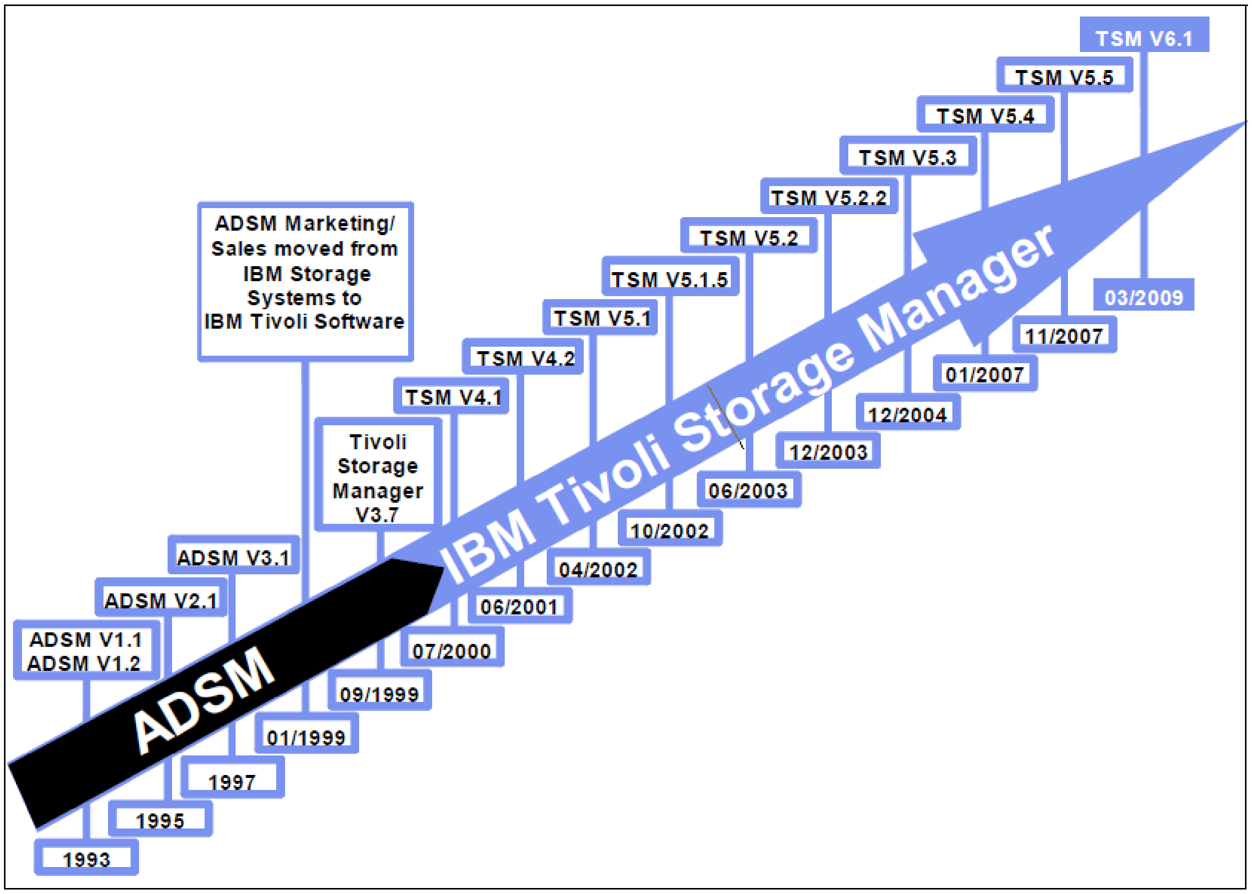
Tivoli Storage Manager OVERVIEW

March 23, 2011

Contents

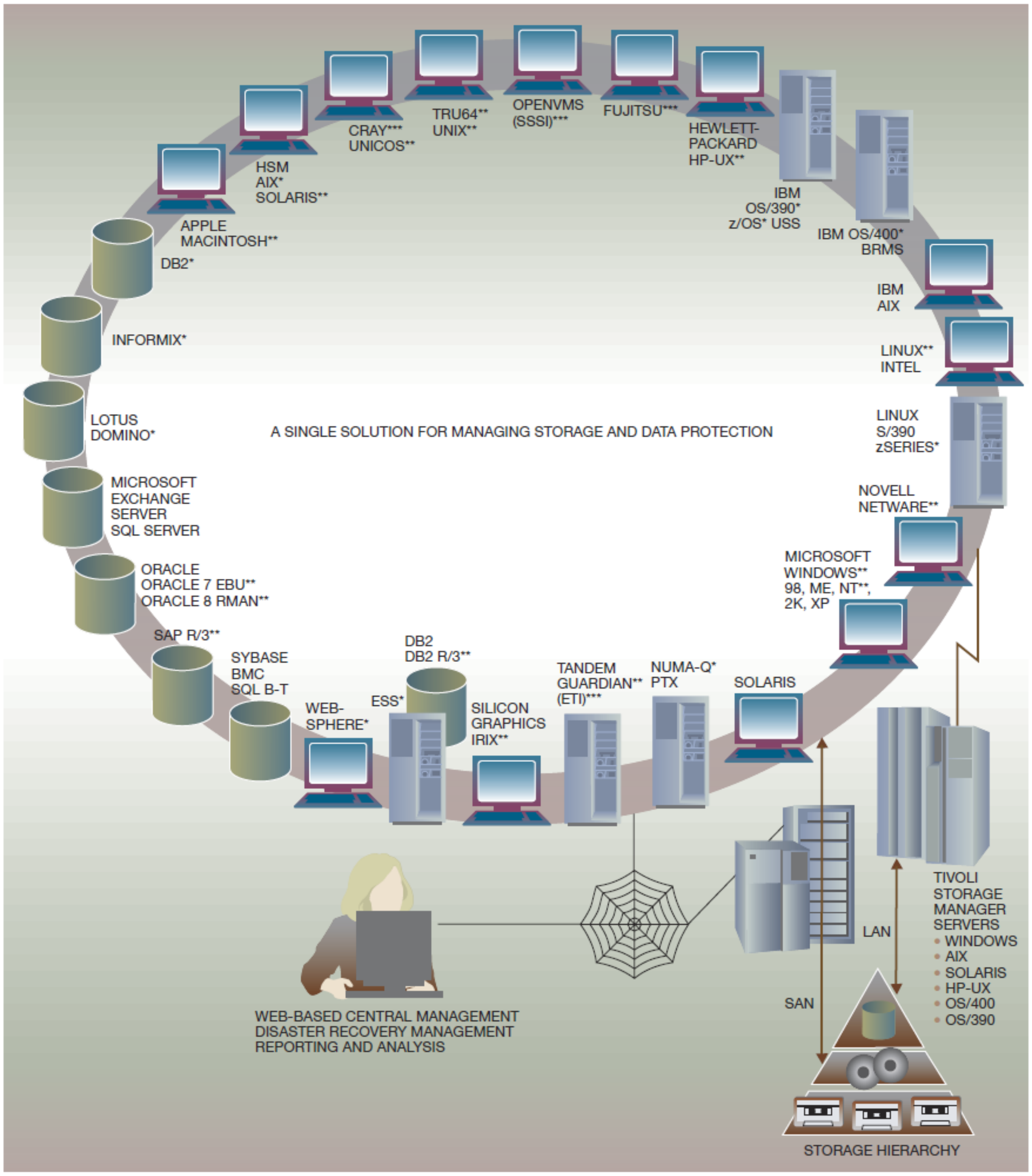
1. History

IBM Tivoli Storage Manager started life as ADSTAR Distributed Storage Manager (ADSM). Figure shows the release time line for the various versions of ADSM, and its subsequent and present name, IBM Tivoli Storage Manager, up to the current version.



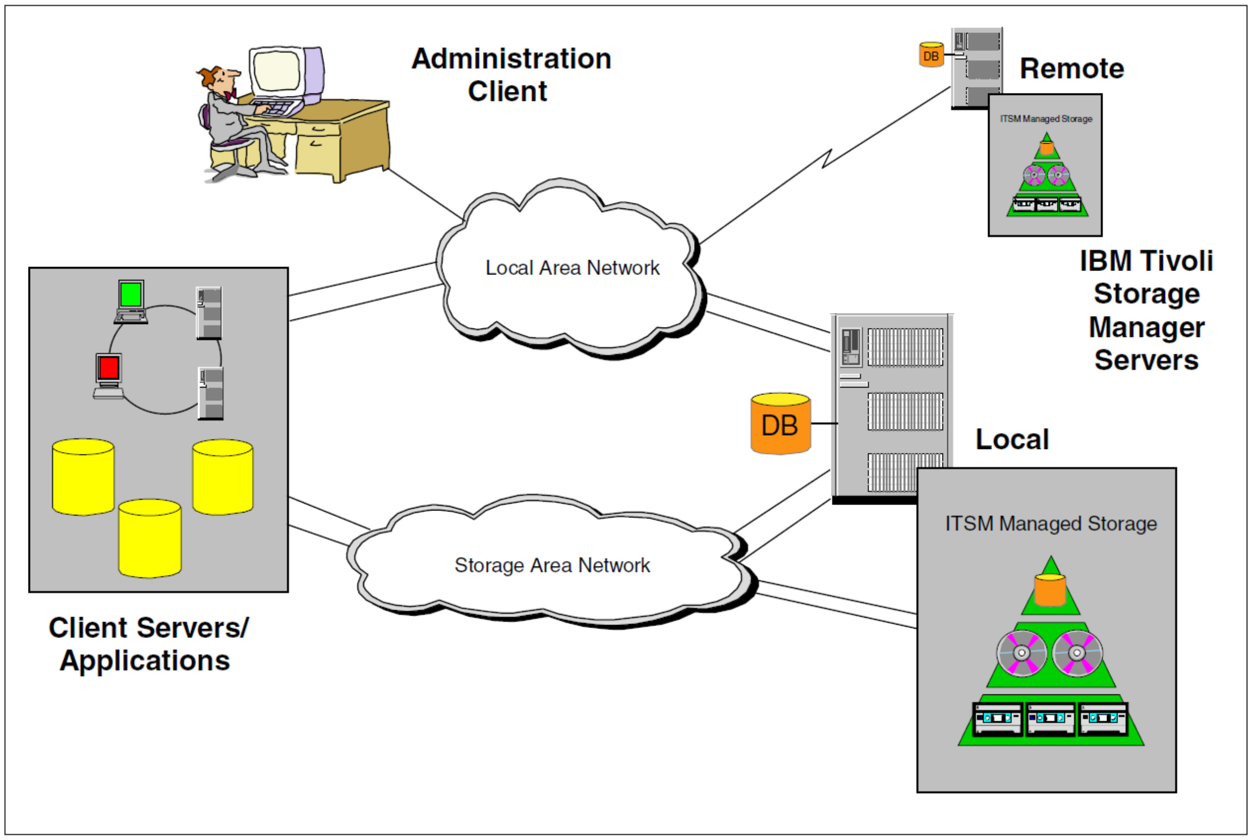
2. Supported systems

IBM Tivoli Storage Manager is a powerful storage software suite that addresses the challenges of complex storage management in distributed heterogeneous environments. It protects and manages a broad range of data, from workstations to the corporate server environment. More than 44 different operating platforms are supported, using a consistent graphical user interface.



