3

PROJECT MODELING

Project modeling is done as an investigation of the design constraints of a project. In a SAN project, the I/O behaviors of the host systems and applications that will use the SAN are examined for extremes and trends. A good project model takes into account the complete range of tasks the SAN is expected to perform and is based on existing systems or reasonable estimates. A good project model does not have to take a long time to complete, provided assumptions can be made about the expected results to limit the amount of examination required. Models allow the designer to verify the resulting SAN behaviors without the risk of moving critical applications and host systems to the SAN. Start modeling your project with assessments of storage and I/O workload requirements.

3.1 Storage Requirements

Storage requirements can be determined from general rules and knowledge of the target SAN type.

General Rules for Sizing Storage

The expected data set size, or the data set size plus the expected growth of the data set if it already exists, determines the storage requirements. Application users, developers, and database administrators should have an idea of the type of data being stored and characteristic sizes, so they are good sources of sizing information. In a new application, if the typical record size in a database or the size of commonly stored data for an application can be determined, then simply multiply the size by the total expected starting number of records to determine the total amount of storage. If growth can be predicted on the basis of the number of users or the number of records, then storage can be sized for expected growth as well. A last resort for sizing information can simply be a guess based on general information about the new application.

Requirements by SAN Type

Depending on the type of SAN being implemented, assumptions about the amount of storage required can eliminate much of the storage size analysis.

STORAGE CONSOLIDATION

To determine a storage consolidation SAN's requirements, look at the target host systems, add up all the storage in use on the host systems, and add the growth rates of the data on those host systems. A storage consolidation SAN more efficiently accommodates the storage space growth rates of the host systems because all storage space in the SAN is available to all host systems.

NAS REPLACEMENT

A NAS replacement is similar to a storage consolidation SAN. To determine the amount of storage needed, add up all the storage

space in use on the existing NAS and look at the expected growth rate of the data stored on the NAS systems.

NAS replacement SANs that implement tape backup require a different storage sizing method due to the unlimited total storage of removable media devices. Size a NAS replacement SAN for tape backup by determining the number of tape drives required to service the backup workload. The number of tape drives depends on all of the following:

- The number of concurrent backups
- The amount of data to be backed up
- The amount of data each tape will store
- The amount of time available for system backups

A complete discussion of backup system sizing would be a digression here; there are several good books on this topic. As a simple rule of thumb, use one tape drive for each concurrently backed up file system. Unless the data sets are small, avoid backing up data from multiple sources onto a single tape. Resource contention and possible restore conflicts can occur if multiple backups for different host systems are on the same tape.

CAPACITY PLANNING

For capacity-planning SANs, determine the size of storage by looking at the data set sizes of systems likely to use the SAN. A capacity-planning SAN that supports data warehouse extraction, transformation, and load (ETL) processing has greater storage requirements than a capacity-planning SAN that supports OLTP. This difference reflects the generally smaller overall sizes of high-performance OLTP applications. To estimate a good storage size, look at the typical storage size and the growth requirements of the target systems. If a given system requires 200GB of storage today and doubles in size every six months, and another like it is deployed every six months as

well, then a reasonable sizing estimate for a six-month capacity-planning solution is 800GB.

EXPERIMENTAL

Experimental SANs require little or no analysis of storage sizing. Because the purpose is pure experiment, the typical size of a system from your environment makes the best test case. If the experimental SAN's storage size is too small, the investigations cannot yield enough information. If the storage size is too large, there is obvious waste.

NEW PROJECT

When attempting to size a SAN for a new project with little or no relation to any existing system, it is important to find out as much as possible about the new application. Investigation of the expected record size and the number of records goes a long way toward determining the required storage space. System and application overhead requires an additional 10 to 25 percent of space. The expected growth of the application data is also important when planning for system growth.

3.2 I/O Size Requirements

To discover the characteristic I/O size, gather information on data access. If possible, also gather information on data access patterns with respect to read-and-write ratios. If the access patterns cannot be discovered by looking at application specifics—such as logs, in the case of Web servers, or transactions, in the case of OLTP servers—then gather the data from host system tools. On UNIX systems use the *sar* command to look at the raw disk I/O behavior. On Windows

NT systems, use the *perfmon* command. This data will shed some light on typical I/O sizes and the access patterns.

The raw data can be processed to show some additional interesting I/O behaviors. With a few simple rules of thumb applied to the raw data and the processed data, the analysis can provide all of the information necessary for the design of the SAN. This process yields a set of boundaries for the SAN design goals with respect to the application. At completion, the analysis provides requirements for maximum bandwidth, maximum IOPS, and the amount of storage space. The next two sections take a detailed look at the examination of I/O workload on host systems.

With definite requirements in hand, hardware and software can be selected and integrated to meet specific application needs. Evaluation follows to determine whether the SAN meets expectations or requires any additional changes.

3.3 I/O Assessment and Analysis Tools

The best way to look at I/O behaviors and performance is to look at system tools on the hosts that run the applications. An examination using system tools provides a top-down view of the I/O subsystem from the host system's perspective. A higher level view of the I/O behaviors can sometimes be extracted from an application, such as a relational database management system (RDBMS), but not all applications have the ability to report this data. Further, there is no consistent way to extract the data for the applications that report I/O statistics.

Because of this inconsistency and because system tools tend to be more consistent in their availability and data set measurement, it is best to start with the system tools themselves. The system tools provide a distilled version of the application I/O behavior at the device

level. Any additional application-level device abstractions are lost, but the raw I/O behaviors will still show through in the analysis.

It is possible to perform an analysis of the I/O system from the storage device point of view in a bottom-up fashion. This method does not have the problems of an application-level analysis because of the common availability of useful statistics on almost all intelligent storage devices. Information gathering takes place with device-specific methods because standards for the contents of the data set and the data extraction method are not quite complete.[1] New storage device management standards will make data gathering from storage devices more complete and consistent, so that all devices can provide the same basic utilization and performance data. Implementation is in various stages depending on the hardware and software vendors, the products in use, and the chosen device management method.

In general, put off device analysis until the host system analysis is complete. The storage device analysis has greater depth and narrower scope, and it requires more effort to perform. Delaying this analysis enables a more focused approach on the storage devices, whose greater amount of storage-specific I/O data can easily swamp the investigator.

A few simple scripts written in Perl or a shell language can quickly examine UNIX hosts that have the *sar* utility. *sar* is a very useful tool to use, available on almost all UNIX operating system variants. The *sar* data set and output are quite consistent from UNIX to UNIX. The data available from the Windows NT *perfmon* command can also be processed fairly easily from its logged format.

A quick look at the *sar* man page on your UNIX host system will provide details on the timing and amount of data gathered. On most

1. The Fibre Alliance is continually updating the Fibre Channel MIB for SNMP, and the SNIA has developed a complete storage management and information standard, the Common Information Model, based on the Distributed Management Task Force Web-Based Enterprise Management.

UNIX host systems, the data is the past week's-worth of system data. A simple spreadsheet analysis of the data can provide information on maximum system bandwidth and IOPS. The analysis can also show patterns of usage throughout a day or several days. Once the script is run on each host system, the collected data can be examined and combined with data collected from other host systems, if necessary, to provide a complete snapshot of the host system's workload.

The get_io.sh script in Example 3.1 performs two functions:

1. It gathers bandwidth and IOPS data from a host system.
2. It outputs data files from *sar* input data for analysis in a spreadsheet.

