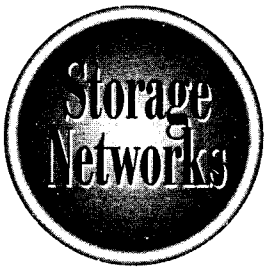
The Complete Reference

Part V

Application—Putting It Together

The Complete Reference



Chapter 17

Defining the I/O Workload

Capacity planning has always been something of a black art. Commodity-level distributed computing has driven IT to conduct a simplistic set of activities meant to plan and implement computing resources. Any discussion on applying the concepts of storage networking must touch on key workload definition concepts that are part and parcel to an effective capacity plan and to the possible rejuvenation of a more effective capacity planning model. Part V orients the reader towards applying storage networking concepts by way of an initial discussion and overview of workload definition and planning. The following chapters contained herein focus on identifying existing and anticipated workloads where storage networking solutions can be implemented for maximum value and productivity to the data center. SAN and NAS configurations will be addressed in analyzing the particular workloads of each.

We have explored and used the term workload many times already, and in Part V, it remains important to our discussion. Chapter 17 provides a detailed examination of workload definitions, identifications, and planning. Workload guidelines will be placed into a real-life context that characterizes both business applications and support activities, as well as the maintenance of capacity strategies as they relate to storage networking. Many of the activities that you manage on a daily basis may be enhanced or hurt by the new technologies involved in SAN and NAS solutions. In an effort to provide as much relative information as possible, generic descriptions and examples are used in Chapters 18 and 19 to cover several specific workload characterizations.

Part V addresses the usage of SAN and NAS as an integrated solution. Chapter 20, for instance, explores situations based on current IT experiences and also looks at future uses of integrated SAN and NAS solutions. This becomes a complex exercise when multiple workload scenarios are cast within the same environment. The ability of microkernels to communicate effectively and for storage caching mechanisms to operate in tandem becomes a challenge when considering the future of heterogeneous storage operation. This is compounded by the ability to maneuver within a distributed file system that incorporates optimum support for both traditional workloads yet sustains the proprietary nature and operation of database technologies. The current evolution of common application services through enhancement of web-enabled solutions will incorporate all the attributes of OLTP, batch, and data-intensive I/O, and will play a key role in these discussions.

Moving storage networking into the future requires an analysis of integrated storage networking solutions, using both FC- and IP-based connectivity schemes. Because of this, Chapter 20 addresses key future technologies, showing how they may affect IT storage networking solutions. Such technologies and standards include iSCSI, the Internet SCSI standard that allows SCSI commands to be encapsulated and transmitted through an IP network. In addition, we address the slowly evolving InfiniBand standard and infrastructure advancements that will result. Finally, we'll look at some of the internal bus connectivity standards and strategies that could have a dramatic effect on the data storage hierarchy as we know it today (see Chapter 5) exploring products such as AMD's Hyper-Transport, the industry standard VI transport mechanisms, and Intel's next bus technology, Rapid-IO.

Storage Planning and Capacity Planning

Planning and installing applications continues to require a big effort within IT, especially when it comes to applications that are considered *enterprise-level* (a term which has never seemed appropriate, by the way). Nevertheless, the term was employed by large system players (read mainframes) to differentiate their solutions from those of the small system players (read PCs) in the early days of the user revolution. Commonly known as the PC "wars," it was a time in which many business users made attempts to buy their own IT infrastructures for pennies on the mainframe.

It was about this time that sophisticated capacity planning and workload management started to go out the door—out the door, out the window, and out of IT standards and practices. All in all though, those tools and activities provided a certain amount of pride and gratification when the systems (read mainframes—again) were upgraded and everything not only worked, but worked the same as it had before, and had improved user response time, balanced utilization, and a batch window that was completed two hours earlier.

Well, we traded those simple days for ubiquitous and prolific computing resources based on everyone's desktop, job, travel agenda, and meeting rooms. We traded the security of centralization that made the capacity planning a manageable and gratifying entity for a Wild West distributed environment that reflected the scalability of the *box* we were operating on, be it UNIX or Windows. The mantra was: If the ones we're running don't work, are too slow, or can't support the network, we can always get new ones. What could be simpler? A capacity plan based at the *box* level. If it doesn't work, just get another one. They don't cost much. We don't need to consider no stinking workloads, we can process anything, and if it grows, we'll just add more *boxes*. The same goes for the network.

A wonderful idea, and to some degree it worked—until the applications started to take advantage of the distributed network features of the boxes, as well as the increasing sophistication and power of the boxes. Yes, the little *boxes* grew up to be big boxes. Others grew to be giants, the size of mainframes. It should be pointed out, however, that many boxes had genes that stunted their growth, keeping them from becoming *enormous* mainframe-like boxes. Even then the *box* applications grew in sophistication, resource utilization, and environmental requirements.

Note *The "gene" mentioned here was referred to as the Gatesonian gene, whose scalability would not grow beyond the confines of a desktop.*

Interestingly enough, we seem to have come full circle. Although nothing essentially duplicates itself, our circle of activities has landed us back near the realm of the—gasp!—workload. It's not like workloads ever really went away, we just haven't had to recognize them for a while, given we all got caught up in the *box level capacity planning* activities, known as the BLCP practice.

BLCP is not unlike the IT practice of going into installation frenzy during certain popular trends—for example, ERP, relational databases, e-mail, office automation, web

site, intranet, and now the dreaded CRM. Such things cause us to be driven by current application trends, leaving us to distinguish between CRM performance, ERP utilization, and e-mail service levels. To a great degree, we handled these installation frenzies with the BLCP (box-level capacity plan); we can implement anything by using the right box, as well as the right *number* of boxes.

Several things within the applications industry have rendered the BLCP practice obsolete. First and foremost is the sophistication of the application. Today, applications are distributed, datacentric, hetero-data enabled, and, the up-and-coming process, de-coupled. Second, the infrastructure has become specialized—for example, there is the network, the server, the desktop (for instance, the client), and the storage. Third, the infrastructure itself is becoming further de-coupled. More specifically, storage is taking on its own infrastructure, with storage network freeing its processing bounds and restrictions from the logic of the application, as well as the network overhead of the client/server model, becoming itself immunized against the Gatesonian gene.

