. MPE provides reasonable facilities for access control and data security. Operational procedures are already in place to ensure proper backup copies of data are taken on a regular basis. None of this security environment is automatically available on a PC. Rules need to be implemented that ensure PC resident data will be adequately protected. Furthermore PC application security is needed to control, as much as possible, access to the processing capabilities provided by the workstation.

With these tools and controls in place, we can turn our attention to developing and delivering distributed applications. These applications must be designed wisely, distributing the data, the processing, and fine tuning the various connections such that the application satisfies the users requirements and makes optimal use of all available computing resources.

The communication facility.

A communication facility is a central requirement for implementing distributed applications. The inter-machine communication enables the activities on the machines to remain coordinated. There are several ways to design the PC to HP3000 communication, lets examine first a batch approach.

Using standard PC - HP3000 file transfer utilities it is quite straightforward to implement a batched, bidirectional communication facility. This can be used during system development to transfer text source files between machines, and equally during system processing to transfer text or binary data files back and forth. With this approach we can design distributed applications that are *batch integrated.* This can be useful with applications that are based on processing cycles. During a processing cycle, transactions can be entered and captured on a PC. At the end of the cycle (nightly, weekly, monthly, etc.) the detailed transactions can be extracted from the PC database and uploaded to the HP3000 where they are posted to the central database. At the same time, updated versions of the reference or master files can be extracted from the HP3000 and downloaded to the PC(s). Software products are available that will work in conjunction with the file transfer utility to automate a scheduled transfer of files.

On the surface, it may not be immediately obvious how it is that this distributed processing approach provides benefits. Consider however that the processing involved with data editing and general transaction processing is offloaded from the HP3000. The PC earns its keep. For remote workstations, there can be savings realized in data communications costs. The workstation need not be connected all day, and when the connection is made, a concentrated stream of pre-edited transactions is efficiently transferred.

The batch integrated approach, although simple to implement, may not be suitable to all applications. It carries several disadvantages:

* The master files may be so large as to make their downloading impractical, if not impossible.

* The downloaded files are duplicated on multiple PCs. Duplication of corporate data profoundly complicates data and information management.

* Often the data entry workstation needs real time access to HP3000 data. Batch updates may not be sufficient.

Distributed applications requiring more timely access to large, sensitive corporate data files are better served by an *interactive* networking strategy. This more sophisticated form of communication results in a tighter, more cohesive integration strategy. Interactive networking enables an application running on a PC to instantly access information (read, write, or update) regardless of where that information resides, in a fashion that is transparent to the application and the user.

Interactive networking is accomplished using data communication software resident at each end of the connection. The PC resident component sends requests to the HP3000 whenever access is required to centralised data. The HP3000 resident component acts as a server to the PC. It waits for data access requests emanating from the PC, and responds to them. Typically the response involves formulating an Image access call, then sending the result of the call back to the PC. In general the type of service provided by the host resident software need not be restricted to accessing Image databases, however this seems to be a very useful service to offer a PC.

This arrangement is conceptually similar to accessing remote Image databases within a DS network of HP3000s. It would be simple enough to implement 'DS' type intrinsic calls within the PC resident software. These intrinsics could be invoked by the application whenever access is required to HP3000 resident datasets. Another approach involves defining the remote datasets within the schema of the PC database. A mechanism is required permitting the database designer to designate one or more datasets as being 'logical' datasets. They are part of the application, but physically reside on another machine. This approach carries three advantages over the 'DS' approach:

1) Accessing remote files is transparent to the application programs. The programs issue a read, write, or update request against a specific record or file. The Database Management System determines where the file physically resides, and hence what is required to satisfy the application request.

2) The local database knows the structure of the remote files since they are defined in the schema. Accordingly any interrogations of the remote dataset structure can be responded to locally, there is no need for remote communications to satisfy the request. This is tremendously beneficial for applications that make extensive use of the DBINFO intrinsic.

3) Within the PC database schema, the remote dataset definition can be a logical view of the actual dataset definition. The PC schema need only define the items of the remote dataset required by the application. Only the items defined in the PC schema will be transferred, thereby optimizing the data communications.