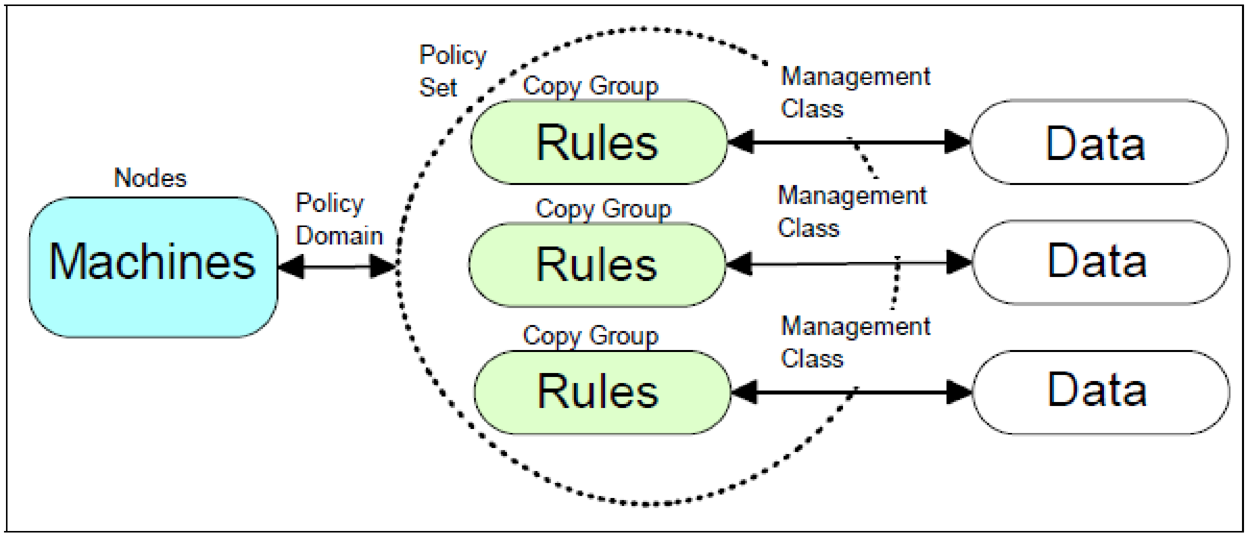
3. TSM Components

Tivoli Storage Manager is implemented as a client-server software application, consisting of a Tivoli Storage Manager server software component, Tivoli Storage Manager backup-archive client, and other complementary IBM and vendor software products.

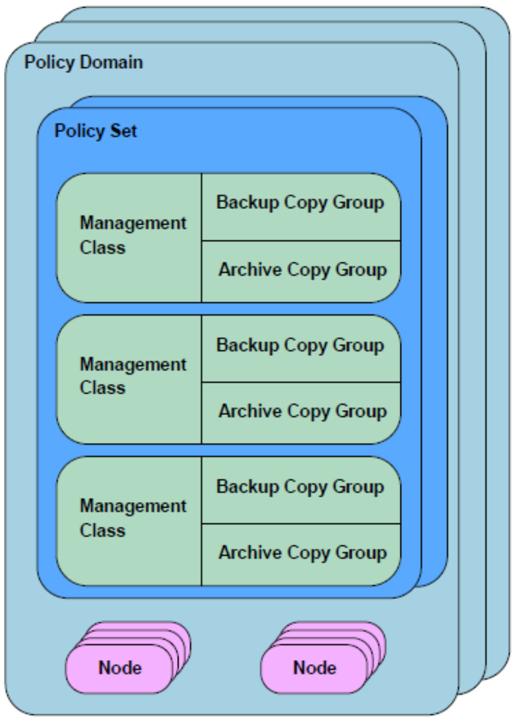


3.1 TSM Server

3.1.1 Policy-based management

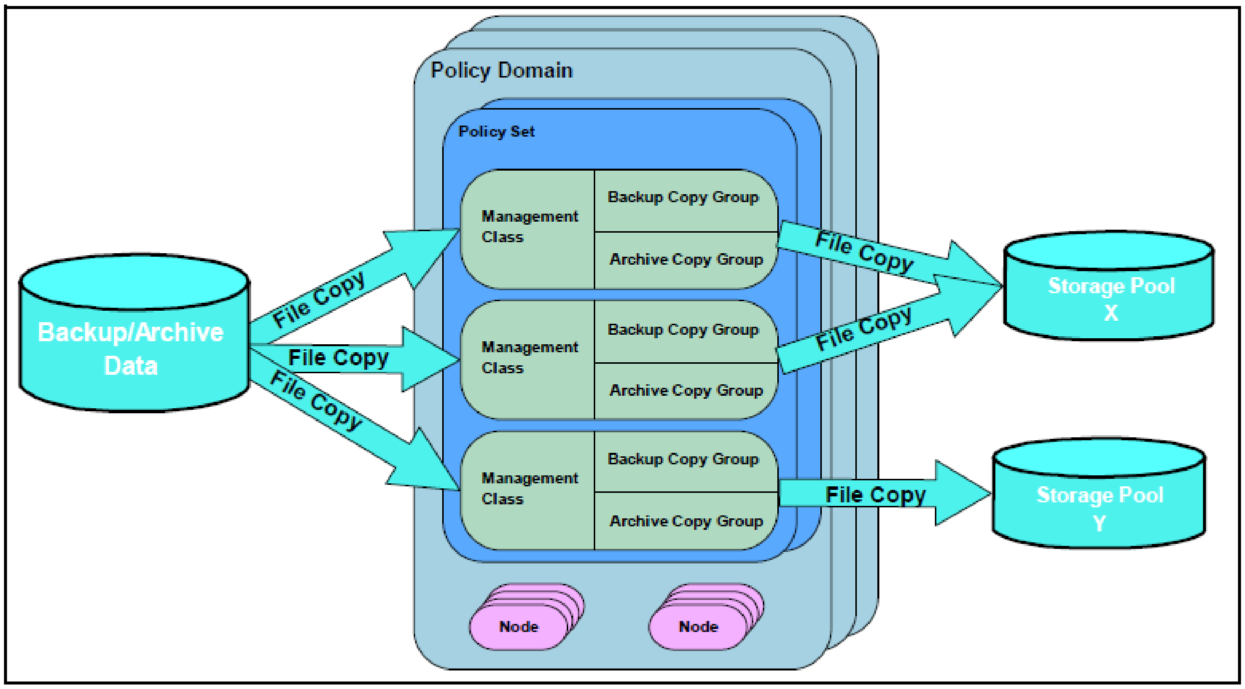


Data storage policy components replacements of the Machines, Data and the rules for storing the data are:



3.1.2 Copy groups

While these two copy groups (Backup and Archive copy groups) serve different purposes, they also share some common ground. They both specify where to store the data sent to them from backup or archive operations, using the COPY GROUP DESTINATION parameter, which specifies a valid primary storage pool to hold the backup or archived data:



Modified files

The **COPYSERIALIZATION** parameter in the copy group provides four possible values to define behavior for modified files:

- The ***shrstatic*** setting specifies that an object will not be stored in Tivoli Storage Manager if it is modified during backup or archive, but multiple attempts will be made to back up or archive the object before the Tivoli Storage Manager client will give it up. If the object continues to be modified through each of these attempts, the object will not be stored at all. The number of additional attempts can be controlled using the CHANGINGRETRIES option in the client options file.

- The ***static*** setting specifies that an object will not be backed up/archived if it is modified during the operation and no additional attempts will be made.

- The ***shrdynamic*** setting specifies that an object will be stored in Tivoli Storage Manager eventually if it is modified during backup or archive but multiple attempts will be made to back it up or archive without modification first. The number of attempt is controlled using the CHANGINGRETRIES option in the client options file. If an unmodified backup/archive cannot be achieved after the number of retries, then the object will be stored anyway.

- The ***dynamic*** setting specifies that an object will be stored even if it is modified during backup or archive. There is no preliminary attempt to back up or archive the object unmodified; it is stored on the first attempt as is.

Note that the **COPYSERIALIZATION** parameter works differently when doing an image backup.

Backup copy groups

The backup copy group is concerned with two logical entities: the ***object*** and the ***object copy***. An object is the actual data on a client node, for example, a file or a directory, while an object copy is a point-in-time copy of the object stored in the Tivoli Storage Manager server.

➢ Backup versioning and retention

An object can be, from a client node's perspective, in one of two possible states: ***existing*** or ***deleted***. When we talk about an existing object on a node, we mean an object that has been previously backed up and still exists on the node. A deleted object is an object that has been previously backed up and subsequently deleted from the node. This simple concept is important when discussing data storage rules.

An object copy can be in one of three states: ***active***, ***inactive***, or ***expired***. An active object copy is the most current server copy of the object, an inactive object copy is a previous copy or version of the object, and an expired object copy is a copy to be removed from the Tivoli Storage Manager server. A backup object copy is set to the expired state when it no longer conforms to the rules specified in the backup copy group.

Whether the object exists or is deleted, the object copy will always pass through the same states in the same order. An object copy will start out as active because it will be the first copy of the object and therefore the most current. Once the objects changes on the client and we make another object copy (by running another backup), the first object copy will change to inactive because we have a more-recent one. Eventually, the first object copy will be expired based on one of two limits placed on it by our copy group rules: number of copies or retention period.

The number of copies that we set in our rules specifies the maximum number of object copies (including the active object copy) that are maintained in the Tivoli Storage Manager server. Suppose we set the number of object copies to three. In this case, Tivoli Storage Manager will keep a maximum of one active copy and two inactive copies.

Once we have these three copies, if we back up the object a fourth time, this will exceed our maximum number of copies or versions to retain. The records about the oldest object copy in the Tivoli Storage Manager database are marked for deletion and subsequently this object copy is expired from the database. If we change an existing copy group value for maximum number of object copies to be retained, let's say we decrease it from three to two versions, the oldest (third) inactive object copy is not automatically marked for deletion. The deletion will occur the next time the client performs a backup, because Tivoli Storage Manager checks the rule at backup time.

Note: There is an exception as to versioning of NDMP objects — versioning applies to complete NDMP dumps only because the Tivoli Storage Manager server is not aware of the single objects included within the NDMP dump.

The retention periods that we set in our rules will govern the length of time that we will retain inactive object copies. It is important to note that there is no retention period for active object copies; they exist as long as the original object still exists on the node.

Whether the object exists on the node will affect which rules are used to expire the object copies. If the object exists, the following two backup copy group parameters are in effect:

- ***VEREXISTS***: Specifies the number of object copies, or versions, to keep. This number includes active and inactive object copies.
- ***RETEXTRA***: Specifies how many days to keep inactive object copies. When an object status changes from active to inactive, it will be kept for retextra days and then removed. It is

important to note that the retention period starts from when the object copy becomes inactive, not from its original backup date.

If an object has been deleted from the client's file system, then during a subsequent backup operation, the Tivoli Storage Manager client will mark the active object copy as inactive in the Tivoli Storage Manager database. At this point, there are only inactive objects copies for this data in the Tivoli Storage Manager server, and the following parameters govern their retention:

- ***VERDELETED***: Specifies how many object copies are to be kept in Tivoli Storage Manager when an object has been deleted on the client.
- ***RETONLY***: Specifies how many days to keep the last object copy in Tivoli Storage Manager when the object has been deleted on the client before.

➢ Backup mode and frequency

The backup copy group defines two other attributes that control the way that backup data is handled: ***MODE*** and ***FREQUENCY***. The ***MODE*** parameter specifies which objects will be eligible for incremental backup. Setting the mode to *modified* will allow an object to be backed up only if it has changed since the last backup. The *absolute* setting means the designated objects will be backed up regardless of whether they have changed or not since the last backup. The latter value therefore turns an incremental backup into a full backup of selected objects, and so would usually be used only in special cases. The default MODE value is *modified*.

The ***FREQUENCY*** parameter specifies how many days must elapse before an object will be eligible for a backup operation, regardless of whether the object has changed. Frequency is honored only during a full incremental backup operation; it is ignored during *partial incremental* or *selective* backups, as the selective operation backs up data regardless of whether it has changed or not. The default frequency value of 0 specifies that objects will be eligible for backup operation any time they change.