Note *Hetero-data is the characteristic of an application that requires multiple data types to perform its services.*

Finally, the common services that support the application logic infrastructure are quickly becoming both commodity oriented and public in nature. We are moving into a future where applications are programmed by end users to employ web services. These services access lower level infrastructures and subsequently complex and proprietary support products operating within a particular information technology structure. Users will likely take it for granted that adequate resources exist.

These conditions will further render the storage entity with its own operating infrastructure. However, this will require that it evolve as a more complex structure, supporting an even more complicated set of common services that includes application logic, network processing, and common I/O facilities. Each of these entities will have to understand and configure themselves to effectively operate with existing and future workloads that will dictate the processing and physical requirements necessary for just minimal performance.

That's why workloads are important.

The Definition and Characterization of Workloads

Workloads are traditionally defined as application processing modules that exhibit similar resource requirements, processing characteristics, and user expectations. Figure 17-1 further depicts this definition with its characterization of banking transactions. For example, at one time or other, we've found ourselves in a bank line with someone who's balancing their bank statement, applying for a car loan, and/or trying to find a missing check from, say, 1982, all at the same time. Their transaction, obviously, is much more resource-intensive than the deposit of your Grandma's birthday check.

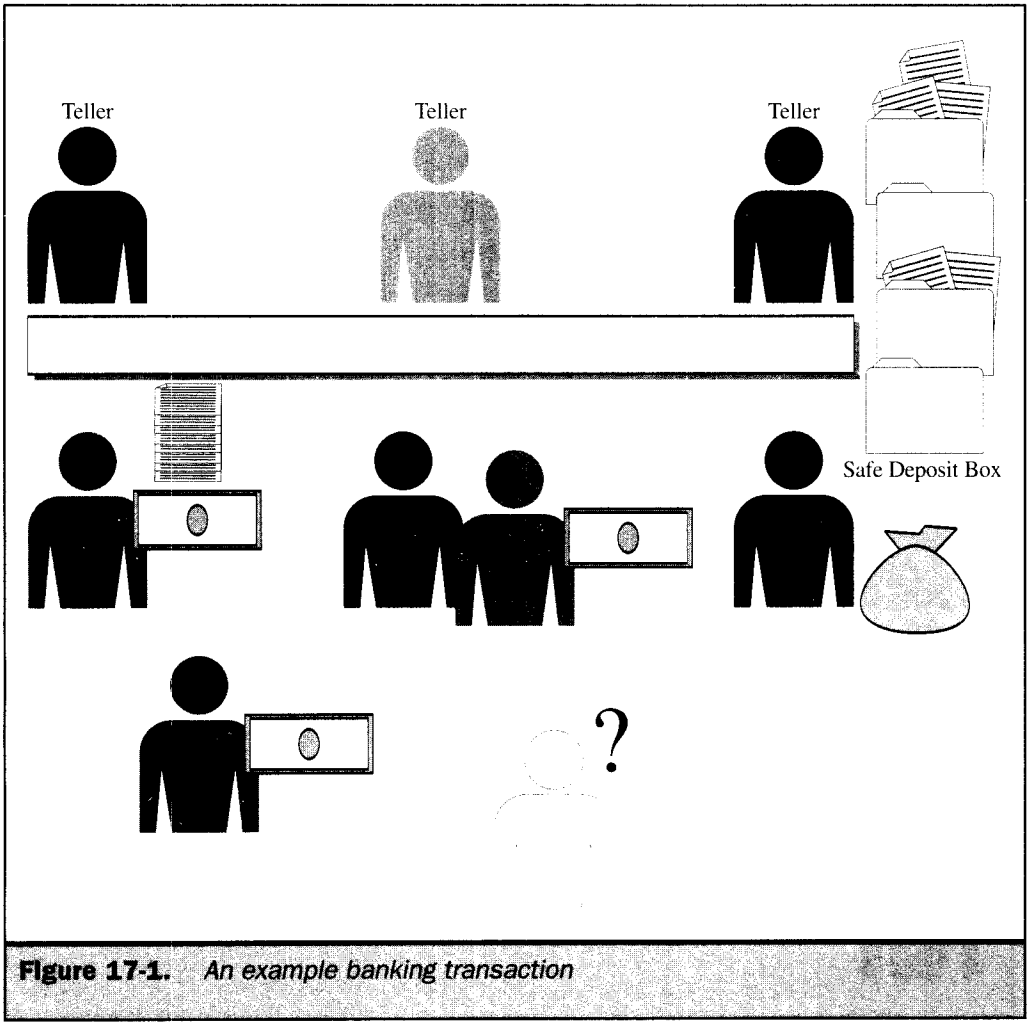


Figure 17-1. *An example banking transaction*

So, if there were a categorization for the types of transactions the bank handles, everyone would get serviced more efficiently within a prescribed set of expectations. However, this would require the tellers to be categorized by the type of transaction they perform. Tellers would have appropriate resources at their disposal to deal with the transactions they service. There you have it, a simple workload categorization for the tellers, who, in this example, can also be characterized as servers (depicted in Figure 17-2).

The tellers have a set of resources and types of transactions they work on, giving certain kinds first priority. However, since these are all general-purpose tellers, they can work on any transaction should they run out of the particular type of work they are optimized for. Keep in mind that they are optimized for a particular type of workload and prioritize their work on that basis. Because of this, they may not handle a different workload as effectively as a peer teller that is optimized for a different type of transactional processing.