The analysis of the data set gathered from the script is performed by putting the comma-separated-value output files of each data type (bandwidth or IOPS) for each day assessed into a spreadsheet. The data can then be graphed versus time in order to visualize the I/O behaviors of the host system under evaluation in the modes of bandwidth, IOPS, and I/O size. The visualization of the data reveals some significant I/O parameters for the SAN design, such as maximum bandwidth utilization, maximum IOPS utilization, workload windows, workload consistency, and characteristic I/O sizes. Additional mathematical analysis may be of use if the visualization of the data provides poor insight into the I/O behaviors of the analyzed host system, but usually this is not required.

The fairly simple script in Example 3.1 takes data collected by the *sar* utility and creates twenty-minute aggregated data points of bandwidth and IOPS from the host system perspective on all I/O channels combined. See Figure 3.3 (on page 61, top) for an example of the output of the get_io.sh script. The two sets of output files from the script can also be combined to find out the typical I/O size of the application being examined over these intervals.

EXAMPLE 3.1. The get_io.sh shell script

```
#!/bin/sh
# get_io.sh
# Gather aggregate bandwidth and IOPS data from a host's sar data files
# Gather bandwidth data from sar archives
day=1
for sarfile in `ls /var/adm/sa/sa[0-2]*`
do
  shour=0
  ehour=0
  min=0
  while [ $shour -le 23 ]
  do
    ehour=`expr $shour + 1`
    interval=0
    # Divide each hour into 3 parts because the data is in 20-minute
    # intervals
    while [ $interval -le 2 ]
    do
      case "$interval" in
        0)
        blocks=0
        sum=0
        # Extract the data from a sar archive file and
        # sum the blks/s column
        for blocks in `sar -d -f $sarfile -s $shour:00:00 -e
        $shour:20:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
        | awk '{print $5}'`
        do
          sum=`expr $sum + $blocks`
        done
        # Clean up any old temp files, then compute bandwidth
        rm -f /usr/tmp/bcfile
        echo $sum " / 2 / 1024" >> /usr/tmp/bcfile
        echo quit >> /usr/tmp/bcfile
        bw=`bc -l /usr/tmp/bcfile`
        # Store the bandwidth result in a csv file
        echo $bw >> /usr/tmp/bw_$day.csv
        # Report the bandwidth result
```

EXAMPLE 3.1 (*continued*). The get_io.sh shell script

```
echo "Bandwidth is" $bw "MBps"
;;

1)
blocks=0
sum=0
for blocks in `sar -d -f $sarfile -s $shour:20:00 -e
$shour:40:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
| awk '{print $5}'`
do
  sum=`expr $sum + $blocks`
done
rm -f /usr/tmp/bcfile
echo $sum " / 2 / 1024" >> /usr/tmp/bcfile
echo quit >> /usr/tmp/bcfile

bw=`bc -l /usr/tmp/bcfile`
echo $bw >> /usr/tmp/bw_$day.csv
echo "Bandwidth is" $bw "MBps"
;;

2)
if [ $shour -eq 23 ]
then
  break
fi
blocks=0
sum=0
for blocks in `sar -d -f $sarfile -s $shour:40:00 -e
$ehour:00:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
| awk '{print $5}'`
do
  sum=`expr $sum + $blocks`
done
rm -f /usr/tmp/bcfile
echo $sum " / 2 / 1024" >> /usr/tmp/bcfile
echo quit >> /usr/tmp/bcfile
```

EXAMPLE 3.1 (*continued*). The get_io.sh shell script

```
        bw=`bc -l /usr/tmp/bcfile`
        echo $bw >> /usr/tmp/bw_$day.csv
        echo "Bandwidth is" $bw "MBps"
        ;;

      esac
    interval=`expr $interval + 1`
    done
    shour=`expr $shour + 1`
  done
  day=`expr $day + 1`
done

# Gather IOPS data from sar archives
day=1
rm -f /usr/tmp/bcfile
for sarfile in `ls /var/adm/sa/sa[0-2]*`
do
  shour=0
  ehour=0
  min=0
  while [ $shour -le 23 ]
  do
    ehour=`expr $shour + 1`
    interval=0
    while [ $interval -le 2 ]
    do
      case "$interval" in
        0)
        ios=0
        sum=0
        # Extract the data from a sar archive file and
        # sum the r+w/s column
        for ios in `sar -d -f $sarfile -s $shour:00:00 -e
        $shour:20:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
        | awk '{print $4}'`
        do
         echo $ios "+ \\" >> /usr/tmp/bcfile
```

EXAMPLE 3.1 (*continued*). The get_io.sh shell script

```
    done
  echo 0 >> /usr/tmp/bcfile
echo quit >> /usr/tmp/bcfile
  # Compute the IOPS
iops=`bc -l /usr/tmp/bcfile`
    # Store the result in a csv file
    echo $iops >> /usr/tmp/ios_$day.csv
    # Report the result
    echo "IOPS are" $iops
  # Clean up any old temp files
rm -f /usr/tmp/bcfile
    ;;

    1)
    ios=0
    sum=0
    for ios in `sar -d -f $sarfile -s $shour:20:00 -e
    $shour:40:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
  | awk '{print $4}'`
    do
     echo $ios "+ \\" >> /usr/tmp/bcfile
    done
echo 0 >> /usr/tmp/bcfile
echo quit >> /usr/tmp/bcfile
iops=`bc -l /usr/tmp/bcfile`
    echo $iops >> /usr/tmp/ios_$day.csv
    echo "IOPS are" $iops
rm -f /usr/tmp/bcfile
    ;;

    2)
    if [ $shour -eq 23 ]
    then
      break
    fi
    ios=0
    sum=0
    for ios in `sar -d -f $sarfile -s $shour:40:00 -e
```

EXAMPLE 3.1 *(continued).* The get_io.sh shell script

```
        $ehour:00:30 | egrep -v "IRIX|sun4|HP-UX|AIX|,|^[0-2]"
        | awk '{print $4}'`
        do
         echo $ios "+ \\" >> /usr/tmp/bcfile
        done
     echo 0 >> /usr/tmp/bcfile
     echo quit >> /usr/tmp/bcfile
     iops=`bc -l /usr/tmp/bcfile`
        echo $iops >> /usr/tmp/ios_$day.csv
        echo "IOPS are" $iops
     rm -f /usr/tmp/bcfile
        ;;

      esac
    interval=`expr $interval + 1`
    done
    shour=`expr $shour + 1`
  done
  day=`expr $day + 1`
done
```

The get_iosize.pl script in Example 3.2 takes pairs of bandwidth and IOPS output files from the script in Example 3.1 and uses the simple equation

I/O size = Bandwidth (KB/s) / IOPS

to generate the typical I/O size over the same intervals.

The output of this script will add a bit more detail to the analysis of the application and host system. See Figure 3.3 (on page 61, bottom) for an example of the output from the get_iosize.pl script. The graphic analysis of the data shows patterns and anomalies. The more regular the patterns look in the graphical analysis in terms of IOPS, bandwidth, and I/O size, the more likely it is that the conclusions drawn from the patterns will be useful. Less consistent graphs indicate

EXAMPLE 3.2. The get_iosize.pl shell script

```
#!/usr/local/bin/perl
#
# get_iosize.pl
# Find the characteristic I/O size from the output of get_io.sh script
$i=1;
while ( $i <= 7 ) {
    # Open the result file for output from this script
    open (OUTFH, ">>/usr/tmp/iosize_$i") || die "Can't open file, $!\n";
    # Open and read the bandwidth and IOPS output csv file pair
    open (BWFH, "/usr/tmp/bw_$i") || die "Can't open file, $!\n";
    @bwinfo=<BWFH>;
    close (BWFH);
    open (IOPSFH, "/usr/tmp/ios_$i") || die "Can't open file, $!\n";
    @iopinfo=<IOPSFH>;
    close (IOPSFH);
    # Make sure the number of data collection intervals
    # in each file matches or quit
    if ( $#bwinfo != $#iopinfo) {
        printf "The files for day $i don't match. Exiting\n";
        exit;
    }
    $j=0;
    # Divide the bandwidth in KBytes by the number of IOPS
    # to get the I/O size
    while ( $j <= $#bwinfo) {
        if ( @iopinfo[$j] != 0) {
           $iosize = $bwinfo[$j] * 1024 / $iopinfo[$j];
           } else {
           $iosize = 0;
           }
        # Report the I/O size result and record it in an output file.
        printf "Typical IO size is $iosize\n";
        printf OUTFH "$iosize\n";
        $j++;
    }
    close (OUTFH);
    $i++;
}
```

more variable system usage, making the sizing task more difficult. Pattern uncertainties can lead to overconfiguration and waste of resources in the SAN design.