For an object to be backed up during an incremental operation from a client node, it has to satisfy three conditions:

- Domain and include-exclude statements allow the object to be considered for backup.
- The object satisfies the mode setting. That is, if the mode is set to modified, the object must have changed to qualify for backup. If the mode is set to absolute, then the object is automatically allowed to be backed up.

- Difference between the server time and the active object copy timestamp must be greater than frequency setting. The frequency is converted to hours to compare to the timestamp difference.

For example, consider a file called /home/admin/redbook.doc that is eligible for backup in the include-exclude list and that has changed since the last backup at 8 a.m. this morning. The server time is 11 a.m. when an incremental backup is started, so the difference between server time and the file copy time is three hours. If the frequency is set to one day, then 24 hours must pass between incremental backups before an object is backed up again. Therefore, the file called /home/admin/redbook.doc will not be backed up, since three hours is less than 24 hours.

➢ Table of contents destination

When performing backup of a NAS node via NDMP, the data being backed up is stored as a single object in Tivoli Storage Manager, either as a full or differential image. When a user wants to examine the directory tree to select files or directories to restore using the Tivoli Storage Manager web client, the Table Of Contents (TOC) must have been saved along with the file system backup. The purpose of the *TOCDESTINATION* attribute is to specify destination primary storage pool for such TOC objects. Note that TOC creation requires additional processing overhead during backups.

Archive copy group

The archive copy group works with entire archives as single unique entities, so it has fewer rules. There is only ever one copy of a particular archive, so we do not have to worry about rules to manage versioning. We still have to specify the retention period for the archive object and that is done with the ***RETVER*** setting. It specifies the number of days to retain the archive copy from the day of the archive operation.

There are three other parameters that the archive copy group uses to handle archive data: ***MODE***, ***FREQUENCY***, and ***COPYSERIALIZATION***. The mode parameter has been discussed, but the archive copy group only allows the value to be set to ***absolute***. This makes sense, since the archive copy group does not link previous archives to the current archive and therefore cannot determine whether anything has been modified. The archive copy group has a frequency parameter, but it can only be set to the value

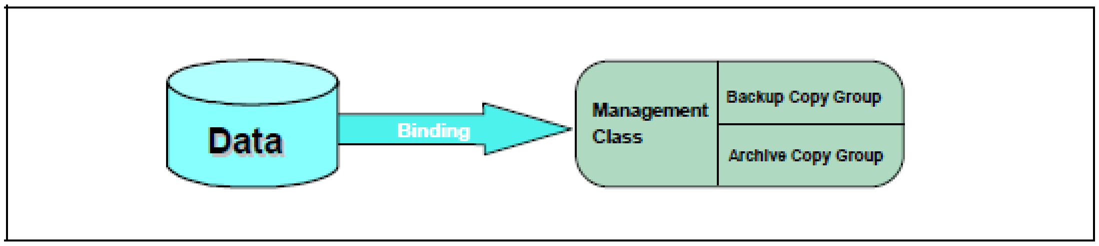
CMD, which means that the archive can be performed on demand. Copy serialization operates in the same way as for backup.

3.1.3 Management class

The management class is a tier in the policy management that essentially serves as an interface between the client’s data and the copy groups whose rules govern the versioning and/or the retention of data. A management class usually contains both a backup and an archive copy group, or it can house either group. A management class can even be empty, that is, without a backup or archive copy group, but in such a case, the management class is useless, as there are no rules that would govern the data.

➢ *Binding and explicit binding*

Figure shows the basic structure of a management class with both copy groups defined. It also illustrates how a management class links the backup/archive data to the rules defined in a copy group. The link is very granular and can be assigned to a single object such as a file, or groups of objects such as a complete file system. When an object is linked to a management class, it is said to be *bound* to the management class.



There is a special instance of a management class called the *default* management class. Each policy domain (logical grouping of client nodes) has a default management class, which contains the rules that will be used for data which is not explicitly bound to another management class. Therefore, there are two ways to bind data to a management class: default and explicit. Unless an object is explicitly bound, the default management class is used. Binding your backup or archive data to different management classes enables you to manage different types of objects with different sets of rules.

Explicit binding to a management class can be divided into four categories, depending on the type of an object: binding file backups, binding directory backups, binding file archives, and binding directory archives.

➢ *Binding backups*

When a client backs up a file for the first time, it binds the file either to the default management class or, in case of explicit binding, to the management class that is specified in the *include* statement in the client options file or in the central client option set on the server. During the life of a file, all subsequent versions that are backed up are bound to the same management class, unless a user changes the binding in the include statement. This is known as *rebinding*.

When the user changes the binding, then both the active and inactive objects are rebound to the new management class during the next full incremental run. In rare cases, if a user changes the binding in an include statement, then instead of running a full incremental backup, only selective or incremental-by-date backups are performed, then only active versions will be rebound, while inactive versions will remain bound to the original management class.

Example shows an include statement that assigns the file /home/admin/redbook.script to a management class called redbook while allowing the rest of the files in /home/admin to go to the default management class. The binding is actually done during the backup operation.

```
include /home/admin/.../*
include /home/admin/redbook.script     redbook
```

Image backups are also bound using the include statement. However, the *include.image* directive is used.

```
include.image /.../* imageMC
```

Backups of directory objects are by default bound to the management class with the largest retention value for *RETONLY* parameter. If there are several management classes with the same RETONLY value, than the directories are bound to the one which is the last in the alphabetical order. The idea behind this is to assure that a directory does not expire earlier than the objects contained within its directory tree. This default binding may result in additional mount points during backups, especially when backing up Windows or Novell client filespaces, even though a user would expect all data to be stored in the primary disk storage pool. To address this, you can explicitly bind directory backups to a management class using the *DIRMC* client option.

In Example, we bind all client directory objects to the directoryMC management class. It is recommended to use a dedicated primary disk storage pool as the destination for this management class, which houses directory objects only. The retention period for this class should be set to at least as long as the longest retention period for the other management classes in that policy set. This ensures directory entries will not expire before the file objects they contain.

```
dirmc directoryMC
```

- ***Binding archives***

Archived file objects are bound to the default management class, unless they are explicitly bound to another management class either using the *ARCHMC* command line option as illustrated in Example 1, or using the include.archive. statement in the client options file, as shown in Example 2. Both ways are equivalent. The file /home/admin/redbook.doc will be bound to the management class redbookarchive.

Example 1:

```
dsmc archive -archmc=redbookarchive /home/admin/redbook.doc
```

Example 2:

```
dsmc archive -archmc=redbookarchive_new /home/admin/redbook.doc
```

Once an object is archived, the management class to which it is bound cannot be easily changed. This makes sense — so that you cannot, by mistake, change the retention period of an archive. If a user changes the binding using either the *archmc* or *include.archive* methods, only subsequent archived objects will be bound to the new management class. Previously archived objects will remain bound to the original management class. In cases where rebinding is desired for previous archives, the user must retrieve the objects back to the client file system and then archive them again.

Slightly different rules apply to binding archived directory objects. By default, the directory objects will be bound to the default management class. If the default management class does not have an archive copy group; then the directory object are bound to the management class whose archive copy group has the shortest retention value.

To explicitly define binding for directory objects, use the archmc command line option or select the *Override include/exclude list* option in the GUI client

Note: Include.archive or **dirmc** statements have no effect on archive directory bindings.

- ***Controlling space managed files***

In addition to copy groups, the management class contains several options that control the migration settings for Space Managed clients (HSM) and their files. The *SPACEMGTECHNIQUE* parameter specifies whether a file using the management class is eligible for both automatic and selective migration, only selective migration, or is not eligible for migration, which is the default. Another attribute, *AUTOMIGNONUSE*, specifies the number of days that must elapse since the file was last accessed before it becomes a candidate for automatic migration.

Notice that these parameters are only for IBM Tivoli Storage Manager for Space Management, not for the HSM Client for Windows.

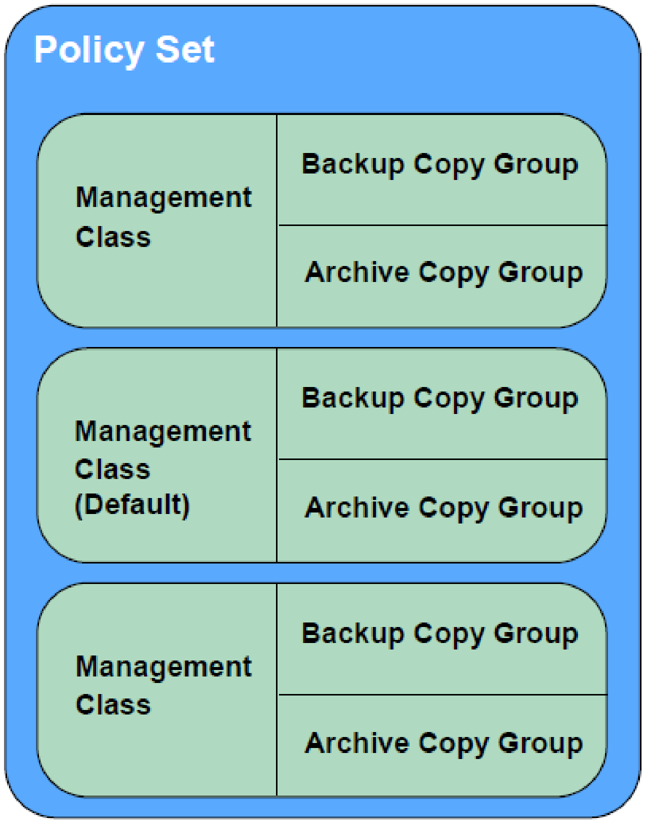
The initial destination for migrated files is controlled by *migdestination* option and this must be a valid primary storage pool. Please note that this option controls destination only for HSM clients, not for backup/archive or API clients. Before a file can be migrated, there must exist a backup of the file in Tivoli Storage Manager, but you can optionally disable the prerequisite by setting *migrequiresbkup* value to *no*.

Please be aware that these options do not apply to the HSM Client for Windows. For details on migration settings for this product, see *Using the Tivoli Storage Manager HSM Client for Windows*, REDP-4126.

3.1.4 Policy set

The next tier in the policy management structure is the policy set. There can be multiple policy sets within a policy domain, but only one is set is active at a time in a domain. The active policy set contains one or more management classes and their associated copy groups. Management classes in any non-active policy sets are not available for binding to clients' objects — only management classes from the active policy set can be used. In practice, multiple policy sets are rarely used, because of the possible confusion of available and non-available management classes.

A policy set can contain many management classes — and of these, one is set as the default management class. The basic structure of a policy set is shown below and indicates that the policy set is used primarily for flexibility. It allows us to group management classes and assign one of them as a default for the policy domain which in turns set the default rules for client objects.



The active policy set is a special entity in the policy domain, and the rules cannot be changed directly. To change it, you must define or update your rules in a policy set that is subsequently validated and activated. The activation process takes a snapshot of the policy set and places it in the active policy set. It is important to note that the active policy set is a point-in-time snapshot of the originating one. Putting it another away — the inactive policy set is like the “shadow” of the active policy set. Changes to the inactive policy set have no effect on the active policy set until the policy set is validated and activated.

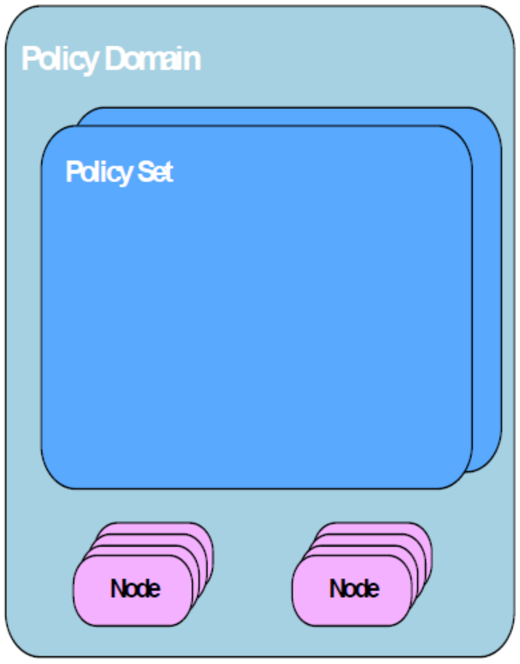
The validation process checks that your policy set is complete and valid. It checks the management classes and copy groups and ensures that the policy set has a default management class. It also ensures that the copy groups point to valid primary storage pools.