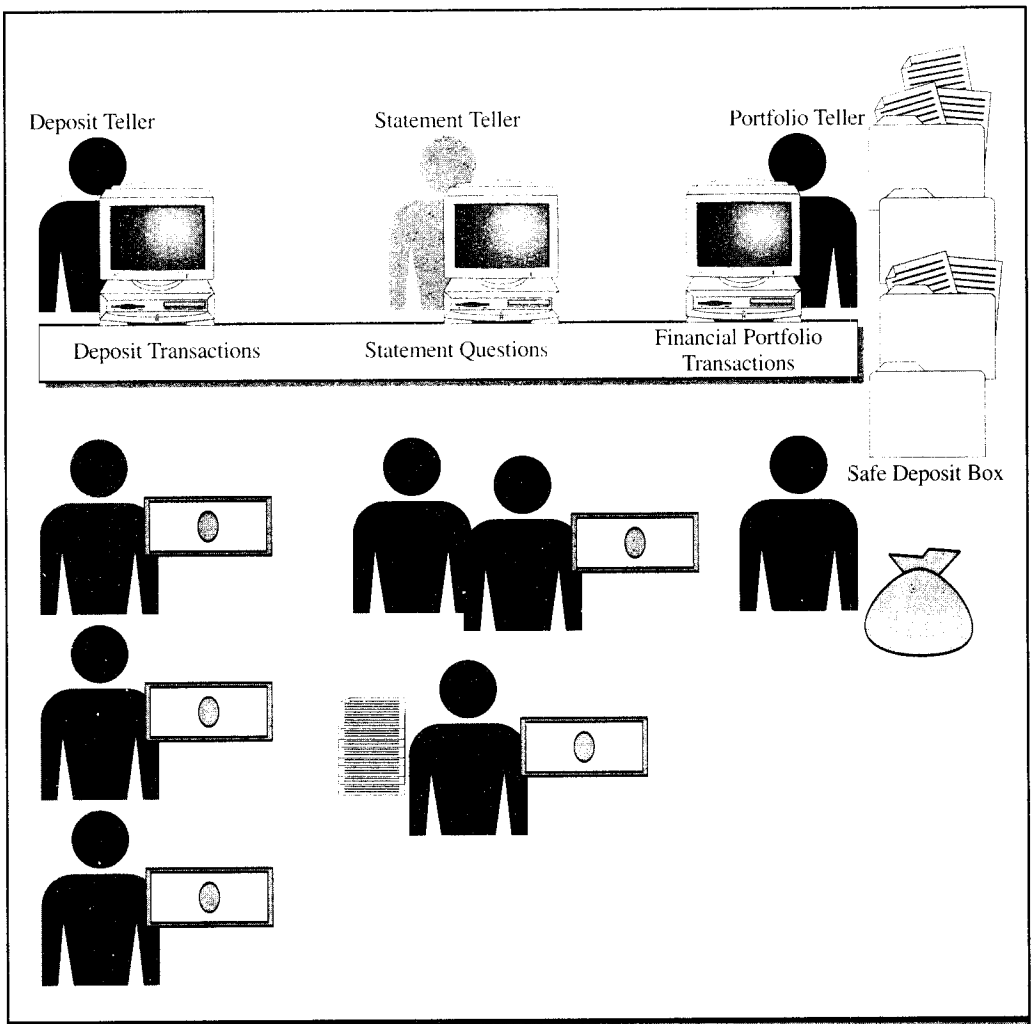


Figure 17-2. *Organizing workloads for maximum performance*

In this example, a simple transaction like depositing your Grandma's check can be taken care of by a teller, who functions similar to a computer transaction server that specializes in, and is optimized for, checking deposit transactions. The teller has the appropriate resources to process your deposit within a service time you are expecting. Other tellers process different transactions, such as safe deposit transactions (very physical, employing various resources), bank statement questions (very process-intensive, requiring much data), and financial portfolio transactions (multiple processes, compute-intensive with high priority).

The Business Application

By using the banking example, we can determine all the necessary considerations for identifying, describing, and characterizing the workloads. We first define the banking activities as the business application. Within the application are several subsets of activities, which can be directly related to application programs that support the banking business application. We will now convert our abstract teller infrastructure, supporting the banking application to a computer infrastructure with servers and related components that allow users to perform transactions. This is illustrated in Figure 17-3 and will serve as our basis for workload considerations for this chapter.

Using Figure 17-3, we see that identifying parts of the application becomes fairly straightforward, as an application development group defines the functions performed by the business application. This provides an inventory of application programs that have been developed and need to be available for meeting the users' needs through their transactions. We evaluate each function, or program, to our computer configuration, as indicated in Figure 17-3.

We do this by understanding the data organizational structures required for each application program, the data paths it uses, and the user access requirements for each. For example, our checking deposit program uses a database and related log files to handle its processes. The user access demands that the deposits be reflected in the daily database but also in the statement of record for the customer, which is in another database. Further subsets of this information go toward an activity log used for updating other applications within the banking business application.

Deposit Application—Workload Attributes

The previous information describing the banking transaction pretty much defines the data paths required for the deposit program. We then coalesce this information into a set of workload definitions for the deposit application. Based upon the organizational

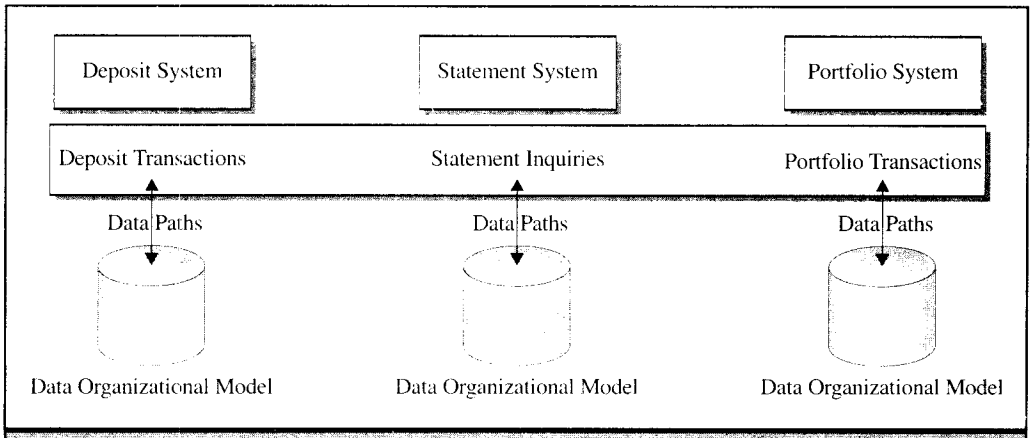


Figure 17-3. *Banking configuration example: defining the business application*

and user access information, we can define additional attributes of the data path. This allows us to describe the deposit application with the following attributes and begin identifying it as a workload.

- **Data Organizational Method (DOM)** Relational database, flat files, and temporary suspense files.
- **User Access (UA)** Online transactions that require immediate update to daily customer tables, with conditional processing based on secondary transactions to customer system record tables and log suspense files.
- **Data Paths (DP)** Macro data paths require update access to daily customer tables, read access to daily suspense tables, and a system of record tables. In addition, there are multiple paths required to facilitate the reliability, availability, and serviceability of the application.