3.4 Analyzing Key Application I/O Characteristics

With some data in hand, it is time to look at several examples of application complexes in order to determine the characteristics of the host systems and applications. A comparison of each application complex with the expected SAN type shows the configurations that work best and the settings that need to be applied.

When looking at the output of the I/O assessment tools used to gather data, apply local environment rules of thumb to the analysis. If the analysis of the data seems to indicate an oddity, then the local behaviors of the users or supporting systems will also need to be evaluated. For example, an oddity may be a moving peak usage time period on a system that runs the same workload every day. Additional analysis can help explain the unexpected behaviors and facilitate a more accurate sizing of the design. For example, a data warehouse batch job that starts daily at different times due to variable size of the input data set is one situation in which a moving peak usage time may be observed.

NAS Replacement SAN for an NFS Server

In the first system for examination, a SAN replaces a NAS server running NFS, as shown in Figure 3.1. The NAS server provides archived business intelligence in order to avoid retrieval of tape backups for recently processed data sets. Retrieval of data sets occurs in the case of processing errors, processing failures, or additional processing needs. The server holds several weeks of data, and the data set sizes are gradually growing. Specifically, the NAS server has been growing at a rate of approximately 100 percent every twelve months. To find

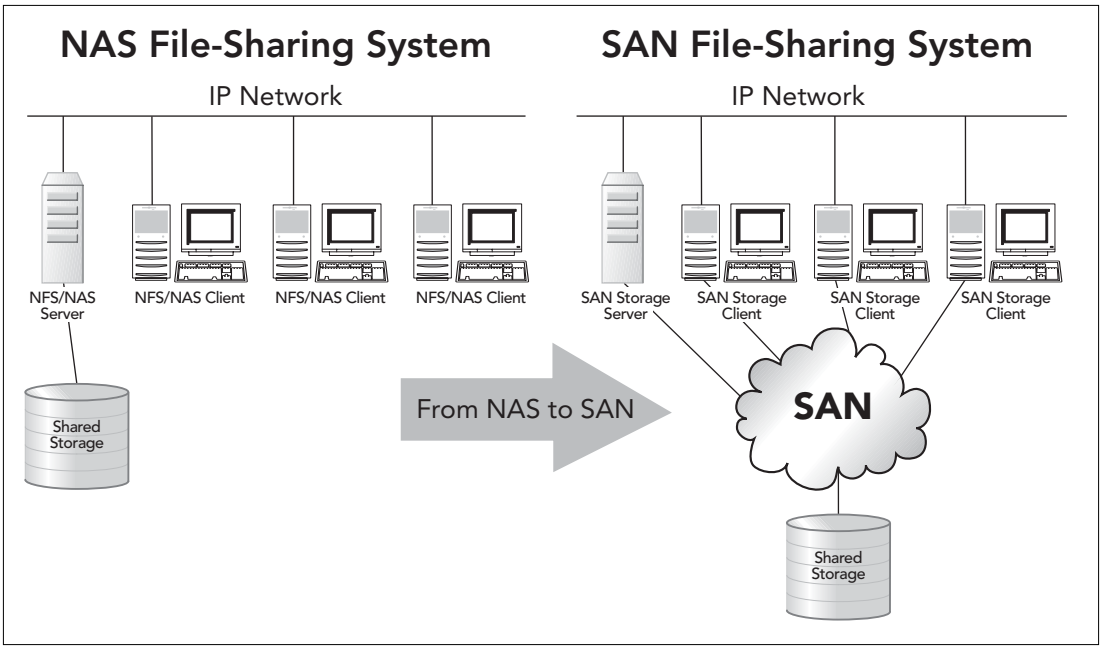

FIGURE 3.1

NAS replacement SAN for file sharing

the growth rate, determine how much storage has been added over the past twelve months and make a few quick inquiries about expected uses over the next twelve months. Now we understand the storage requirements for the SAN system.

Using the output of the scripts in Examples 3.1 and 3.2, it is possible to create several graphs of the data. The graphs show a few interesting characteristics of the NFS server. Figure 3.2 shows the bandwidth usage for the entire system over the period of a week.[2] This aggregate display of bandwidth shows that the application does not consume much bandwidth. Only a fast SCSI or slower device interconnect has trouble with the peak bandwidth of the system. This fact

2. A week may not be enough data, so further data gathering may be required. In this case, one week is enough.

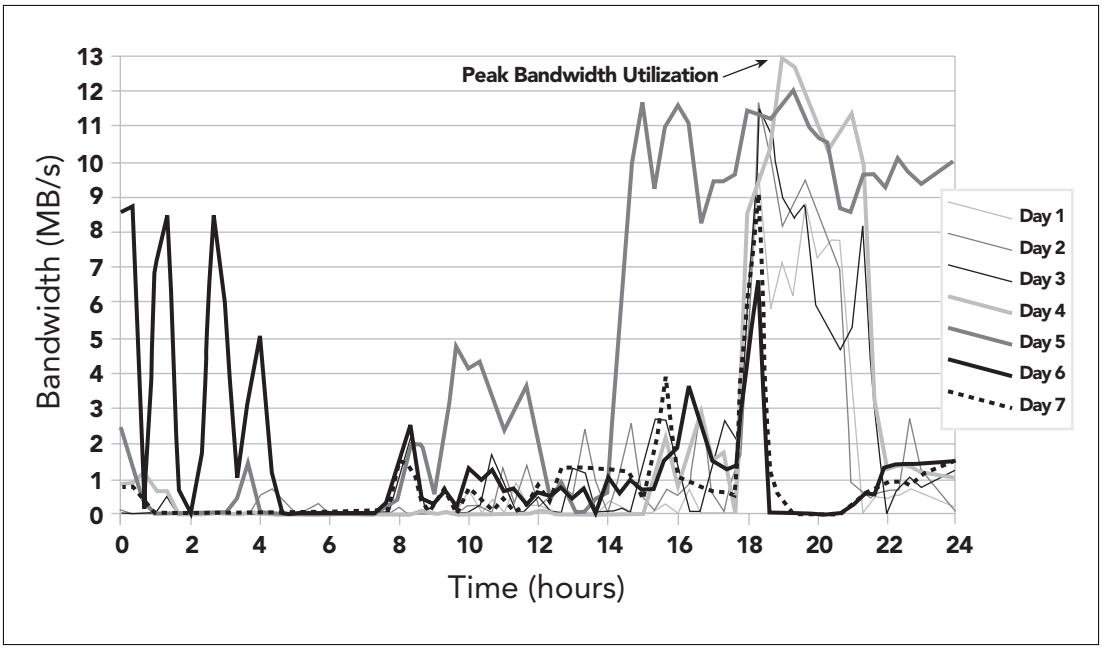

FIGURE 3.2

NFS server bandwidth versus time

gives a great deal of flexibility when choosing the SAN infrastructure and topology, because Fibre Channel or any other interconnect can easily handle this bandwidth.