The activation process verifies that the default management class and copy group definitions are correct (using the same checks as the validation process) before activating the policy set. Activating the policy set copies its structures to the active one.

When making changes in a policy set, such as adding a new management class or changing the rules in an existing management class, it is a good habit to copy the contents of an active policy set into a temporary one. You then edit the management classes and copy groups in the temporary policy set and then validate and activate it. This ensures that you always make changes to the appropriate policy set.

3.1.5 Policy domain

A policy domain is a way to group Tivoli Storage Manager clients depending on how you want to treat their data. It also contains policy sets that have all of the rules that you want to apply to your data. The figure shows a simplified view of a policy domain with multiple policy sets and nodes contained within it. As discussed, there can be many policy sets, but only one is active.



A policy domain enables you to logically group the machines in your organization according to:

- **Policy rules:** The set of rules to apply to the clients. The rules define the storage management policy including how many copies of data to keep and how long to keep them. The default management class of the active policy set contains the default rules applied to the clients within the domain.

- **Administrative control:** Delegating administrators to control the domain, for example, register clients to the domain, locking/unlocking nodes or changing passwords. Access to the policy rules can be restricted to certain administrators by granting or revoking the administrators access to the policy domain.

Let us consider a typical organization consisting of several UNIX and Windows servers and workstations. The UNIX machines are large database servers that need many copies of their data maintained for a long period of time. The UNIX support group is the only group authorized to access the UNIX machines. The UNIX policy domain would hold all of the UNIX machines, and would only be accessible to Tivoli Storage Manager administrators from the UNIX support group. The active policy set rules in this domain would apply only to the UNIX machines.

The Windows machines are application servers and workstations that need a few copies of their data maintained for a short period of time. The Windows support group is the only group authorized to access the Windows machines. The Windows policy domain would hold all of the Windows machines, and access to it would be restricted to Tivoli Storage Manager administrators in the Windows support group. The active policy set rules in this domain would only apply to the Windows machines.

This is a good example of how default policy and administrative control can be used to break up your organization into policy domains.

Each policy domain includes two rules for governing expiration of stored client objects in cases where all management classes and their associated copy groups have been deleted from the active policy set. When this occurs, there are no longer any rules to govern the retention of data objects within the domain in Tivoli Storage Manager. Rather than expiring all objects immediately in this case, the Tivoli Storage Manager server provides two settings: *BACKRETENTION* and *ARCHRETENTION* that specify, in days, how long to retain backed up and archived objects. The default values are 30 and 365 days, respectively.

3.2 Scheduling

IBM Tivoli Storage Manager includes a central scheduling component that allows the automatic initiation of administrative and client operations at pre-defined times. An administrator is responsible for creating and maintaining the schedules in each policy domain.

Tivoli Storage Manager scheduling is divided into two categories: administrative scheduling and client scheduling. The two categories differ in three key areas:

- **Execution location:** An administrative schedule performs an action on the Tivoli Storage Manager server while the client schedule can only execute on a Tivoli Storage Manager client.
- **Domain privilege:** Only an administrator with system privilege can manage an administrative schedule, while an administrator with policy privileges in the client's domain can manage the client schedule. This granularity can be very useful when distributing management control across a large enterprise.
- **Commands:** An administrative schedule can only initiate an internal Tivoli Storage Manager command, whereas a client schedule can initiate an internal client action such as an incremental backup, or run an external command such as a shell script or executable.

For both types of schedules, there are four key pieces of information:

- A command or action to be executed
- When the command or action executes
- The period, or window, in which the command or action should start
- How often the command or action should be repeated

The command or action that you run may be an incremental backup (client schedule) or a storage pool migration (administrative schedule) that you should run every day at a particular time. You also have to estimate how long the command will run so that you can synchronize your schedules and balance the load on the server.

For client schedules, you can specify whether the client should poll the server regularly to receive information about scheduled actions (client polling), or whether it should wait to be contacted by the server to start a scheduled action. This option is called the scheduling mode on the client and is set in the client options file.

There is another type of client schedule called *Clientaction*, which is for actions which you want to run only once as opposed to recurring scheduled actions. The Tivoli Storage Manager server clock determines all schedule start times, regardless of the time zones where the clients are located.

Introduced with Tivoli Storage Manager V5.3 are different *styles* of schedules. Both client and administrative schedules can now be either *Classic* or *Enhanced*. The styles refer to the way in which schedules are started and repeated. *Classic* refers to the original style of setting the start time and repetition of the schedule using a limited number of options. *Enhanced* refers to the new style, which provides more granular options for setting the repetition. You can now configure a schedule to happen, for example, on the last day of each month.

3.2.1 Administrative schedules

An administrative schedule is a directive to trigger an action within the Tivoli Storage Manager server. It consists of a server command and extra parameters describing when the action should happen. As each administrative schedule can only run one server command, the command itself may be a **run** command, which runs an internally defined server script containing other internal server commands.

Scripting administrative commands can greatly assist timing and sequencing events; the scripting engine has directives to serialize commands and wait for them to complete before continuation of processing.

You should define any actions that you perform on a regular basis to manage the Tivoli Storage Management environment as administrative schedules. Automating these operations to occur in a quiet period, such as overnight, enables the administrator to ensure that server resources are available when clients need them.

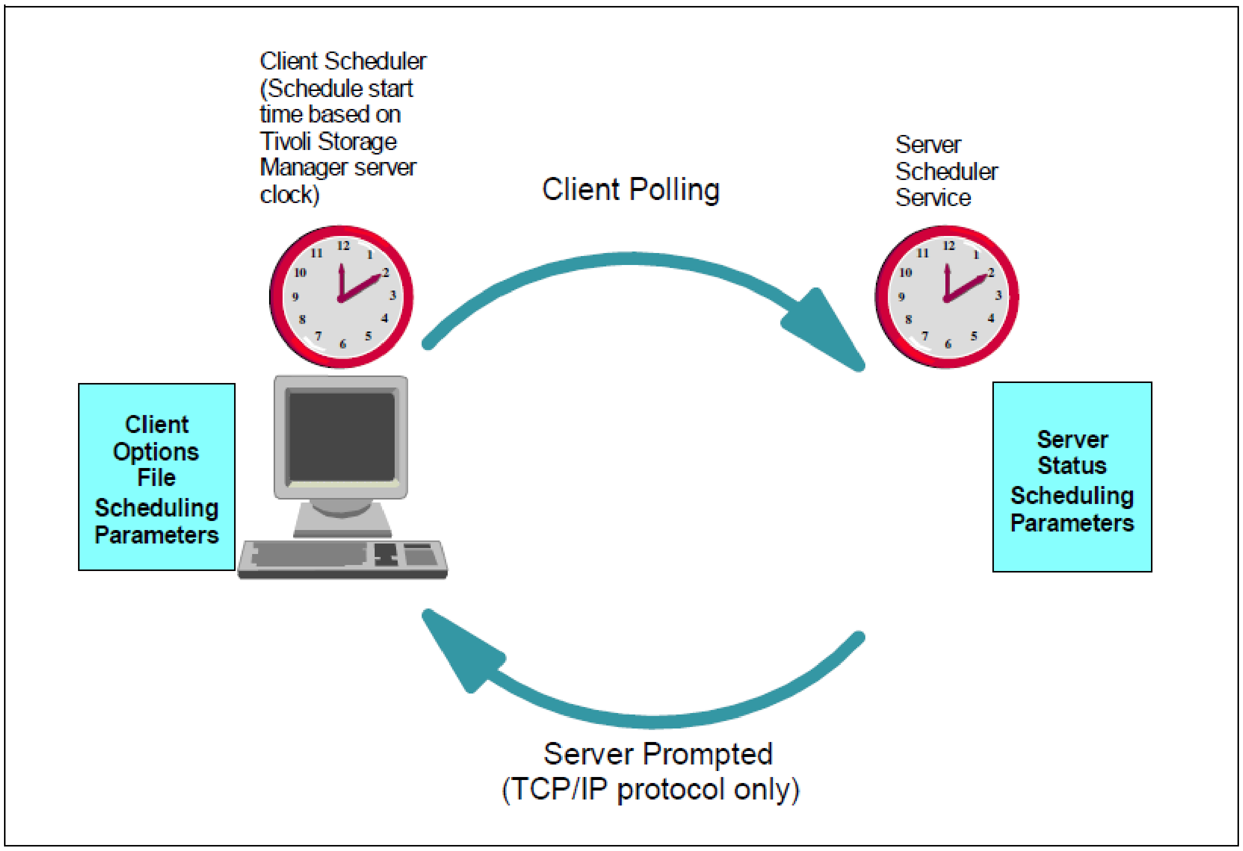
3.2.2 Client schedules

A client schedule is a directive to trigger an action on one or more Tivoli Storage Manager client nodes. It is different from an administrative schedule in that it specifies an action to be performed on the Tivoli Storage Manager client. The client scheduling system consists of a server portion and a client portion. The server part is integrated into the Tivoli Storage Manager process. The server part is responsible for maintaining the schedule parameters and tracking which nodes are associated with each schedule. The client scheduler is a separate process on the Tivoli Storage Manager client that communicates schedule information between the server and client. A client must be running its scheduler process to execute scheduled operations. Otherwise, the operation will be missed and will be logged as such in the Tivoli Storage Manager Server activity log.

The server event and activity logs record the success or failure of each scheduled operation. The administrator can query the logs to find out the status of the schedules. The client also keeps a local log of scheduled operations.
The definition of the client schedule includes the actions to be performed. The action can be a single native Tivoli Storage Manager Client command, such as a backup, restore, archive, or retrieve command. The action can also be to execute a *macro*, which is a collection of native commands, or an external command script. An *external command script* is simply any script that you can execute within the client operating system, such as a Windows batch command file, a UNIX shell script, or perl script. The command script itself can contain Tivoli Storage Manager client commands plus logging, setup, error detection, post-backup procedures and so on. Or you can use the script to schedule functions totally unrelated to Tivoli Storage Manager functions if required. The scheduler is therefore a very flexible general purpose scheduler.

You define a schedule within the policy domain, so that only the client nodes belonging to that domain are eligible to execute that schedule. After defining the schedule, the administrator then specifies which client nodes (from those in the domain) execute the schedule. This action is called *associating* clients with a schedule. The administrator can choose to associate all of the nodes in the domain, or just a subset, according to requirements. The associations can be changed at any time.

There are two scheduling modes available for establishing communication between the client and server to start a scheduled event, as shown. The selection of which mode to use is set in the client options file and is also dependent on the basic communication protocol used between the client and server. If the communication method is anything other than TCP/IP, then only the client polling method may be used. If TCP/IP is used, then the client may select either method. Optionally, the server can override the client’s preference if required.