By further defining the technical attributes of the workload, we begin to make decisions on where this particular program will be placed within our infrastructure. These are illustrated in Figure 17-4, which are depicted as updates to our initial Figure 17-3. We can now overlay workloads within this configuration. What we have in our sample banking application is a classic OLTP workload that supports the deposit subset of functions. The OLTP workload dictates several configuration characteristics at a minimum, with additional considerations based upon estimated traffic and growth.

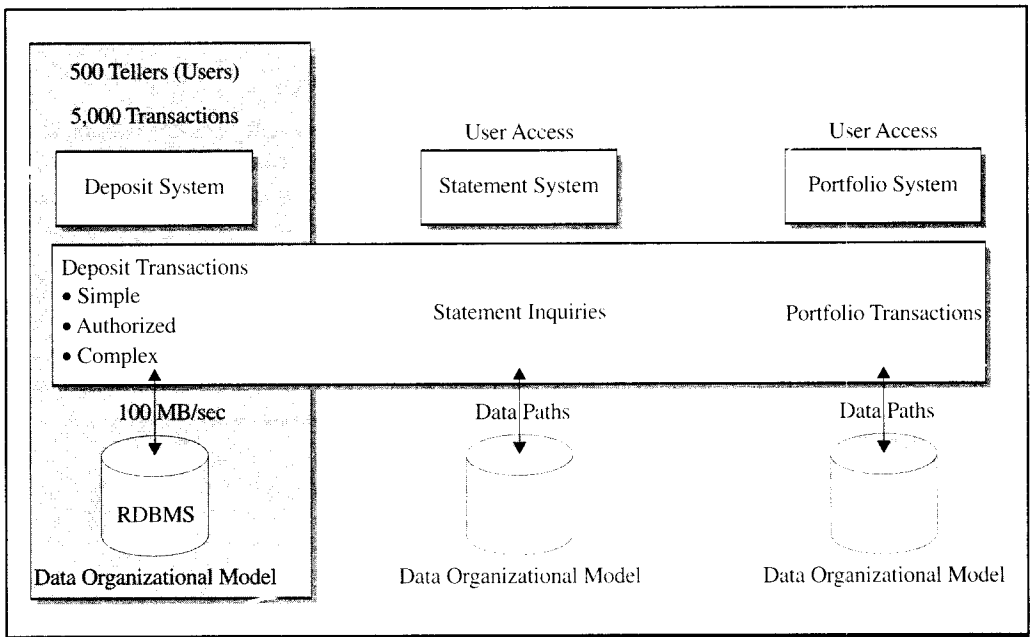


Figure 17-4. *Placing workloads within an infrastructure*

This defines a simple model for identifying and describing a macro set of attributes for the workload. Having done this, there are additional details that become necessary as we move to place the application into our sample configuration. Now it becomes necessary to further understand the resource utilization, traffic arrival rates, and environmental conditions surrounding the support of this workload. However, before we proceed with deriving additional details from the deposit application, we must address in summary fashion the analysis of the complete inventory of the Banking Application. This requires using the same process that created our initial workload definitions for the deposit application.

Given this iterative process, the importance of analyzing the entire Banking Application in context cannot be overemphasized. During this process, we find that many of the applications within the Banking Application have similar workload characteristics, such as same database, similar user access demands, and process requirements. On the other hand, we also recognize that many applications have completely diverse sets of attributes. Our example shows two other distinct workloads for placement within our configuration—a batch workload and an archival workload—each of which has a unique set of user transactional requirements.

Note *It should be noted that such "distinct" workloads were chosen to reflect the ubiquitous nature of each configuration. These can be found in almost all IT data centers and remain applicable across business and scientific applications.*

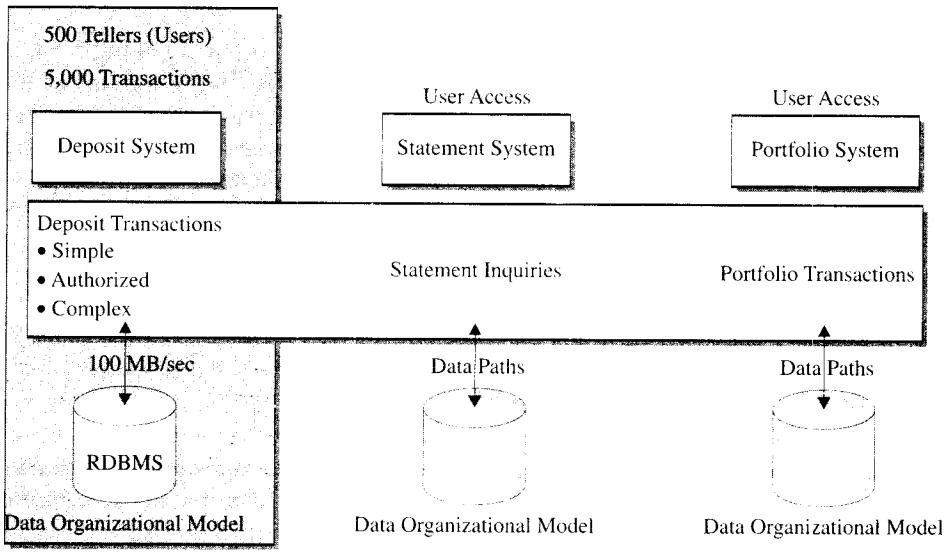
This requires an accumulation of resource, access, and environmental elements to derive a total picture of the workload. As such, our original example of the banking deposit becomes only a part of the Banking Online System, which we have concluded from its accumulated characteristics to be an OLTP workload, with a set of requirements designed to meet user expectations. Figure 17-5 offers guidelines for identifying workloads and defining their characteristics.

Another interesting aspect to this exercise is something you may have already concluded as we moved through our workload identification and definition stage. That is, with few exceptions, the majority of resource requirements and characteristics center around the I/O of the workload. Certainly I/O plays a pivotal role in processing transactions, given the necessity of timely execution of data acquisition and transfers in relation to response time. Within this simple concept lies an interesting conclusion: over 90 percent of the workloads that IT professionals deal with are I/O-intensive transactions. In other words, the majority of activities encapsulated within commercial processing consist of I/O-related activities.

The corollary to this observation is that the majority of workload planning and implementation centers on the I/O system that supports the processing infrastructure. A further realization is the importance storage architecture plays within the infrastructure and the tremendous impact putting storage on a network will have. Why? Because it changes the implementation strategy of the workload so dynamically that workload identification and definition becomes paramount. Otherwise, just throwing disparate

workloads within a storage networking infrastructure can be a hit or miss affair—one with a low probability of success. However, making it too simple and only implementing homogeneous workloads may not leverage the real value of storage network technology in how it supports disparate workloads.