Figure 3.3 shows the performance of the NAS server. The first graph in Figure 3.3 shows that the system will have an IOPS load close to, but not exceeding, the lower region of the IOPS performance scale for a single HBA, which Table 2.2 shows to be 500 IOPS (see page 36). This load allows for flexibility in the SAN configuration because the configuration requires only one HBA to service the IOPS and bandwidth load. Obviously other factors such as multipath I/O will affect the final number of HBAs used, but performance is not an issue based on the likely choices of hardware and the application requirements.

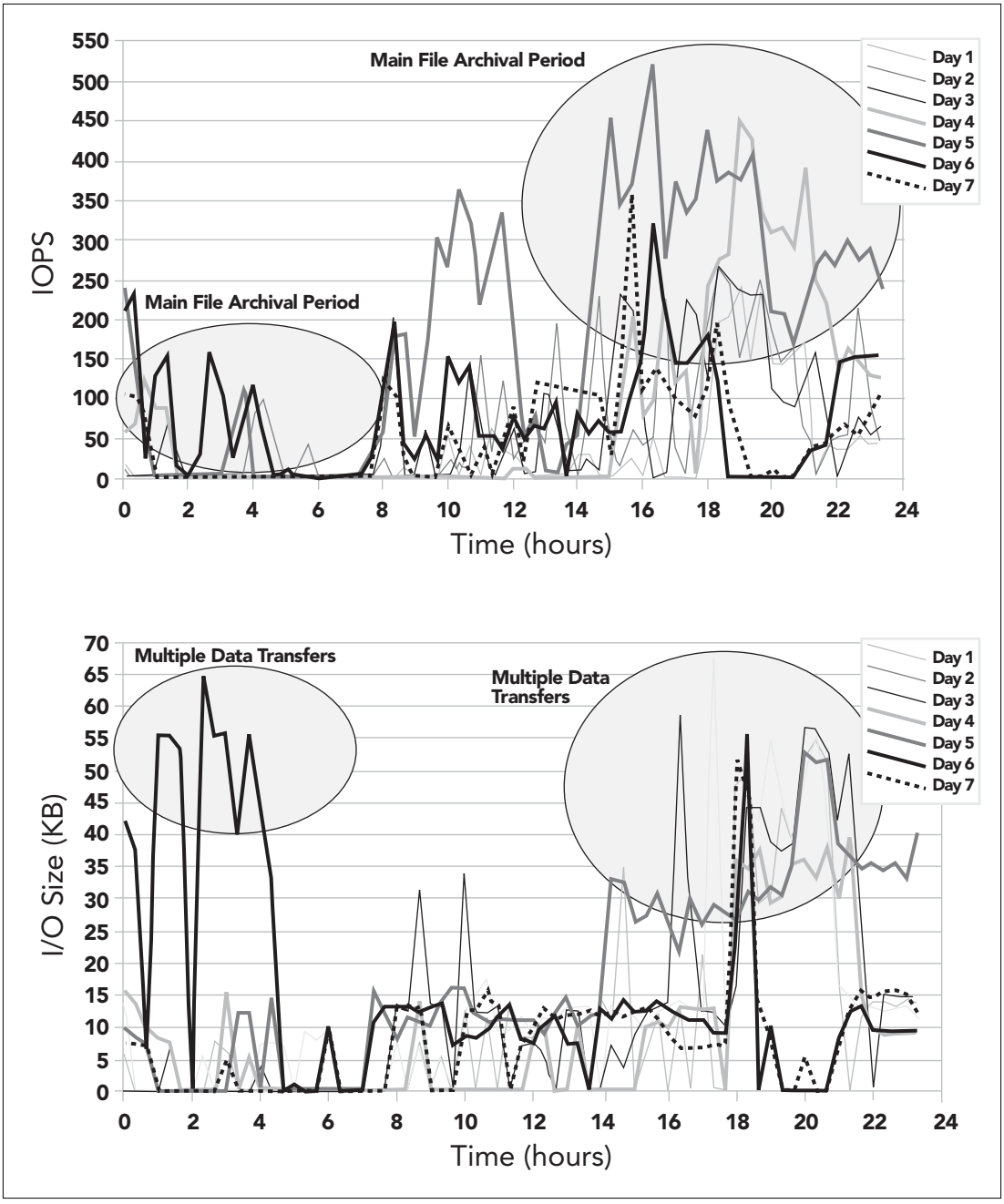

FIGURE 3.3

Top: NFS server IOPS versus time. *Bottom:* NFS server I/O size versus time

The second graph in Figure 3.3 shows I/O size with respect to time for the period of a week. The I/O size graph shows that the system performs I/O in the 12KB to 16KB size characteristic of NFSv2. Peak I/O sizes can be larger than the NFS transfer size, because this is a systemwide analysis; the larger I/O sizes are approximate multiples of the typical NFS transfer size. Based on knowledge of the application, it can be assumed that during these times, multiple data transfers cause the aggregate I/O size to appear larger than expected. A quick inspection of the system processes during one of these periods shows that the assumption of multiple data transfers is correct.

No real oddities have been found from the analysis of the NAS server, and the parameters for the design have been obtained. Before defining the I/O model created to test the SAN design, a few more system types should be examined.

Storage Consolidation of a Data Warehouse (ETL) System

Data warehouse (ETL) staging systems make good examples of systems that are appropriate for storage consolidation. Figure 3.4 shows the systems.

The host systems perform daily ETL tasks for a data warehouse system in a large customer service organization. The data provides general information about groups of customers in order to help provide more focused services to individuals in those groups. The storage devices are initially empty and then filled as projects arise. ETL systems perform mostly memory-intensive data transformation tasks. The I/O load on these systems consists mostly of file writes of the transformed data and data transfers to and from the host system.

STORAGE SPACE REQUIREMENTS

The amount of storage required for these systems is the sum of the following factors:

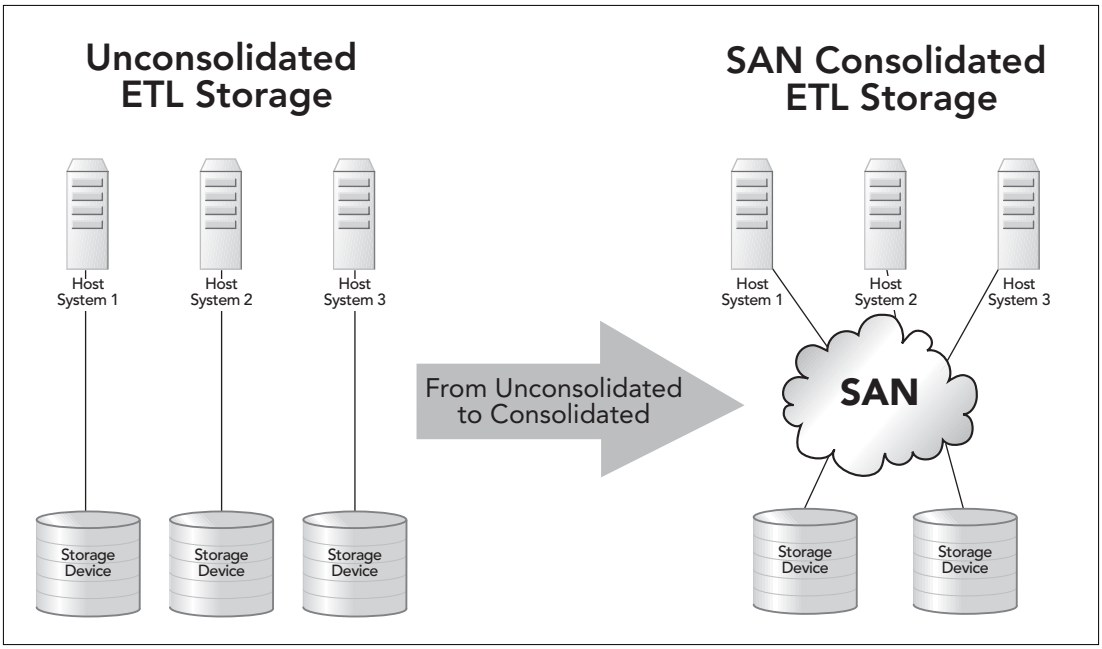

FIGURE 3.4

Storage consolidation SAN for data warehouse

- The space to receive the raw business intelligence files
- The scratch work space for file transformation
- The output area for the processed files
- The archive area (if any)

To gather this information, look at the existing host systems.