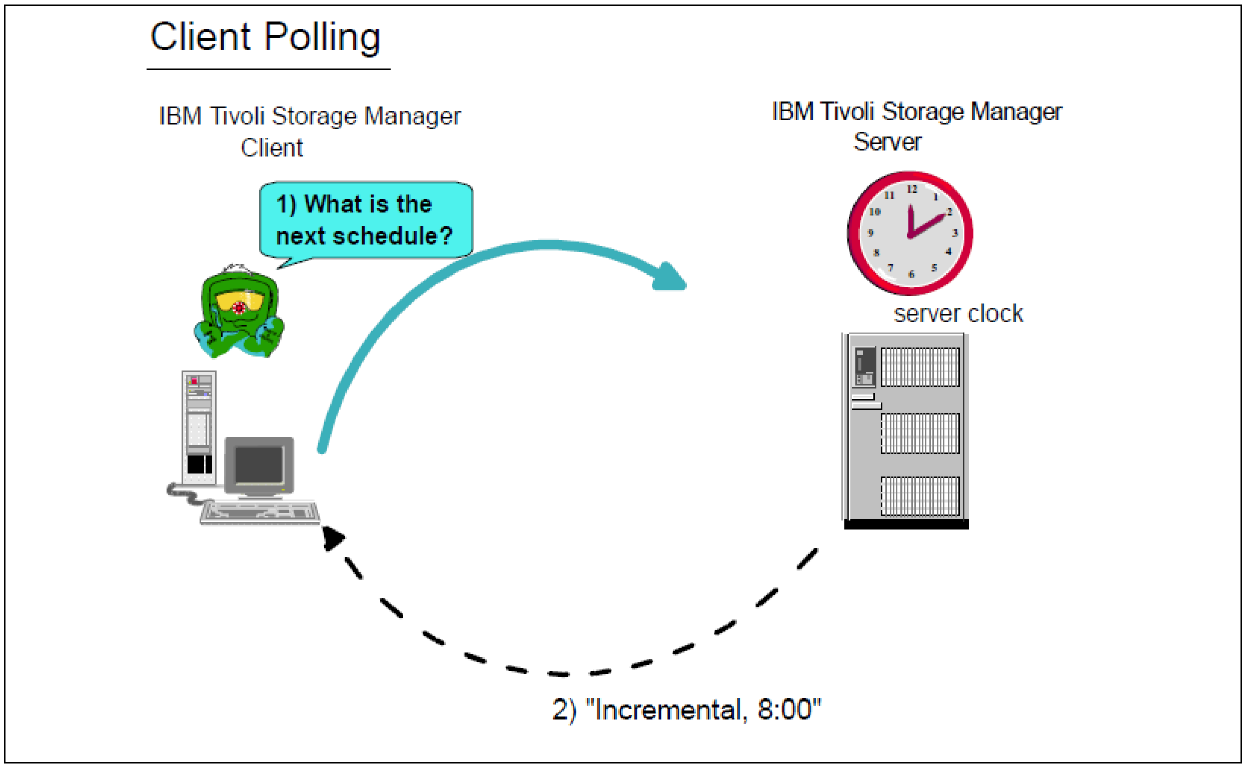

Client Scheduler (Schedule start time based on Tivoli Storage Manager server clock)
Client Polling
Server Scheduler Service
Client Options File Scheduling Parameters
Server Status Scheduling Parameters
Server Prompted (TCP/IP protocol only)

Client polling

If a client has selected the client polling scheduling mode, the client contacts the server to find out if there is a schedule defined for it and when it should be run. The client continues to contact the server at regular intervals set in the client's options file so that it can respond dynamically to any changes made to its schedules by the administrator. The administrator can also override this interval by setting a server parameter.

When the client contacts the server, if there is a schedule to execute, the client receives a start time from the server. The client then counts down to the start time - when the start time arrives, the client performs the actions defined by the schedule. If there is no scheduled action to run between the time of contact and the next contact time, the client simply waits.

For example, a daily schedule may back up all of the client's file systems incrementally. Figure shows a client polling schedule configuration in which the client regularly queries the Tivoli Storage Manager server about the next operation to run. In this case, the Tivoli Storage Manager server informs the client when the schedule must run, but if it is not the right time, the client waits. If the next poll is due to occur before the schedule is due to run, the client polls the server again at the time specified by the interval (QUERYSCHedperiod client parameter). If nothing has changed since the last poll, the server simply responds with the same information

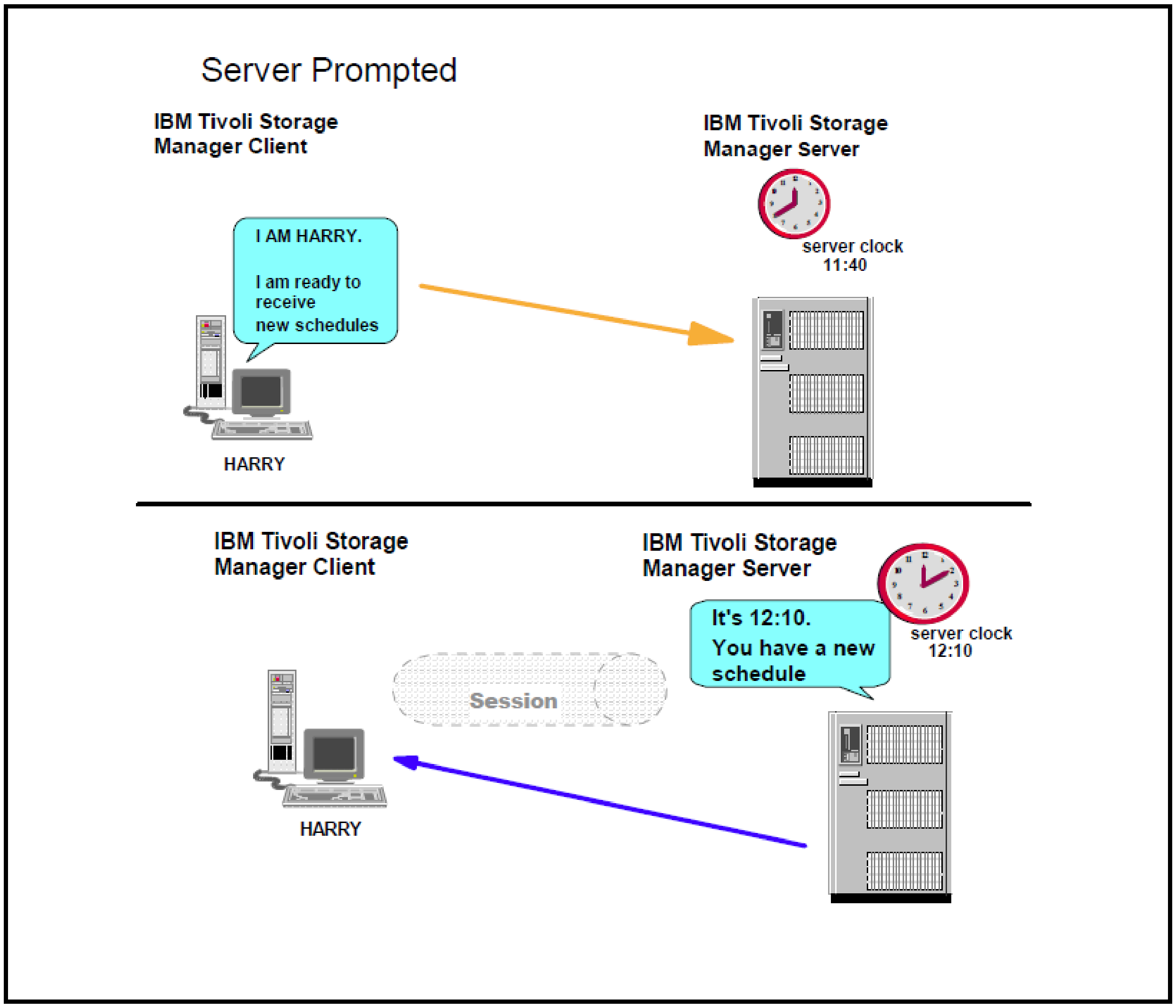

Client Polling
IBM Tivoli Storage Manager Client
1) What is the next schedule?
IBM Tivoli Storage Manager Server
server clock
2) "Incremental, 8:00"

The main advantage of client polling mode is that you can have the server automatically assign random start times for each client's schedule execution within a proportion of the schedule's start window. Randomized start times are useful if many clients have to execute the same scheduled operation and you do not want to overload the server by starting them all simultaneously. The randomization function is not available when the schedule mode is server-prompted.

Server-prompted

If a client has selected the server-prompted mode, the client first contacts the server to notify the server that it is running and the client is able to receive new schedule notifications. The server responds to let the client know if it has schedules or not. The client then sleeps until the server contacts it. If the server detects that a new schedule is ready to run, it contacts the client and informs it that the client must start a new operation. The client receives the schedule definition and starts the operation.

Figure shows an example of the communications between client and server when a scheduled operation must run on the client.


Server Prompted
IBM Tivoli Storage Manager Client
IBM Tivoli Storage Manager Server
I AM HARRY. I am ready to receive new schedules
server clock 11:40
HARRY
IBM Tivoli Storage Manager Client
IBM Tivoli Storage Manager Server
It's 12:10. You have a new schedule
server clock 12:10
Session
HARRY

If you use server-prompted scheduling, you can quickly and easily rerun a failed backup process by restarting it from the server. The other advantage of server-prompted scheduling is that you do not have the continued regular polling traffic from each client that is required by the client polling method (although the polling traffic is minimal).

One-time client schedule

As well as regularly scheduled (repeating) operations, you may also define a schedule for one-time-only processing, on a single client or set of clients. A one-time-only client schedule is known as a *clientaction*.

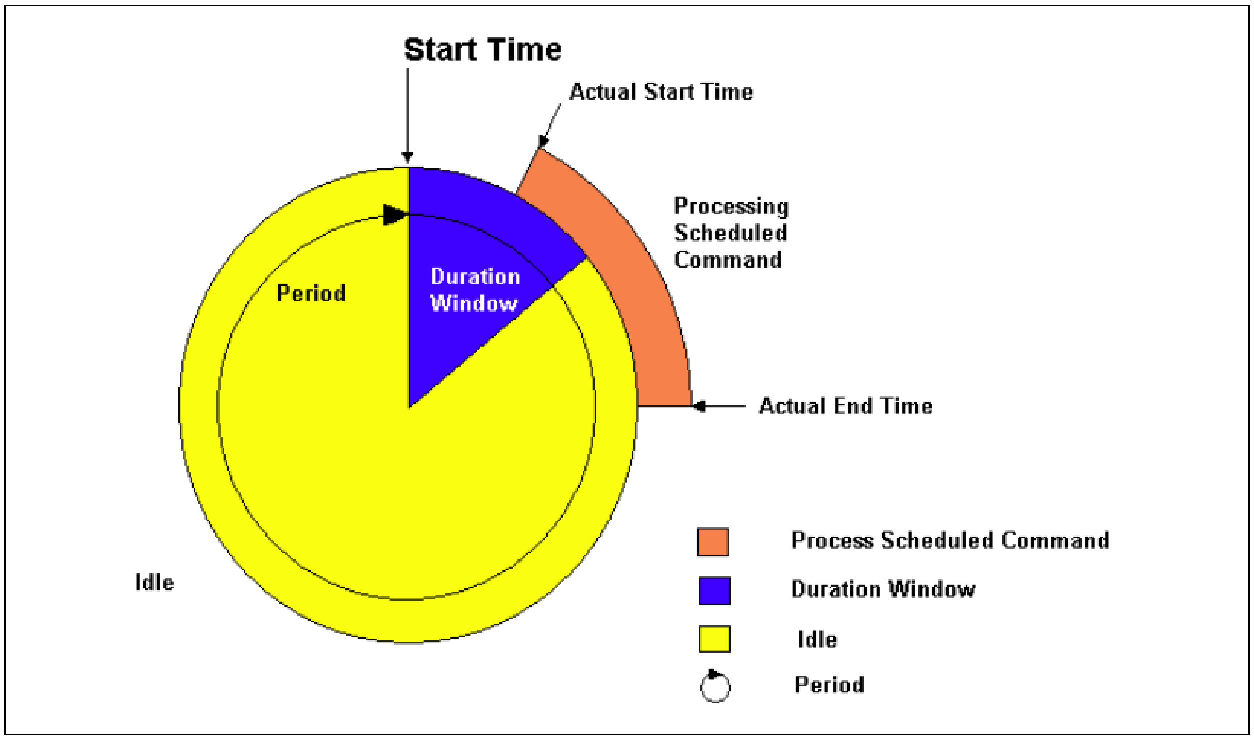
Unlike the regular client schedule, defining a one-time client schedule and associating the nodes with it is done as a single command by the administrator. Once you define the schedule, client nodes associated with the schedule that are in server-prompted mode, receive the schedule and process the command virtually immediately. Clients in client polling mode will receive the schedule the next time they poll the server.