The conclusion is that workload identification, definition, and planning are critical activities to the effective application of storage networks.



Total Transactions

Number of Users (application *n*) × Transactions (per user) = Workload Transactions (application)

WTransactions (application1) + WTransactions (application2) + WTransactions (application *n*) = Total Workload Transactions

Total Bytes Transferred

Average Bytes per Transaction (application *n*) × Workload Transactions = Application I/Os (application *n*)

I/Os (application1) + I/Os (application2) + I/O (application *n*) = Total Workload I/Os in Bytes

Total I/O Workload Rate

Total Workload I/Os in Bytes / Time Period = Total I/Os per Second Required

Total I/Os per Second / Block Size = Application I/Os per Second Required

Application I/Os per Second + System I/Os per Second = Total I/O Workload Rate

Figure 17-5. *Guidelines for identifying and defining workloads*

I/O Content and Workloads

Potential resource utilization and traffic arrival rates play a very important role in integrating the workload into an existing infrastructure, or in building a new infrastructure, for that matter. As discussed previously, the categorization and accumulation of workload attributes provided the definition and requirements for the workload. Continuing with our banking application example, Figure 17-6 illustrates the final inventory, categorization, and definition of the workloads. This allows us to accumulate a set of estimates of resource requirements for each workload and provide an estimated sum of the entire set of workloads. In order to accomplish this, we must look closely at each category of workload.

The resource requirement details are contained in the major areas of I/O activities, data organization, data paths, and user access. Putting these areas together forms a picture of the required infrastructure for the workload and, as we will see, a picture of the total infrastructure.

The Data Organizational Model

The most important aspect of the workload is the data organizational model it uses. In today's inventory of applications, both internally developed by IT application groups or IT implemented through application packages, the majority are based upon a relational model. The use of relational database technology (RDBMSs) defines the major I/O

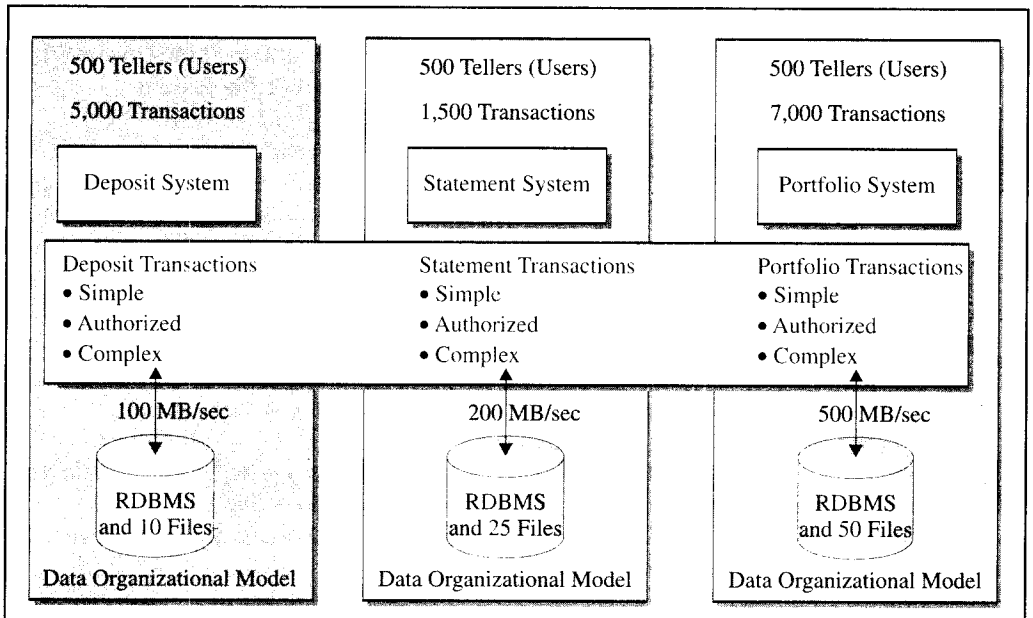


Figure 17-6. *The workload inventory, categorization, and definition of the Banking Application System*

attributes for commercial applications. This is a good thing because it provides those workloads that use RDBMSs a set of processing metrics for estimating I/O behavior and utilization. The use of the relational database has become so accepted and widespread its macro behavior is very predictable. Additional resource utilization is provided by the workload characteristics that define internal processing requirements such as caching, temporary workspace, and partitioning.

Don't make the mistake of overlaying the storage infrastructure too quickly—the consideration of recovery scenarios and requirements needs to be considered at the macro level first and then decisions made to handle specifics of the workload or particular subset of workload (for example, the specific application program). Therefore, we will have all our workload consideration taken into context before we make any conclusions about storage system features such as RAID, cache sizes, and recovery strategies.

The data organizational model provides the following sets of information to the workload behavior:

- **Block Size** The size of the block of data moving from the application transaction is dictated by the setup of the database. This can be either file or database attributes.
- **Partitioning** This attribute defines behavior regarding user access patterns in relation to the data itself. It influences decisions regarding the type of fault resiliency strategy for the workload (for example, RAID levels) and software recovery mechanisms.
- **Physical Design** The physical design of the database drives how a supporting file system is used. This is perhaps one of the most important attributes to consider given the type of database and its performance when using a file system.

Relational databases continue to prefer the use of raw disk partitions. This is important given the performance penalties one may encounter when performing I/O operations that are duplicated through the file system and passed off to the RDBMS system for further read/write operations. Referred to as the "double-write" penalty, this will be covered in more detail in Part VI.

- **Maintenance** Probably the most overlooked attribute in today's implementation-crazy environments, this defines workload behavior itself and includes requirements for backup/recovery, archival, and disaster recovery.

Regarding the use of RDBMS technology, the backup/recovery group not only includes basic backup operations but more important recovery operations that have to occur within a transactional basis. They must therefore include the necessary log files and synchronize the database to a predefined state.

This articulates a macro view of workload considerations and attributes supported by the analysis of the data organizational model.

User Access

User traffic defines the data hi-way attributes for the workload. Notice that we address this set of information prior to our topic on data paths. Obviously, we need to know the estimated traffic to understand the type, number, and behavior of the data hi-ways prior to associating that with the workload expectations. We basically need three types of information from the end users or their representatives, the application systems analysts. This information is shown in Figure 17-7 and consists of a number of transactions, the time-period transactions needed to execute, and the expected service level.