The storage growth of the consolidated host systems is the sum of these two requirements:

- The amount of storage needed to contain data sets as they grow
- The amount of storage needed to accommodate additional data transformation output by any new processes

There is an additional potential reduction in excess storage from redeployment of unused storage using the shared pool in the SAN.

An examination of the three data warehouse staging hosts shows that the amount of storage grows about 1TB every six months. Each of the three systems has 1TB of storage (a total of 3TB), and each system will need an additional 2TB of storage each in the next twelve months. Therefore, the storage consolidation SAN requires 3TB of storage now plus 1.5TB for the first six months of growth. This configuration actually allows the hosts to grow exactly as if they had local storage. But storage now can be allocated to each host, as needed to accommodate uneven growth patterns.

This configuration requires the same amount of storage, but the timing of the deployment is different. The free pool of storage in the SAN can be equal to 1.5 times the size of a single host system's storage instead of 3 times the storage that a single host needs for growth. As a result, the storage consolidation SAN requires more frequent storage acquisitions to achieve the same growth rate, but allows the acquisitions to be smaller and the idle storage on the systems as a group to be smaller, because deployment is easier and more flexible.

PERFORMANCE REQUIREMENTS

An examination of the three ETL systems using the get_io.sh and get_iosize.pl scripts (Examples 3.1 and 3.2) sets the performance requirements for the ETL storage consolidation SAN. The bandwidth graphs in Figure 3.5 show widely varying usage from host system to host system.

Host 1 has the highest aggregate bandwidth and the least consistent usage timing even though the bandwidth utilization is mostly consistent. Hosts 2 and 3 have more consistent usage patterns and do not have extremely high bandwidth requirements. If it is decided to compromise on absolute bandwidth or if the peak workload on Host

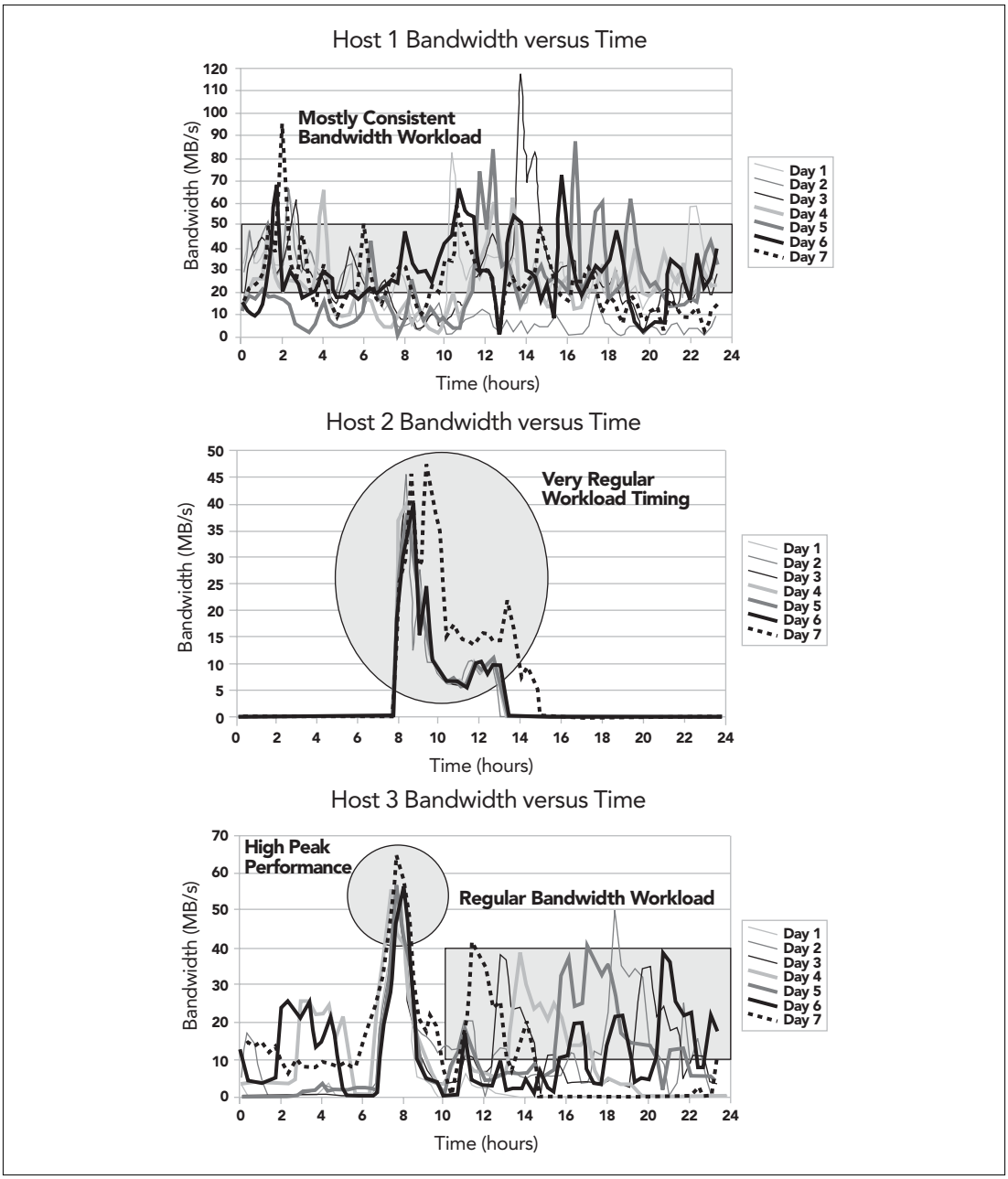

FIGURE 3.5

Three consolidation candidate host systems, bandwidth versus time

1 can be relocated to one of the other hosts during a less busy time, then the bandwidth requirement for this SAN can be set at 100MBps.

The IOPS requirements we see in Figure 3.6 show an average-to-high channel demand and an average overall demand for the combined requirements of these host systems.

This information, along with the preceding bandwidth information, enables us to select the following:

- The number of channels required per host system
- The number of channels per storage device
- The number of paths through the fabric per host system

The aggregate I/O size value for these host systems is not nearly as useful in this case due to the high number of overlapping jobs running on each host system. A full assessment of the characteristic I/O size for the systems requires a detailed application analysis of each job. It is not necessary to perform the assessment at this time because the other characteristics are much clearer, and they provide the necessary amount of information about the I/O behavior.

Analyzing I/O in Other SAN Types

Examining the I/O behaviors of a system for capacity planning or a new project is difficult because the system does not exist before the deployment of the SAN. These types of SANs do have some similarities to a storage consolidation SAN and can be assessed in the same way. The results of the assessment will have less certainty but still allow for the setting of SAN parameters that will hopefully achieve a good SAN design.