Another administrator-defined parameter, ***CLIENTACTDuration*** defines the length of time the clientaction schedule exists on the server. After this time (in days), the schedule is removed whether associated nodes have run the schedule or not.

3.2.3 Frequency and duration

When defining a schedule, as well as a start time and date, the administrator must define the time period (known as the *duration*) during which a schedule can *start*, and how often to repeat the schedule (*frequency*). The classic and enhanced schedule styles have different methods of specifying the frequency parameters.

An example of a typical classic schedule would be “every night at midnight”. The start time is 00:00; the period between startup windows is 24 hours (1 day). The duration is set to two hours, as shown:



3.2.4 Retry and randomization

A number of server parameters control the maximum number of concurrent client sessions allowed to connect to the Tivoli Storage Manager server, and the percentage of these that are scheduled sessions. If you restrict the number of scheduled sessions allowed on the server, you prevent a client from running a schedule when the maximum number of sessions has been reached. Through options that you can set globally at the server or individually for each client, the client can retry a certain number of times to run the schedule, with a specified time interval between retries.

As mentioned earlier, the server can randomize the start time of a client schedule within the configured start up window duration for clients in client-polling mode. The randomization parameter, controlled with ‘SET RANDOMIZE’, specifies the percentage of the startup window in which random start times are calculated. For example, if RANDOMIZE is set to 50 (percent) and a schedule has a startup window duration of 2 hours, a client in polling mode will be assigned a start time anywhere within the first hour (50 percent of 2 hours).

The retry and randomization options provide considerable flexibility in balancing the network load.

Consider a scenario where an administrator associates 100 workstations with a backup schedule that has a startup window of between 2 a.m. and 6 a.m. every Friday. If the randomization option is set to 50 percent, Tivoli Storage Manager staggers the start times of the 100 backup sessions so that they start at different times between 2 a.m. and 4 a.m. The randomization prevents a large bottleneck from occurring if the 100 schedules all start at 2 a.m. Note that the more clients associated with a schedule, the larger the startup window should be in order to allow the randomization to be effective.

3.2.5 Logging schedule events

Scheduled operations, also referred to as *events*, are stored in the Tivoli Storage Manager database. The results of scheduled events are stored in a "log" (really a database table). You can find other information regarding completed schedules in the server activity log (also a database table).

You can view which schedules ran successfully, were missed, and are scheduled to run in the future. If there are many schedules, and you are only interested in those that failed or were missed, you can view only those schedules that failed or did not run as scheduled.

Detailed reports of the schedule are logged at the client level, with high-level data being sent to the server for logging (such as completion status, number of bytes sent, and so on). High-level results are sent to the event log so you can view whether or not schedules ran successfully.

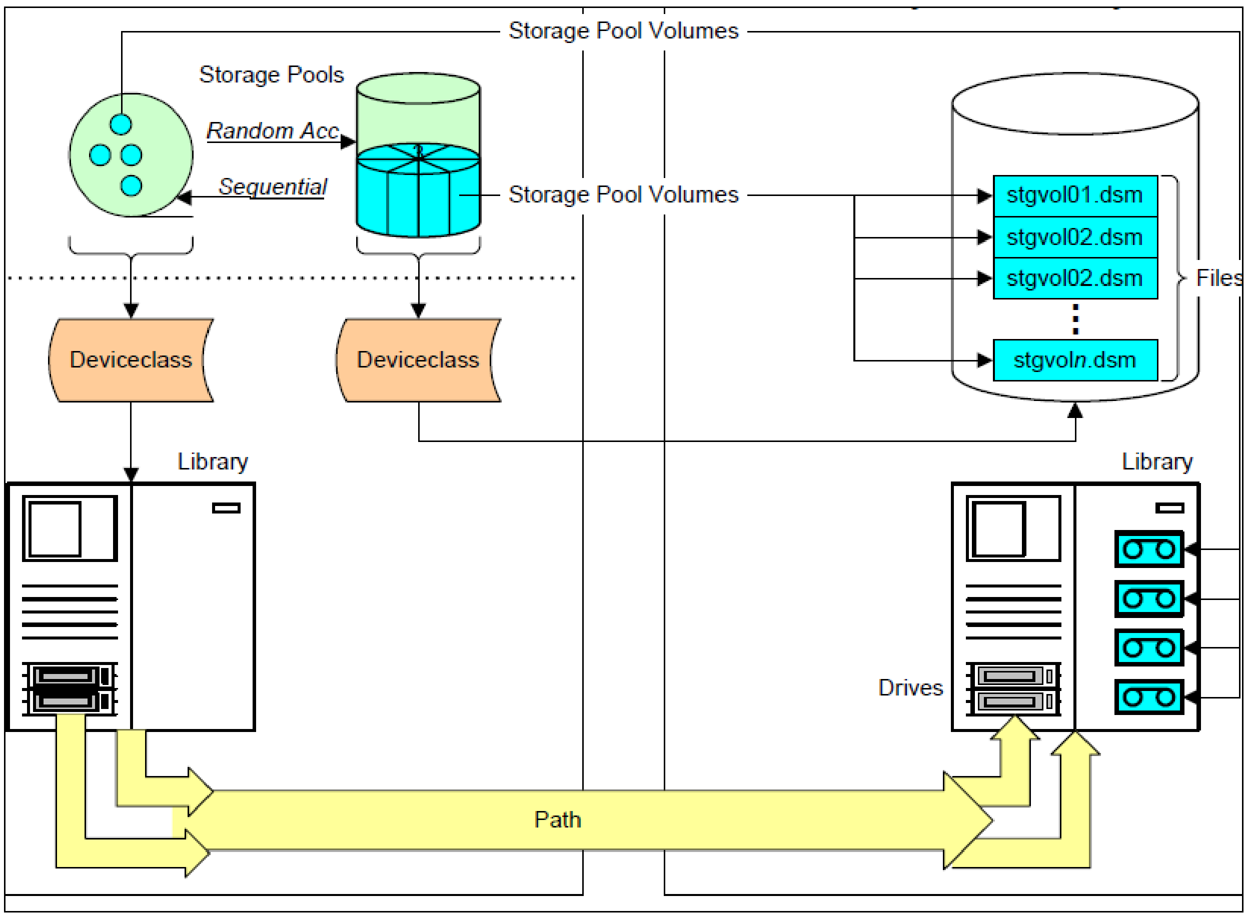
A number of server parameters configure how long event and schedule data is kept in the event and activity logs. The default for event data is 10 days, while the default for the activity log is one day. The activity log can also be managed by size rather than date if so desired.

3.3 Data storage

IBM Tivoli Storage Manager represents data storage as administrator-defined objects: storage pools and storage pool volumes physically stored on data storage devices such as disks, libraries and tapes.

Tivoli Storage Manager devices and media are represented by objects that have been defined by an administrator. Information about the objects is stored in the database. The objects represent the devices and media used by the Tivoli Storage Manager server. You can define, query, update, and delete the objects.

Figure shows an overview of the Tivoli Storage Manager storage objects and their relationships.



3.3.1 Storage pool

A storage pool is a logical entity that represents a collection of physical storage pool volumes; each storage pool represents one type of media. For example, a storage pool for Linear Tape Open (LTO) represents a collection of only LTO tapes, and a storage pool for an IBM 3590 represents a collection of only 3590 tapes. A storage pool created on a disk has files formatted under Tivoli Storage Manager as volumes and are collectively grouped in the storage pool (but not necessarily on the same physical disk).

Volumes can be added or removed to and from the storage pool without interrupting server operations. In this way you can increase or decrease the size of a storage pool dynamically without affecting the Tivoli Storage Manager service.

The two main categories of devices supported for storage pools are: random access and sequential access devices.

- The term *random access* devices refers to devices that can be accessed in a random fashion, that is, data can be read from or written to any part of the media in a series of I/Os. Random access devices are usually magnetic disks.
- The term *sequential access devices* refers to devices where data is accessed sequentially, that is, one block at a time, one after the other. Sequential access devices usually are tape devices and/or optical devices such as MO, CD, or DVD. It is also possible to configure a sequential access storage pool on a disk device (using a FILE device class).

Storage pools in Tivoli Storage Manager can be defined on a wide range of supported devices that may be attached locally to the server (ATA, SCSI, and so on), via a LAN (library sharing) or accessed via a storage area network (SAN).

Tivoli Storage Manager has two types of storage pools: *Primary Storage Pool* and *Copy Storage Pool.*

Primary storage pools

When a client node backs up, archives, or migrates data, the data is stored in a primary storage pool.
When a user tries to restore, retrieve, or export file data, the requested file is obtained from a primary storage pool wherever possible. Primary storage pool volumes are always located on-site, usually within a library. Stored objects are only restored from copy storage pools (see below) when the expected primary storage pool volume is unavailable.
A primary storage pool can use random access storage (DISK device class) or sequential access storage (for example, tape, optical, or FILE device classes).

Copy storage pools

A copy storage pool provides an additional level of protection for client data and is created for the express purpose of backing up a primary storage pool. Copy storage pool volumes are intended for shipment offsite, to provide recoverability of the Tivoli Storage Manager server environment. The copy storage pool contains all current versions of all files, active and exactly as they appear in the primary storage pool.
A copy storage pool provides recovery from partial and complete failures of a primary storage pool. A partial failure would be, for example, if a single tape in a primary storage pool is damaged by a failing

drive, or simply has too many read or write errors. When a client attempts to restore a file that was on this volume, the server will automatically request the appropriate copy storage pool volume. If the volume is still in the library, the server can seamlessly restore the client's data. If the volume is offsite, the server will issue a request for the tape to be returned and mounted. The complete failed volume can be rebuilt onto another volume using the data on the appropriate copy storage pool volumes. If all volumes in a primary storage pool are destroyed (for example, in a major disaster), the copy storage pool is used to recreate the entire primary storage pool.

A copy storage pool can only use sequential access storage devices (for example, tape, optical, or FILE device classes). Copy storage pools can also be created remotely on another Tivoli Storage Manager server, providing electronic vaulting.