This activity is the most challenging. Trying to get either end users or their application representatives (for example, application system analysts) to estimate the number and type of transactions they will execute is difficult and, although important, can and should be balanced with a set of empirical information. Let's look at our banking application example and the deposit transactions. If we query the user community, we find that each teller handles approximately 100 deposit transactions per day. There are ten branch locations and, on average, five tellers per branch working an eight-hour shift. This results in an estimated 5,000 deposit transaction per day that need processing between the hours of 9 A.M. to 5 P.M.. Tellers expect their deposit transactions to be available during the entire eight-hour shift and have transactional response times ranging from subsecond processes to those lasting no more than five seconds.

However, the deposit transaction comes in three forms: simple, authorized, and complex. Simple requires an update to a customer table, authorized needs an authorized write to the permanent system of record for the account, which requires a double-write to access more than one database table. The complex, meanwhile, requires the authorized transactions but adds an additional calculation on the fly to deposit a portion into an equity account. Each of these has a different set of data access characteristics yet they belong to the same OLTP workload.

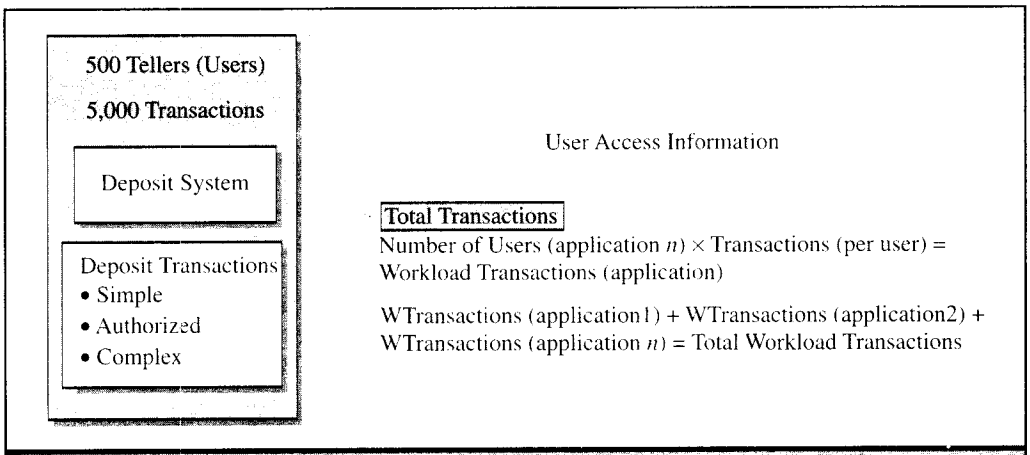


Figure 17-7. *User access information for workload analysis*

This is important given the potential for performance degradation if the appropriate data access points are not taken into consideration. Not all of this information can be expected to come from the end users, although we can estimate by analyzing the transaction history to determine the mix of subtransactions and thus plan accordingly. Consequently, the importance of fully understanding the transactional estimates generally goes beyond the end user or even the application developer's estimates, providing critical information regarding decisions in I/O configuration.

However, it's also important because it defines the I/O content of the transactions. I/O content is defined as the amount of user data transferred during an I/O operation. We have discussed previously that both bus and network transfer rates differ in the amount of bytes transferred. However, this is dependent on several variables including operating system, file system, and data partitioning in RDBMSs. Consequently, it is not always the case that the bus is full when executing an I/O transaction. Therefore, the more intensive the I/O content, the more throughput occurs, and the less time it takes to complete a transaction based on obtaining the amount of data needed.

An example is the deposit transaction set where the simple transaction only requires access to a database record within the database table. Even though this customer record only consists of a small amount of data, it still requires the server OS to execute an I/O operation. This design hardly makes the trip productive given the system overhead of the I/O operation using SCSI/PCI configurations or the larger amount of system process overhead necessary to leverage the tremendous payload of Fibre Channel. However, if the application design requires that each separate transaction is provided a single transfer packet or frame facilitated by the I/O operation, then the efficiency of the I/O must be considered to understand the necessary system requirements to support the I/O workload. Although this may provide an extremely fast I/O operation and subsequent response time, the amount of system resources dedicated to accomplishing this limits the number of transactions supported.

Using this example analysis of a single I/O per transaction, the number of transactions processed within the stated time period becomes very important. In the case of the example deposit application, the simple transactions make up the bulk of the workload. This leads to the conclusion that the capacity of the system is simply based upon the number of I/Os required for processing. Nevertheless, we know that the efficiency of the I/O system is highly inefficient and subject to non-linear response time service should any anomaly in increased transactions occur. Consequently, basing a capacity estimate simply on the number of I/Os a system is capable of is not necessarily the only metric required for balancing or building an effective and scalable I/O infrastructure.

The final piece of critical information regarding user access is the expected service level. This places our eventual, albeit simple, calculations into a framework that defines the resources needed to sustain the amount of operations for the OLTP workload. From our initial information, we find that there are two goals for the I/O system. First, the banking application's data needs to be available from 9 A.M. to 5 P.M. each workday. Second, the transactions should complete within a time frame ranging from subsecond response times to those lasting no more than five seconds. For the sake of our example, we will not address the networking issues until later in Part VI. However, it is important to note that although

the I/O can, and will, take up a great deal of response time factors, network latency issues do need to be considered.

By adding our service-level expectations to the mix, comparing these to user transactional traffic estimates, and considering the detail of the subset of deposit transactions, we find that the deposit transactions require a system that enables a transaction rate of 5,000 transactions/8 hours, or approximately 10 transactions per minute. That adds to our cumulative amount, which provides the capacity for the entire banking application. The cumulative analysis further concludes that a 500-MBps transfer rate is needed to successfully meet user expectations.

In addition, the infrastructure must support the 100 percent uptime that users expect in terms of data availability. This is a foundational requirement for meeting response time (for example, the data must be available to process the transactions). However, this begins to define the type of storage partitioning and structure necessary to provide this. Consider also that in meeting these goals the system must continue to process in the event of error—an area handled by RAID. In particular is the decision to provide level-1 or level-5 solutions necessary for efficient response time even during recovery processing.