If a company deploys a new data warehouse application every three to six months, with the same amount of storage and layout, then it is useful to deploy a capacity-planning SAN using several of that application's host system types. Examine one of the prior data warehouse

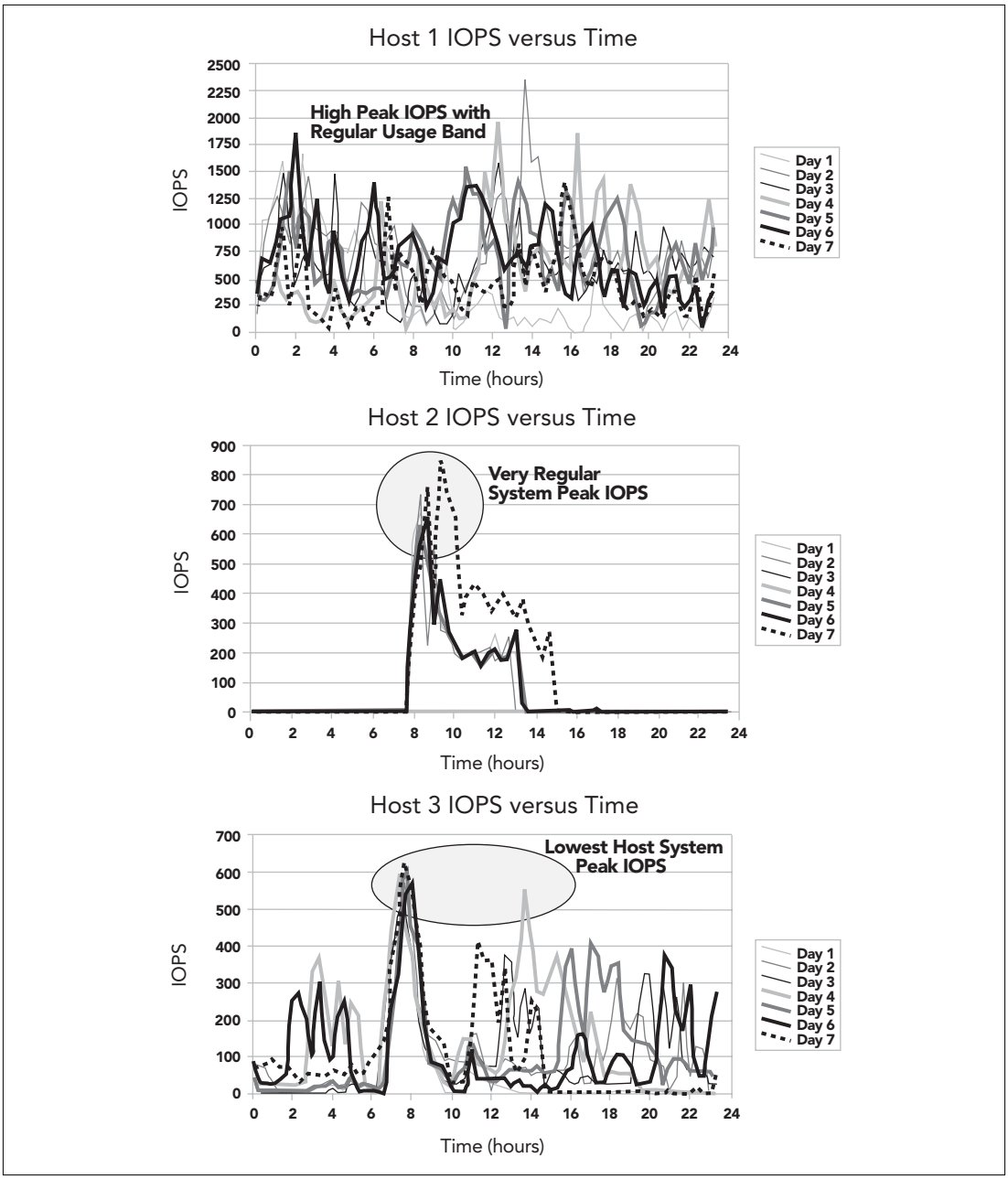

FIGURE 3.6

Three consolidation candidate host systems, IOPS versus time

host systems and then use it as a template for the host systems in the capacity-planning SAN. The advantage of this method, as opposed to just deploying some number of application host systems with directly attached storage, is that the capacity-planning SAN can accommodate changing requirements without any new physical work on the host systems or their storage.

If a new data warehouse application is expected to require twice the typical amount of storage that the template application host system has, the storage can easily be accommodated in the capacity-planning SAN by making changes to the SAN configuration that logically reassigns storage. If deploying a group of host systems with directly attached storage where one host system needs an increase in storage size, the host must either have storage physically reconnected from some other host system or benefit from a new storage acquisition. This reconnection potentially leaves one host system short of disk space, takes longer than a configuration change, or requires an additional storage purchase, leaving other storage underutilized. The savings in labor alone will make this a worthwhile use of a SAN.

Use the same tools for examination of the template host system, but accept more variability in the design. The bandwidth assessment of a typical midsize data warehouse system in Figure 3.7 shows peak host bandwidth in the average range.

Take the per-channel I/O bandwidth into consideration when deciding the type of I/O channel to use and the required number per system. This choice can push the per-channel I/O bandwidth from the average range to the high range using four or fewer I/O channels per host system. Fewer than four low-bandwidth I/O channels can constrain peak bandwidth, but this design choice needs some justification because there is a potential for reduced performance.

The second graph in Figure 3.7 shows the IOPS behavior of the data warehouse template system. The system has a peak IOPS performance characteristic that is in the average region for a host, but the