Copy pools are not considered part of the storage hierarchy. Files are not migrated to or from copy storage pools as they are between primary storage pools. There are two ways to store files in a copy storage pool:

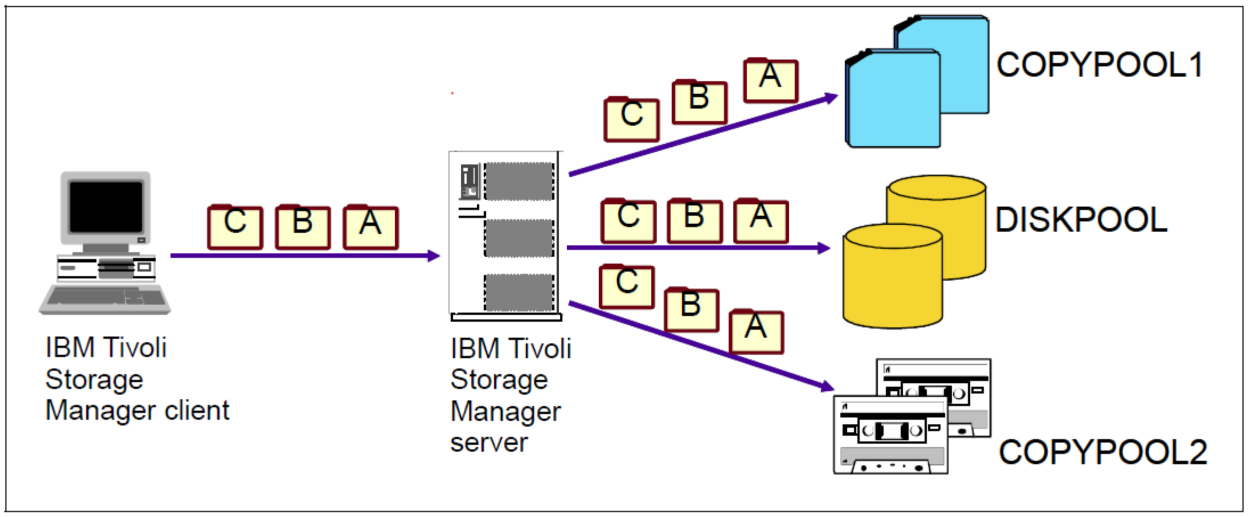
- Copy the primary storage pool to a copy storage pool using the BACKUP STGPOOL command.
- Do a simultaneous write to copy storage pools during client data transfer activity.

Simultaneous writes to copy storage pools

Tivoli Storage Manager can simultaneously write a client's files to each copy storage pool specified for the primary storage pool to which a client's files are written. The simultaneous write to the copy pools takes place during backup or archive from the client (in other words, when the data enters the primary storage pool hierarchy), or during data migration from an HSM client. Up to three copy storage pools can be specified for each primary storage pool. The simultaneous write facility is not a replacement for the BACKUP STGPOOL operation. The BACKUP STGPOOL command cannot write to multiple copy storage pools at the same time.

Figure shows a Tivoli Storage Manager client backing up files to a primary pool — called DISKPOOL, and at the same time, the files are sent to two copy storage pools, COPYPOOL1 and COPYPOOL2.

Simultaneous writes are not supported for server-free or LAN-free backups, or when a NAS backup is writing a TOC file.


IBM Tivoli Storage Manager client
C
B
A
IBM Tivoli Storage Manager server
COPYPOOL1
DISKPOOL
COPYPOOL2

3.3.2 Device class

The device class is the basic building block for storage on a Tivoli Storage Manager server. A device class defines the type of storage hardware used for a particular storage pool. The device class not only includes the storage device type but also links the storage pool to the specific operating system-defined device (in the case of removable media devices).

Each device defined to Tivoli Storage Manager is associated with one device class. That device class specifies a device type and media management information, such as recording format, estimated capacity, and labeling prefixes.

Tivoli Storage Manager provides a set of specified removable media device types, such as 8MM for 8-mm tape devices, or UTLTRIUM3C for LTO3 drives in compressed format. Magnetic disk devices are the only random access devices. All disk devices share the same device type and predefined device class: DISK.

When configured for a sequential media type, a device class is usually associated with a library (containing tape drives) that will physically store the data from the storage pools that use this device class.

Mixed media libraries

Mixed media in a Tivoli Storage Manager server refers to libraries containing different device types (in the device class definition) in the same logical library. Ultrium2 and SDLT are examples of two devices that need different device classes but can co-exist in the same library.

Mixed generation libraries

A mixed generation library in a Tivoli Storage Manager server describes libraries that use the same device types despite capacity differences. In order to have mixed generation devices in the same device class, the media types must be distinguishable.

Note: In mixed generation environments, the current generation device can generally read and write current and previous generation media. The previous generation device can only read and write previous generation media. LTO Ultrium devices are an example of mixed generation devices. For further details about implementing LTO devices and IBM Tivoli Storage Manager, refer to the redbook Implementing IBM Tape in UNIX Systems, SG24-6502, and Implementing IBM Tape in Linux and Windows, SG24-6268.

3.3.3 Library

A library represents a storage entity that contains drives and tapes for storing data. A library always has drives defined to it. **Note** that a defined library only represents a logical object within Tivoli Storage Manager. The direct link between the logical and the physical library will be defined next with *path* objects.

3.3.4 Drive

A drive is part of a library and is used to write data to, and read data from, tape volumes stored in the library. The special hardware-based format used by the tape device is represented in the corresponding device class. Though a device class has no direct connection to a device object, they are linked by the library object that contains the drives. As for libraries, a drive is only a logical object. The direct connection to the physical device will be established next with the *path* object.

3.3.5 Path

Paths configure physical access to drives and libraries. A path definition specifies a logical source, a logical destination and a physical destination. The path connects the logical layer within the Tivoli Storage Manager server with the real-world physical hardware. By using paths, all storage devices on a particular server can be shared between itself and other Tivoli Storage Manager servers, storage agents, and clients. All of them use the same logical object but the individual connection to the associated hardware is established with an individual path definition for each of them.

The source accesses the destination, but data can flow in either direction between the source and destination. Here are two examples:

- In LAN and most SAN operations, control data and client data flow across a path from the Tivoli Storage Manager server to an automated library that is defined to the server. During a restore, client data also flows from the library back to the server.
- In NDMP operations, backup data flows across a path between the source, a data mover defined for an NAS file server, and the destination — a tape drive. Restore data flows back across the path from the tape drive to the NAS file server. Paths can address the same destination device from different sources. For example, in NDMP operations, paths are always required between the data movers that represent NAS file servers and the drives to which the file servers transfer data for backup. Paths can point from multiple NAS file server data movers to the same tape drive.

3.3.6 Data mover

Data movers are devices that accept requests from Tivoli Storage Manager to transfer data on behalf of the server. Data movers transfer data:

- Between storage devices.
- Without using significant Tivoli Storage Manager server or client resources.
- Without using significant network resources.

For NDMP operations, data movers are NAS file servers. The NAS data mover definition contains the network address, authorization, and data formats required for NDMP operations. A data mover enables communication and ensures authority for NDMP operations between Tivoli Storage Manager server and the NAS file server.

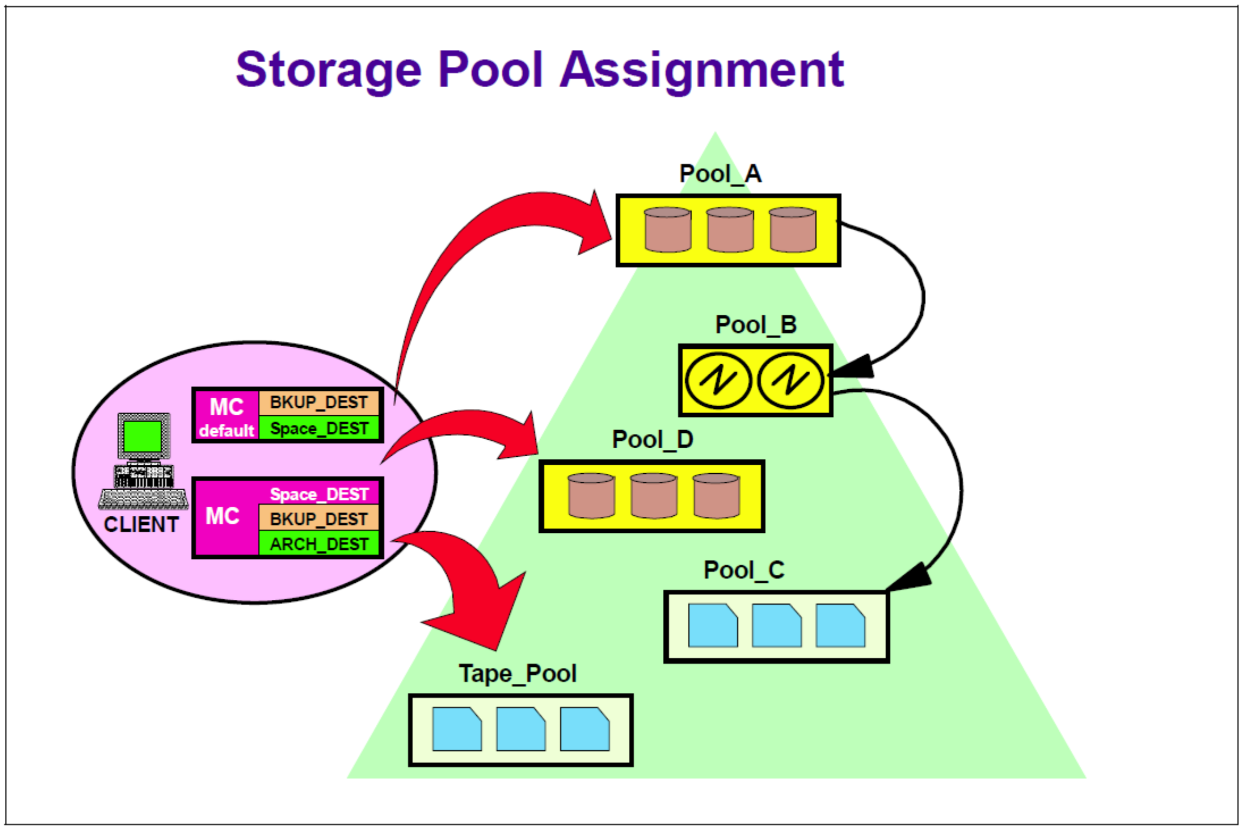
3.4 Storage pool hierarchy

Tivoli Storage Manager enables you to configure primary storage pools to provide the best combination of performance throughput and data retention. In most cases, keeping client data on tape or optical media is a requirement. However, making the backups direct to tape may not give the best performance, especially where there are many clients to back up concurrently, or many small files are being backed up.

Because of the limitations of storage media, Tivoli Storage Manager provides the ability to configure storage pool hierarchies. A storage pool hierarchy allows different types of devices to be in a "chain", with one pool overflowing to the next. A typical hierarchy consists of faster, smaller-capacity devices (disks) overflowing to slower, larger-capacity devices (tapes). A client initially backs up to the disk storage pool. When the amount of data in the disk storage pool reaches a pre-defined high threshold, files are automatically migrated to the next storage pool in the hierarchy — the tape storage pool. The client continues its backup operation without interruption. The migration process continues until the amount of data in the disk storage pool falls to a pre-defined low threshold, at which point, migration stops.

Migration is controlled by the high and low thresholds, plus a number of other parameters, set on the storage pool.

Figure shows a configuration with five defined storage pools. The default management class sends backups first to storage pool A, which are then migrated by the server to pool B and finally to pool C. The other management class uses storage pool D for space managed files, whereas backups are sent directly to Tape_Pool. Sending data directly to a tape device would be appropriate for large client files (for example, application database files) that can take advantage of the streaming performance of a tape device such as LTO3.



Note that the storage pool hierarchy concept is for primary storage pools only. Copy storage pools do not have a hierarchy.

3.5 Backup-archive client

The backup-archive client has two distinct features:

- Backup: This allows users to back up (and manage) a number of versions of their data onto the Tivoli Storage Manager server (or servers) and to restore from these if the original files are lost or damaged. Some examples of loss or damage are hardware failure, theft of a computer system, and virus attack.

- Archive: This is intended for making long-term copies of data, and for later retrieval if necessary. Typical uses of archive are to meet legal requirements, to return to a previous working copy if the software development of a program is unsuccessful, and to archive files that are not needed locally on a workstation. In this last case, the original files can be deleted from the client, thus freeing up space.

Each of these features has a complementary function as well:

- Restore: This function enables users to recover any data that has been previously backed up, and to restore older versions of the lost data.
- Retrieve: This enables users to request the return of previously archived data.