Data Paths

Now, let's look at the data hi-way required for this workload. From an analysis of our first two categories we create a picture of the logical infrastructure necessary for the workload. By comparing the data organizational model (for example, the type of database and characteristics) and byte transfer requirements with something called the concurrent factor, we can begin to formulate the number of data paths needed to meet workload service levels. The concurrent factor, as mentioned previously during the user access discussion, determines the minimum, logical set of paths required to sustain our service level given the probability that all tellers at some point may execute deposit transactions simultaneously.

This calculation provides a more accurate picture of the resources needed to sustain the service level in real time. In reality, the probability that all the tellers will execute a deposit simultaneously is actually quite high and is calculated at 90 percent. Therefore, for each time period, 90 percent of the total tellers would be executing a deposit transaction. From our previous calculation, we estimate a mix of simple, authorized, and complex deposit transactions, which would be something like 80, 15, and 5 percent, respectively. This calculation provides for the average number of bytes transferred while taking into account the different I/O content of each transaction.

Figure 17-8 illustrates the accurate requirement for our workload. With this I/O workload analysis information, we can evaluate existing configurations to see if any of them will sustain the load, or develop a new model to configure a system that *will* sustain the load. Most likely you will want to do both. As in our sample case, we can see that we need a large amount of sustained workload I/Os for the entire business application. If we overlay this existing solution of direct-attached SCSI storage systems with capacities of no more than 50 MBps and arbitrarily-based device execution, it is likely this will be completely deficient in meeting workload service goals.

However, if we develop our own model, we find that a simple FC SAN solution with a 100-MBps rate through a frame-based transport will likely sustain the workload and support the concurrent transactional workload I/Os. If you add storage controller

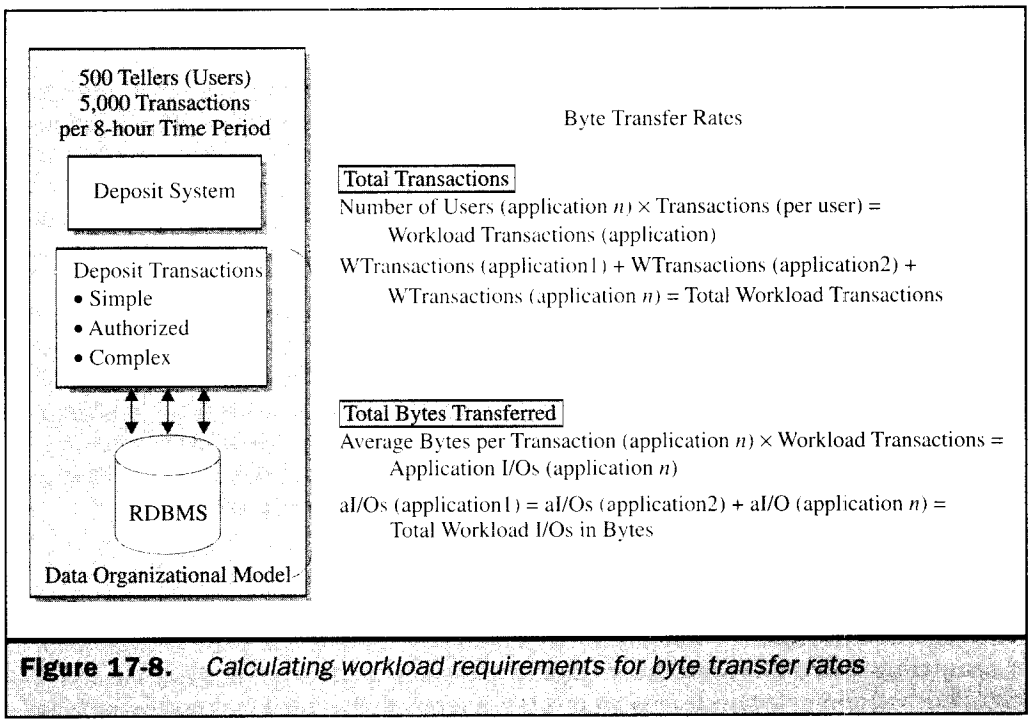


Figure 17-8. *Calculating workload requirements for byte transfer rates*

requirements of RAID, data maintenance, and recovery applications, we can estimate three data paths totaling 300MB burst rates. An estimated 240MB sustained rate thus will not only provide sufficient transfer rates but also a safe zone to compensate for peak utilization and growth factors before additional paths are required. Figure 17-9 illustrates a logical model built from our calculations.

Note *The 240MB transfer rate referred to previouly is calculated at 80 percent of total capacity (2,000 bytes × 80% = 1,600 bytes), which provides for switch and device latency.*

From this model, we can begin to assign specific technologies to find the most appropriate fit. Cost will certainly be a factor in determining the best solution for the workload. However, cost notwithstanding, the value of workload identification, definition, and characterization starts to become evident when moving the workload analysis into real implementations.

Considerations for I/O Workloads in Storage Networking

Implementing a solution that meets a set of business applications requires a workload analysis, as we have demonstrated. It then becomes necessary to analyze an existing configuration or develop a new configuration that can sustain the workload and meet user service levels. However, in the world of storage networking, IT faces additional challenges.