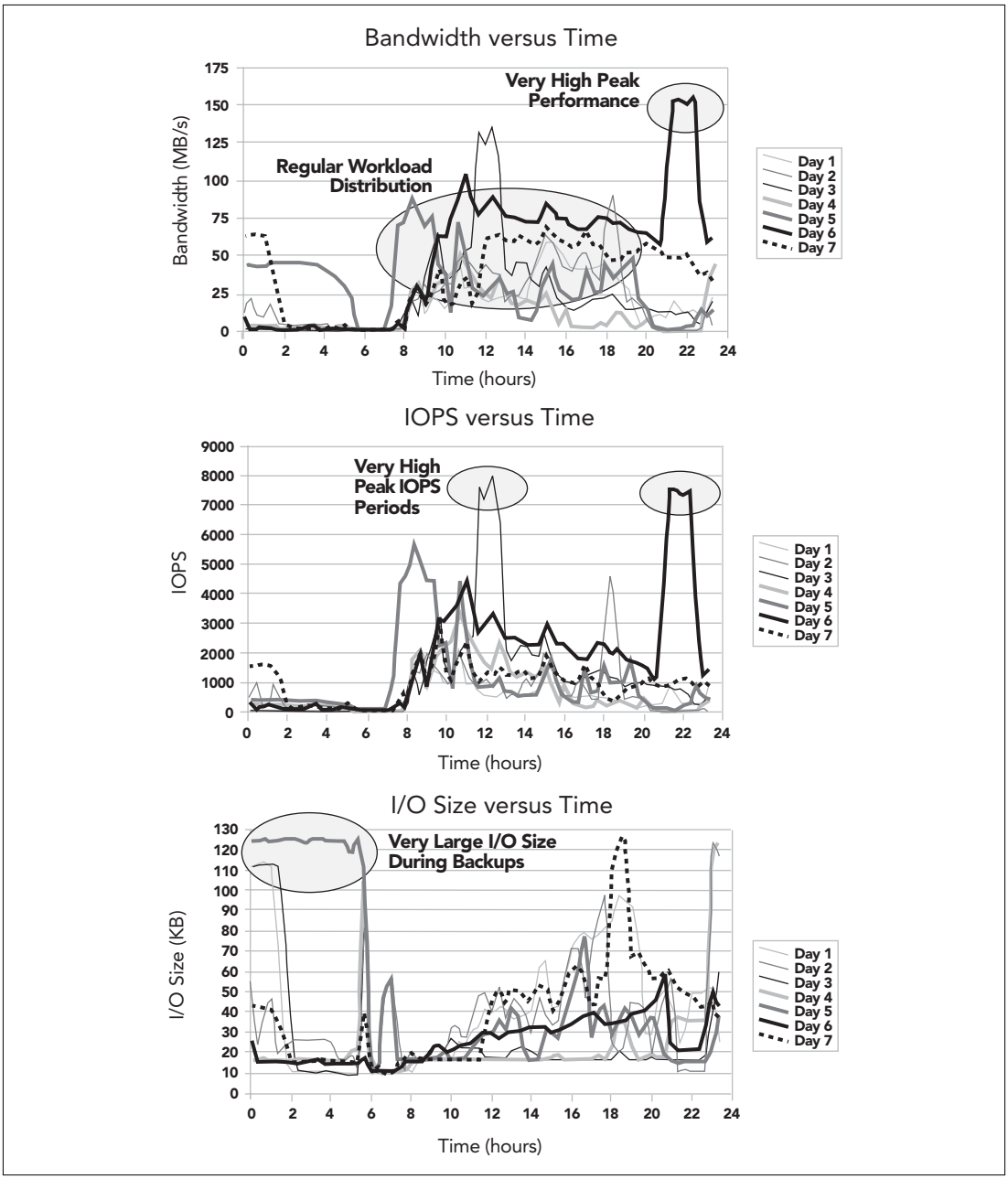

FIGURE 3.7

Data warehouse SAN candidate, host system I/O analysis

per-channel IOPS performance moves into the high performance region with fewer than six I/O channels available.

The system bandwidth and IOPS analysis shows that the peak bandwidth occurs at a different time than peak IOPS. A quick look at the I/O size during these times can rule out obvious errors in the IOPS or bandwidth assessments. In Figure 3.7, the I/O size during the peak bandwidth period is indeed larger than during the peak IOPS period. Another interesting characteristic to note is that the peak I/O size occurs during a low IOPS time but still requires a significant amount of bandwidth.

It is now possible to determine the number of I/O channels and the expected performance of the host systems on this capacity-planning SAN, based on the IOPS and bandwidth assessment. For example, one I/O channel should be allocated per host system for every 50MBps of bandwidth or every 1000 IOPS. Two I/O channels should be added for every 50MBps of bandwidth or 1000 IOPS if multipath I/O is required. The I/O size information helps validate the assessment and gives some useful information for creating an I/O model for design verification testing.

3.5 Simplified SAN Application I/O Models for Verification

Now that the performance assessment of the template applications and host systems has been completed, use the information gathered from the assessment to model the expected behaviors of the host systems. The verification model can be simple and should try to re-create the I/O behaviors of the system being modeled. Not all I/O behaviors need to be built into the model, because modeling everything is extremely complex and time-consuming. The verification model tries to emulate peak performance for the chosen I/O characteristics. The verification model can also test failure modes and evaluate SAN behaviors while working with specific features of the SAN.

Modeling the NAS Server Replacement

The I/O model for the NAS server replacement SAN in Figure 3.1 (page 59) should emulate the archival processes that the NAS server currently services. This application simultaneously transfers several large files to the NAS server, and the model for the file transfers can be quite simple. The tester places a set of test files on one client host system or more and then writes a simple set of scripts that transfers these files to and from the new SAN file server.

The tester then measures the transfers for bandwidth performance and checks for reliability. Performance should be evaluated and assessed from several places in the SAN. Ideally, the throughput of the NAS replacement SAN has been measured from the client, the server, and the fabric devices that make up the SAN.

Testing of the failure cases in the NAS replacement SAN includes these tasks:

- Simulating device failures during data transfers
- Powering off fabric devices
- Rebooting host systems
- Unplugging cables in a controlled manner to evaluate behaviors under failure or maintenance conditions

These tests provide a better understanding of the failure cases and may uncover problems in maintenance methods or the design.

Modeling the Data Warehouse ETL Consolidation SAN

A model for the storage consolidation SAN in Figure 3.4 (page 63) is more complex than the NAS replacement SAN test model. The systems in the storage consolidation SAN will use the fabric-attached storage for file creation in addition to reads and transfers, which differs from the dedicated data transfer use of the NAS replacement

SAN. The I/O model must include file creation, reads, and writes. Modeling must also include an approximation of the timing of the processes.

The first step is the creation of a few simple scripts that create, read, and write files. These scripts can then be grouped together to simulate I/O behaviors of the systems being consolidated on the SAN. Example 3.3 shows a Perl script that randomly reads a file.

This simple script performs a specified number of random 1KB reads throughout a specified file. A similar script in Perl can randomly write updates to a file, as shown in Example 3.4.

The writer.pl script inserts an all-zero, 1KB update into a specified file at a random location. It is easy to modify the size and content of the update for customization.

Much simpler scripts can also create files. Because a new file will be sequentially written with the typical I/O size of the application in most cases, a file creation script can use the UNIX system tool *dd*. Example 3.5 shows a *dd* command to write an 800MB file in 8KB-size blocks.

In Example 3.5, the parameters are:

- Input file (if)
- Output file (of)
- Block size (bs)
- Number of IOPS (count)

To create a file of any size with any I/O size, change the block size and the count.

Use a wrapper script to run the scripts or file creation command numerous times. Simulate CPU processing time with delays in the wrapper. A wrapper script that simulates a load operation in a data warehouse is shown in Example 3.6.

EXAMPLE 3.3. A random file reader script (reader.pl)

```
#!/usr/local/bin/perl
#
# reader.pl
# Perform random reads of a file
#

# The first argument to the script is the file name
# The second argument to the script is the number
# of reads to perform
$file = $ARGV[0];
$count = $ARGV[1];

# open the file to be read and find its size
open(FH, $file) || die "Can't open $file\n";
seek(FH, 0, 2);
$filesize = tell(FH);
close(FH);

srand(time);

open(FH, $file) || die "Can't open $file\n";

# perform 1KB reads of the file at random offsets
# $count times
while ( $i <= $count) {
    $fpos = int(rand $filesize) + 1;
    read(FH, $dump, 1024);
    $i++;
    }

close(FH);
printf "Done reading file $file\n";
```

EXAMPLE 3.4. A random file updater script (writer.pl)

```
#!/usr/local/bin/perl
# writer.pl
# Perform random updates of a file
#
$LOCK_SH = 1;
$LOCK_EX = 2;
$LOCK_NB = 4;
$LOCK_UN = 8;
# The first argument to the script is the file name
# The second argument to the script is the number of writes to perform
$file = $ARGV[0];
$count = $ARGV[1];

# Make a 1KB buffer of zeros
$buf="0" x 1024;

# open the file to be read and find its size
open(FH, $file) || die "Can't open $file\n";
seek(FH, 0, 2);
$filesize = tell(FH);
close(FH);

srand(time);

# open and lock the file for writing
open(FH, "+<$file") || die "Can't open $file\n";
flock(FH, $LOCK_EX);

# perform 1KB writes to the file at random offsets $count times
while ( $i <= $count) {
    $fpos = int(rand $filesize) - 1;
    seek(FH, $fpos, 0);
    print FH $buf;
    $i++;
    }

flock(FH, $LOCK_UN);
close(FH);
```

EXAMPLE 3.5. Simple file creation using *dd*

```
dd if=/dev/zero of=/fs1/file01 bs=8192 count=100000
```

EXAMPLE 3.6. Data warehouse load simulation wrapper

```
#!/bin/sh
# Data warehouse load I/O model

# create 10 2GB files sequentially
dd if=/dev/zero of=/fs1/file01 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file02 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file03 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file04 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file05 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file06 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file07 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file08 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file09 bs=8192 count=250000
dd if=/dev/zero of=/fs1/file10 bs=8192 count=250000

# read and write previously created
# simulated catalog file at random
# 250000 times simultaneously in
# 10000 I/O chunks with 30 seconds
# of simulated calculations between chunks
i=1
while [ $i -le 25 ]
do
  reader.pl /fs1/simucat 10000 &
  writer.pl /fs1/simucat 10000 &
  i=`expr $i + 1`
  sleep 30
done
```

These tools simulate the I/O workload of the ETL systems on the storage consolidation SAN. Use the same I/O workload simulation for failure mode and maintenance evaluation by simulating failures and performing maintenance tasks while the model runs.