To visualize how this works, take as an example, a manufacturing application involving a large Parts master file. We can undertake transaction entry and processing on the PC, designating the Parts file as a logical dataset, physically resident on the HP3000. At run time, any "DBGETs" directed against the parts file will be trapped by the PC Database software, and sent via the communication link to the HP3000. The request is accepted by the network server, and formulated into a "DBGET" directed at the appropriate Image database. The result of the call (the record buffer if successful, otherwise the status array with the appropriate error information) is then transferred back to the PC where it is accepted by the Database software and returned to the application program. The same concept applies to other database access calls (writes, deletes, updates, locks).

Although simple in concept, the actual implementation of this communication facility can be quite complex, allowing for all of the possible data communication configurations, data compression techniques (for faster data transfer rates), and data encryption (for securing data as it is passing through the communication link).

The interactive networking facility permits the definition of physical (local) and logical (remote) datasets when defining the application database. A third type of dataset can be considered; a blend of local and remote. The implementation of this type of dataset provides a form of *Caching.* For example, if a request is received to read a record from our parts file, the DBMS looks for the record in the local dataset. If it does not exist there, it is retrieved from the HP3000 and written to the local dataset. The next time we wish to access the same record we may retrieve it without a remote access. Over time, the local dataset will be populated, on an 'as needed' basis, until it holds all of the master records required by this PC workstation. Although this implies data duplication, it means that at some point in time the communication link can be severed without service disruption. This concept is well suited to applications that operate according to Paretos' Law, where eighty percent of the processing is directed against twenty percent of the data records.

Regardless of the options chosen, the principle remains the same. By defining the structure and the location of application data within the database, the run time component of the DBMS is able to handle the data communication. Access to the remote files happens automatically in a fashion that is transparent to the programmer and the user. Distributed processing is made possible.

Connections

PCs using the communication facility are connected to the HP3000 in a standard fashion. Once connected, there are several means we can consider to optimize the connection.

At its simplest, the connection takes the form of a serial communication line from the PC to a terminal port on the HP3000. In reality, this connection might be direct, or pass through a very complex data communication network, including modems, multiplexors, and packet switching networks. Generally speaking, if we can connect the PC such that it can initiate a terminal session, then we can acomplish batch or interactive networking through the connection.

When using interactive networking, the PC can initiate a terminal session, and within the session activate the HP3000 resident portion of the communication software. This process might be activated automatically from the PC as part of the "DBOPEN" and similarily terminated automatically as part of the "DBCLOSE".

It may be considered wasteful (or perhaps excessive) to allocate a terminal port, MPE session, and network communication process to each PC in the application network. We can optimize this somewhat by dedicating resources to the application.

If it is known that a specific port or ports will be used only for interactive PC networking, then it becomes possible to launch a single communication server process that treats the ports as files, and waits to service requests emanating from the attached PCs. This strategy reduces the number of HP3000 processes (one per network instead of one per port), and the number of MPE sessions (the PCs don't initiate a terminal session), at the expense of dedicating the port(s) and scheduling the systematic initiation and termination of the network server.

Depending on the design of the application, there may not be sufficient volume of data traffic to warrant the allocation of one port per PC. In many HP3000 shops ports are expensive and scarce resources. In situations where specific ports are dedicated to interactive networking we can attach several PCs to one port and enhance the communication facility to channel the data messages separately for each PC sharing the port. This serves to optimize port utilization.

We can extend this concept further. If it is known that all of the PCs in a given network will be processing the same application, then the HP3000 based server can be customized for the application. By always having files and databases open, and other initialization type processing completed, the PC will never have the servicing of its request delayed unnecessarily.

These options pose a tradeoff: optimizing resource utilization versus flexibility.

Application Design - Resources

Having covered the concepts and some of the possibilities, let's concentrate on putting them into practice. By evaluating the various software and hardware components that we have at our disposal, we can develop application design guidelines that result in optimal use of the resources.