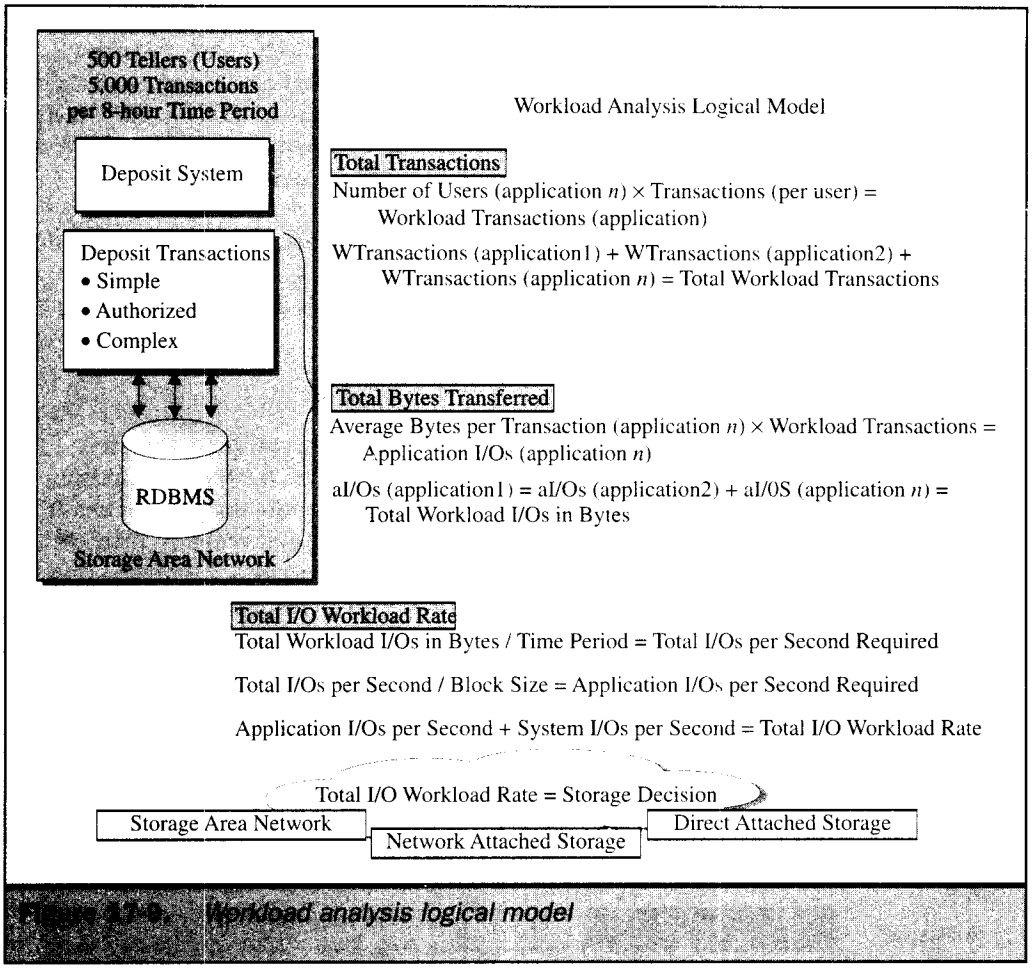


Figure 17-2 Workload analysis logical model

These issues focus on the increasing separation from the server complex, something which becomes an infrastructure in and of itself (such as a storage network infrastructure).

The most visible issue will be the discrete activities necessary to analyze the I/O infrastructure. Even though it must be taken in context with the business application and subsequent workload analysis, I/O becomes its own entity when moved into a storage networking model. Therefore, it is imperative to plan, model, and configure storage networking solutions with sufficient detail and I/O specifics that are consistent with the workloads you are supporting.

In some cases, the I/O workload is only part of a larger capacity planning exercise. Consequently, I/O workload planning needs to become integrated into existing planning activities and be driven from workload estimates that have been previously discovered. If that is the case, the minimum elements of I/O workload planning, as indicated in the earlier section "Deposit Application—Workload Attributes," are still required and should be discovered even though they may come from an existing capacity planning exercise.

The ability to support multiple workload types cannot be overlooked. This will probably be the most challenging activity required. Certainly SANs and, in some cases, NAS solutions will be required to support workloads that are disparate in their characteristics and processing service levels. This makes workload analysis imperative when planning the I/O infrastructure so workloads can be identified and sized before implementation takes place. This is also necessary in order to defend existing configurations when planning upgrade strategies.

This leads us to the next two chapters in this part of the book, which concern the application of storage networking solutions and technologies. Given that we have already identified, described, and characterized our workloads at a macro level, the right solution should become evident.

SAN or NAS Solutions

With current information, you can decide pretty early whether you're looking to leverage an IP-based NAS solution or a more comprehensive FC SAN configuration. The decisions based upon workload analysis will provide accurate direction when it comes to choosing a storage networking solution. Keep in mind, it will also provide accurate justification if staying with current direct-attached storage solutions looks to be the best alternative. Figure 17-10 offers some guidelines on initial workload analysis when macro decisions need to be made regarding SAN, NAS, or direct attached.

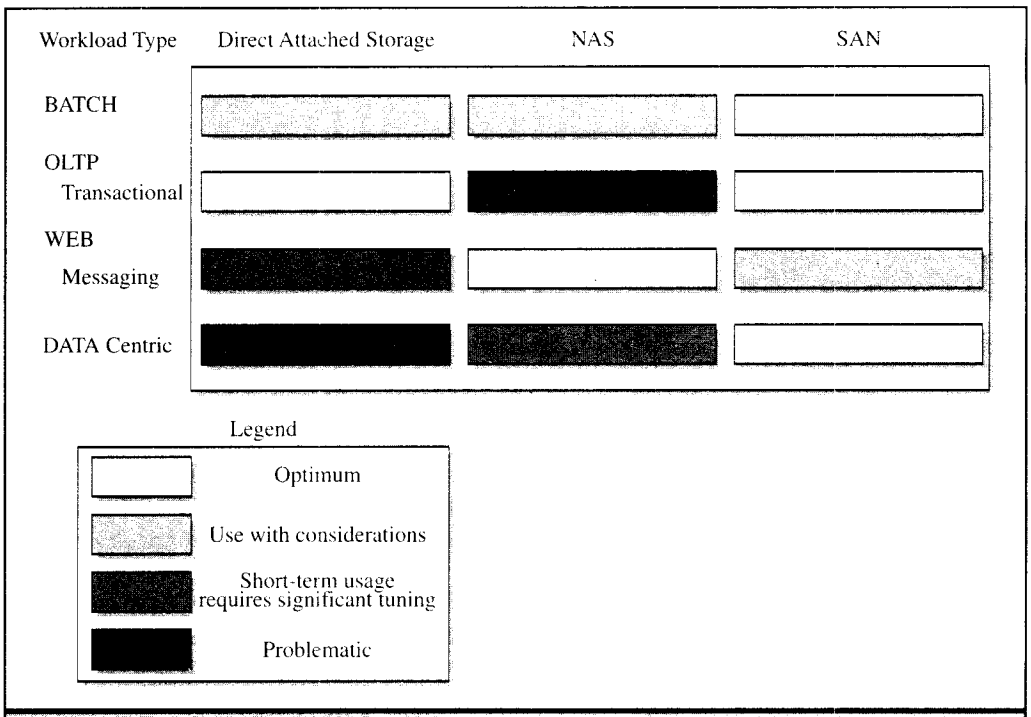


Figure 17-10. *Workload solutions matrix*