Model the I/O behaviors of the systems on a capacity-planning SAN for midsize data warehouse applications using the same set of tools. In addition, use a nonrandom read command, because data warehouse systems tend to scan large tables sequentially. Example 3.7 shows a *dd* command that performs a simple sequential read.

This command reads 8KB blocks of the file created in Example 3.5. In this case the command simply reads and discards the data because the data is not needed for anything else.

The four simple I/O workload components just described can be assembled to simulate the I/O behavior of the data warehouse systems in almost any mode. Simulation of the staging, loading, and querying of the data warehouse system requires several wrapper scripts in order to combine these I/O workload driver tools. The wrapper scripts would be variations on Example 3.6 and can also be very simple.

In a capacity-planning SAN where zone changes can be frequent due to unknown initial system configurations, evaluation of zoning changes is particularly interesting. Make changes to the capacity-planning SAN configuration while running the I/O model to determine the exact behavior of the systems, fabric devices, and storage devices.

Create an experimental SAN I/O model out of the same components used for the capacity-planning SAN in order to exploit the SAN performance characteristic or behavior. Running several copies of the sequential reader at the same time will drive up bandwidth on the SAN. Multiple copies of the random reader and writer scripts will create high IOPS loads. Additional combinations of the I/O work-

EXAMPLE 3.7. Simple sequential read using *dd*

```
dd if=/fs1/file01 of=/dev/null bs=8192 count=100000
```

load components can simulate the interesting workloads found in most environments.

Model a SAN for a new project in the same fashion as an experimental SAN. The SAN for a new project has more clearly defined performance expectations that facilitate a more accurate model of the expected I/O workload. The SAN does not have to be intentionally stressed, but it can be evaluated with an I/O model that creates the expected performance level for the host systems and applications that will be using the SAN.

3.6 Final Project Definition

Use the information from assessments of the system and application I/O characteristics to define the project parameters. The type of SAN and the I/O behaviors point to the performance parameters and host system behavior expectations. The project definition also takes into account failure modes and other operational considerations such as dynamic SAN reconfiguration. Use the definition as a yardstick for measuring whether or not the goals of the SAN have been accomplished.

NAS Replacement SAN Definition

The definition of the design for the NAS replacement SAN is fairly simple. (See Figure 3.1 on page 59.) The parameters that drive the design are the bandwidth required for the application and multipath I/O channel infrastructure that prevents a systems outage in the case of a single I/O channel failure.

The bandwidth required is minimal, with a peak measured usage of 13MBps. This means that any single Ultra SCSI or Fibre Channel I/O interface can meet the bandwidth requirement for this SAN. The

multipath I/O channel configuration requires a minimum of two channels per host system or storage system. Because two I/O channels provide from 72MBps to 200MBps, depending on the selected type, the bandwidth requirement can easily be met. The SAN requires 1TB of storage to accommodate its current data set and an additional 0.5TB of storage to accommodate six months of growth. All of the interconnections between fabric devices, if any are necessary, will also require two I/O channels.

Storage Consolidation SAN Definition

The definition of the storage consolidation SAN project is more complicated due to higher performance requirements and more tradeoffs to accommodate the different host systems and applications. (See Figure 3.4 on page 63.) Fabric bandwidth is one of the defining parameters of the SAN. Although only one of the systems has bandwidth requirements in even the average range for a single host system, the bandwidth requirements of all the systems being consolidated must be serviced concurrently on the SAN fabric. The storage consolidation SAN requires a multipath I/O channel configuration for failure resilience and load balancing, if possible. This SAN supports a data warehouse ETL workload, so the SAN includes a data movement tool that improves data transfer times and removes load from the consolidated host system's IP networks.

Aggregate bandwidths of 400MBps in the fabric and 100MBps per host system are necessary in this SAN. This performance should be adequate given a more evenly balanced workload across all of the systems. A balanced workload eliminates the spikes in the peak usage of the one host system with needs that exceed 100MBps. The SAN requires at least a pair of Ultra SCSI II controllers (or faster) to meet the SAN host system performance requirements. Because of the multiple controllers required for bandwidth, the multiple channel I/O failover and load balancing configuration requirement can also be met. The storage space required for this SAN is 4.5TB at the start.

This allocation provides storage space for the current data set on all three host systems, plus the capability to grow all three host systems by 0.5TB or any individual host system by up to 1.5TB on an immediate need basis.

Capacity-Planning SAN Definition

The bandwidth and flexibility requirements of the host systems characterize the project definition for the data warehouse capacity-planning SAN. The requirements also include a multiple I/O channel configuration for host system and storage device resilience. Features include data replication for scalability and disaster recovery that support the business-critical data warehouses targeted for the SAN.

Each host system requires 200MBps of bandwidth for storage devices, and the fabric must support the aggregate traffic of four host systems. These requirements mean that the fabric will require 800MBps of bandwidth to support the concurrent load of the host systems. The storage devices must also support the 200MBps from each host system either individually or as a group, depending on their size and the final allocation to each system. Two Fibre Channel I/O channels can meet the bandwidth and multiple I/O channel failover needs of each system. Only two I/O channels require high per-channel IOPS performance, so a trade-off that installs more I/O controllers to meet the IOPS needs of the host systems may be necessary. A higher number of the same, or lower, performance I/O channels can meet the IOPS needs of the host systems and provide a lower per-channel IOPS solution. However, the lower performance I/O channels might not meet the bandwidth needs.

The storage space requirement for the capacity-planning SAN is 1TB per deployed system or 4TB total to start. It is likely that there will be data growth, so some expansion capacity can be built into the SAN. To provide for the data replication scheme, the SAN requires installation of some additional fabric connectivity in order to increase available bandwidth without slowing the data warehouse

application systems usage. Chapter 4 shows how the host design parameters defined here can translate into useful SAN designs.

Other SAN Types

The SAN design definition for a new project is set to meet the requirements of the project. A good strategy for setting these requirements involves finding applications or host systems that may have performance and host system needs that meet the requirements of the new project. Then apply the parameters of those systems to the new project SAN.

The design for an experimental SAN meets the testing requirements of the SAN. For example, if performing IOPS-limit evaluations, then use a low number of channels and a high IOPS–capable storage device. If testing failover under stress, then specify at least one alternate I/O channel. Test SAN limits and behaviors by constraining the I/O parameter to be tested and then observing what happens to the host systems, storage devices, and fabric devices when an extreme load is placed on the SAN.

3.7 Summary

Using the tools described in this chapter, I/O analysis can be completed and the SAN project type can be determined. All planning aspects of the SAN project should now be finalized. Next, the design stage can begin: time to select the components, create I/O models for validation, plan the physical integration, and start evaluating trade-offs.