I/ Available pool of software:

<u>HP3000</u>	<u>PC</u>
MPE	MS-DOS
Image / File System	Image Clone
Development Language	Same Language
	Personal Applications

II/ Hardware:

<u>HP3000</u>	<u>PC</u>
Disk (lots, shared)	Disk (less, dedicated)
Memory (limited, expensive)	Memory (limited, cheap)
CPU (powerful, shared)	CPU (powerful, dedicated)
Peripherals (high speed)	Peripherals (few, low capacity)

III/ Data Communication Link:

- Data transfer rate	(slow)
- Expense	(varies with configuration)
- Reliability	(varies with configuration)

An evaluation of the respective strengths and weaknesses of the available resource pool shapes our application design guidelines.

On the software side, MPE provides more sophisticated file system and security facilities. Included with the MPE file system is automatic multi user capability. By equipping ourselves with the same application development environment on both machines, we make Image and our chosen development lanaguage generally available. The PCs have an edge with readily available, powerful and friendly personal application software that can be used for data analysis (spreadsheets), graphics, and text processing.

Of the available hardware resources, the PC complement is dedicated to a single user, while the HP3000 hardware is shared among several (many) competing users. Generally speaking, the hardware devices connected to the HP3000 are higher capacity and boast faster access speeds, although in reality they may service an individual user in a more sluggish fashion when heavily subscribed.

The data communication network is arguably the weakest link in the chain. Even the fastest of serial data transfer rates pale in comparison to the rate at which data is transferred from disk to memory inside an HP3000. Furthermore, the network is susceptible to unavailability due to a software or hardware failure of the host (HP3000), or failure of any component of the communication network.

Application Design - Guidelines

The remaining challenge is to design distributed applications wisely. In our wisdom we must distribute the processing as well as the data storage in a manner that most effectively utilizes the available resources. Based on our evaluation of resources we should:

* fully exploit the availability of PC hardware,

* offload personal data analysis tasks to the PC

* reduce dependence on overtaxed HP3000 resources.

* economize on interactive data transfers

* use the MPE file system when data must be shared, or data security/protection is critical.

This translates into:

1) Utilize the computing capability of the PC for things like data entry tasks. Data edits, validations, calculations, error handling, screen compilations and screen I/O are then offloaded from the HP3000. The data entry operator will appreciate the consistently crisp response that can be provided by the PC.

2) Distribute data such that files requiring shared access remain on the HP3000. These files can of course be accessed through the interactive communication network. For most applications, this guideline results in reference files (frequently the "Master" datasets) residing on the HP3000, while the transactions ("Detail" datasets) can be captured on the PC. Regular validations (lookups) to the reference files during data entry can be handled by the communication facility, the output of the data entry (the transactions) in many cases need not be communicated to the host. This strategy offloads a significant amount of disk I/O from the host and also minimizes traffic on the data communication network. Should the PC resident transactions be required on the HP3000, for batch reconciliation or reporting purposes, we can consider a batch transfer at off hours to accomodate this.

   Although on the surface it may seem advantageous to leave all the files on the HP3000, this strategy heavily taxes the communication network (the weak link), and in some cases floods the HP3000 with file access requests.

3) Design the application to be fault tolerant. Local processing should still be possible during temporary periods of host unavailability. This can be accomplished by making some data edits or validations optional (flag the transactions as requiring some sort of batch validation later on). When host unavailablity is predictable, an optional downloading of the master files (or some subset of them) can be done ahead of time, and the data entry programs instructed to access the local files. This is similar to the application caching concept that automatically downloads remote records as they are accessed, and permits disconnection from the HP3000 without service disruption.

4) Leave large volume batch oriented tasks on the HP3000. The host has the speed, power, and large volume peripheral devices to handle these tasks efficiently.

These points offer general application design guidelines that may prove useful as a starting position when conceptualizing the design of distributed applications.

Summary

The technology exists today making distributed processing a reality. For many HP3000 shops this constitutes a new frontier, presenting new opportunities to address traditional problems of computing resource allocation and utilization. It also adds a new dimension to application design considerations.

It is through a solid understanding of the underlying concepts, and available options that we can adopt and implement an effective distributed processing strategy that will help to achieve the stated goals of PC integration.