3.5.1 Interfaces

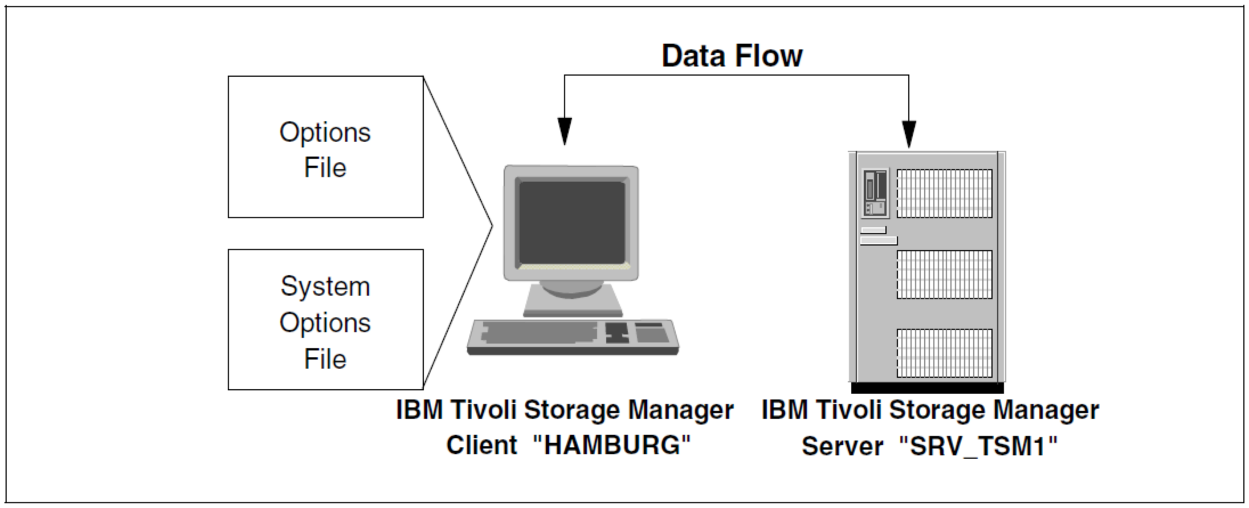
- Graphical user interface (dsm GUI)
- Graphical user interface (dsmj Java GUI)
- Command line interface (CLI)
- Web client interface (Web client)

3.5.2 Configuration and option files

Configuration files and options files are used to specify one or more servers and communication options for backup and restore services. The file can include authorization options, backup and archive processing options, scheduling options, and, where applicable, IBM Tivoli Storage Manager for Space Management options.

On UNIX/Linux platforms, the Tivoli Storage Manager options reside in two options files: the client system options file (dsm.sys) and the client options file (*dsm.opt*). On other platforms, there is only one file — the client options file (*dsm.opt*), which contains all of the parameters. The client systems administrator sets up these files when the Tivoli Storage Manager backup-archive client is first installed on the user's workstation.

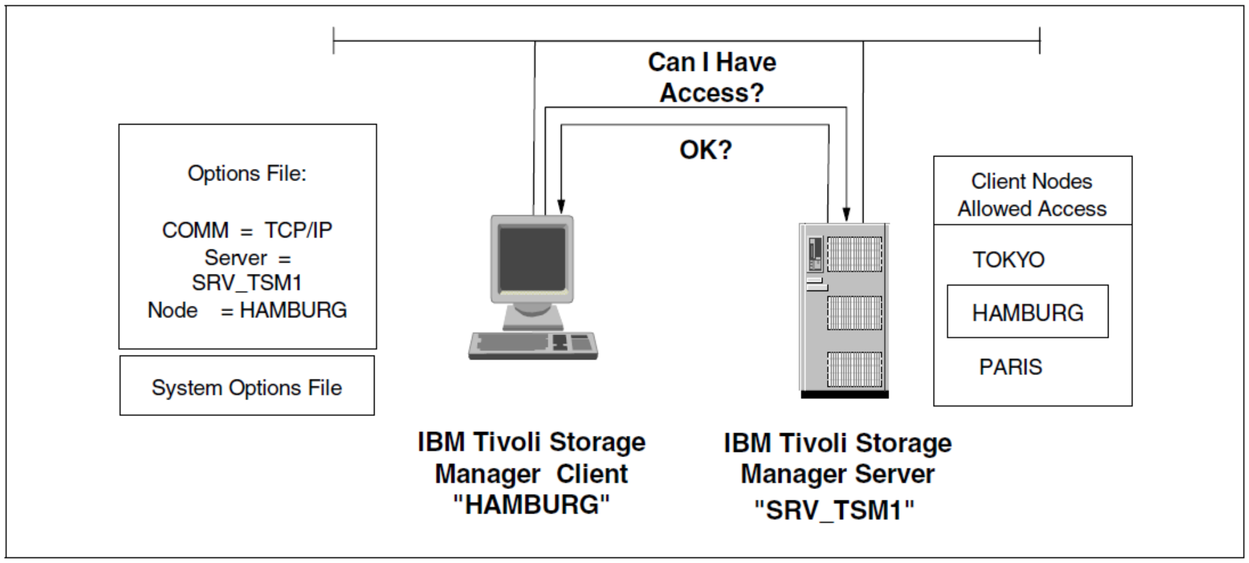
Figure shows a schematic view of the options file for the client.


Data Flow
Options File
System Options File
IBM Tivoli Storage Manager Client "HAMBURG"
IBM Tivoli Storage Manager Server "SRV_TSM1"

The minimum configuration parameters for successful communication are:

- **Communication Protocol:** Client and server using the same type, typically TCP/IP
- **Tivoli Storage Manager server address:** Identifies the correct Tivoli Storage Manager server to use
- **Nodename:** The name by which the Tivoli Storage Manager server knows the client. This information is required so that the server allows the client access. The node name and password are set up on the server, and if different from the machine name, they also must be added in the client options file.

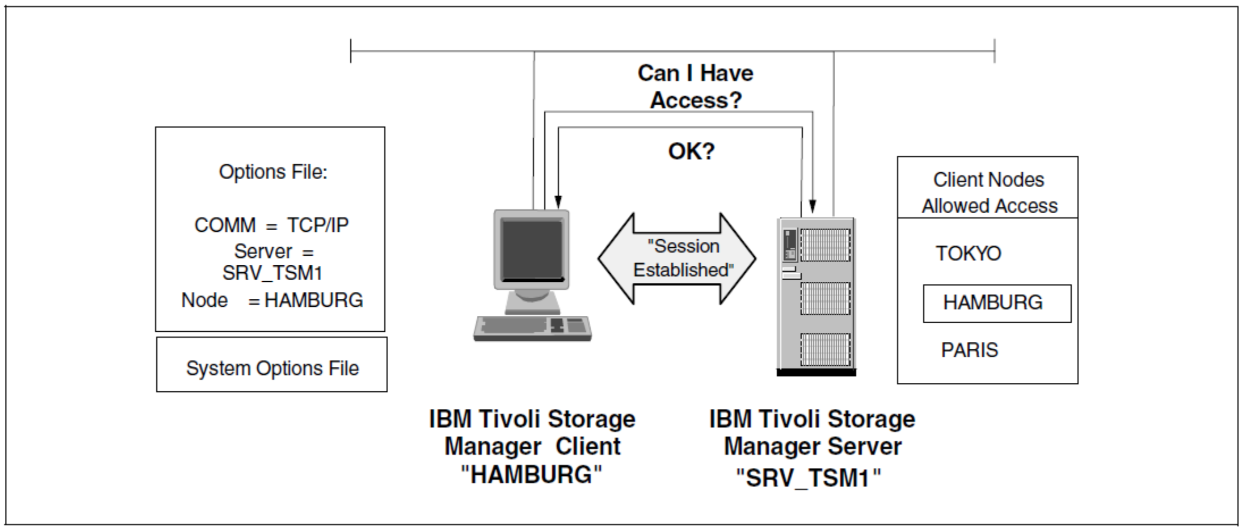
Figure shows an example of a client (HAMBURG) and its minimum configuration settings.


Can I Have Access?
OK?
Options File:
COMM = TCP/IP
Server = SRV_TSM1
Node = HAMBURG
System Options File
Client Nodes Allowed Access
TOKYO
HAMBURG
PARIS
IBM Tivoli Storage Manager Client "HAMBURG"
IBM Tivoli Storage Manager Server "SRV_TSM1"

3.5.3 Establishing a session

The Tivoli Storage Manager backup-archive client node must be registered with the Tivoli Storage Manager server. Once registered, the Tivoli Storage Manager client starts its communication with the server by a sign-on process. This sign-on process requires the use of a password that, when coupled with the node name of the client, insures proper authorization when it connects to the server.

Figure shows an example of the steps to establish a session with the Tivoli Storage Manager server.



3.6 TSM DB (DB2)

The Tivoli Storage Manager database that comes with pre-6.1 servers uses a proprietary B+ Tree database that originated from IBM Almaden Research. This database utilizes state of the art ARIES/NT1 technology for the recovery log.

This proprietary database was chosen for two primary reasons at the time:

- The Tivoli Storage Manager proprietary database was portable across platforms.

The V1R1 of ADSM shipped for the MVS and VM platforms. In subsequent releases, this platform coverage expanded to OS/2, AIX, HP, SUN, Windows, and so on. This expanded platform coverage was possible because the proprietary DB package was imbedded in the product and the architecture was to be platform independent. At the time Tivoli Storage Manager was first being delivered, DB2 was not available on all of the platforms where Tivoli Storage Manager was expected to run, or on platforms that Tivoli Storage Manager was expected to be ported.

- The Tivoli Storage Manager proprietary database was chosen for performance.

The Tivoli Storage Manager proprietary database does not have locking or other common database serialization constructs built into it. It was optimized more for performance than what typical database products provided at the time.

The database has surpassed every expectation in terms of scalability and performance, supporting up to 530 GB databases. Over time, however, Tivoli Storage Manager development had to develop and maintain code that essentially keeps alternative indices for every table and can audit and correct referential integrity problems between tables. The database does not support secondary indexes, and a lot of code had to be written to implement and maintain alternate index information to speed searches. In addition, Tivoli Storage Manager development has written and maintained its own SQL engine for query processing and to load the data warehouse from the server database.

Here are some significant characteristics of the Tivoli Storage Manager proprietary database:

- Locking is done at an "advisory" level within the application itself.

The database does not implement locking at a record level and such, there are latch semantics at a page level that latch a page in “exclusive” mode when a given record on the page is being inserted, updated, or deleted.

- The database package does not provide or maintain indexes for tables.

The Tivoli Storage Manager application maintains the tables and any apparent index or alternate table view that is currently needed or supported by implementing these as separate tables. For example, if a given object on the server is represented by a record entry in tables A, B, and C, it is the application’s responsibility to insert all three of these records when the object is created, the application must make any necessary updates across these records if an object is updated, and it must delete all three of these records if a given object is deleted.

- Referential integrity checking is done by the application.

For Tivoli Storage Manager, this means that the application code has to manage and enforce any referential dependencies between rows in tables and such. Along these lines,the AUDITDB facility within Tivoli Storage Manager was implemented to evaluate andenforce many of the referential constraints that exist with Tivoli Storage Manager.Because referential integrity checking is done at the application levels, it allows forinconsistencies in how the checking is done. It also provides the opportunity to introduceerrors in the database because of misunderstanding the relationships that exist.

- The upper limit of the database size is at 530 GB.

The database uses internal control structures to manage the available size of the database. Today customers are reaching the upper bound limit with their production databases. Similarly, the disaster recovery products (DR450) and other content manager type offerings have brought forward concerns over the maximum number of objects that could be stored in the server database.

As Tivoli Storage Manager has evolved over the years, enhancements to the database have generally been small. There has been investment in the database such as the following enhancements: DUMP/LOAD/AUDITDB, BACKUP/RESTORE, and SQL. However, the investment in the database has been primarily on an as-needed basis.

Goals of the transition to DB2

Through the years, while discussing with customers the transition to DB2 for the Tivoli Storage Manager database, the following major expectations or objectives were chosen:

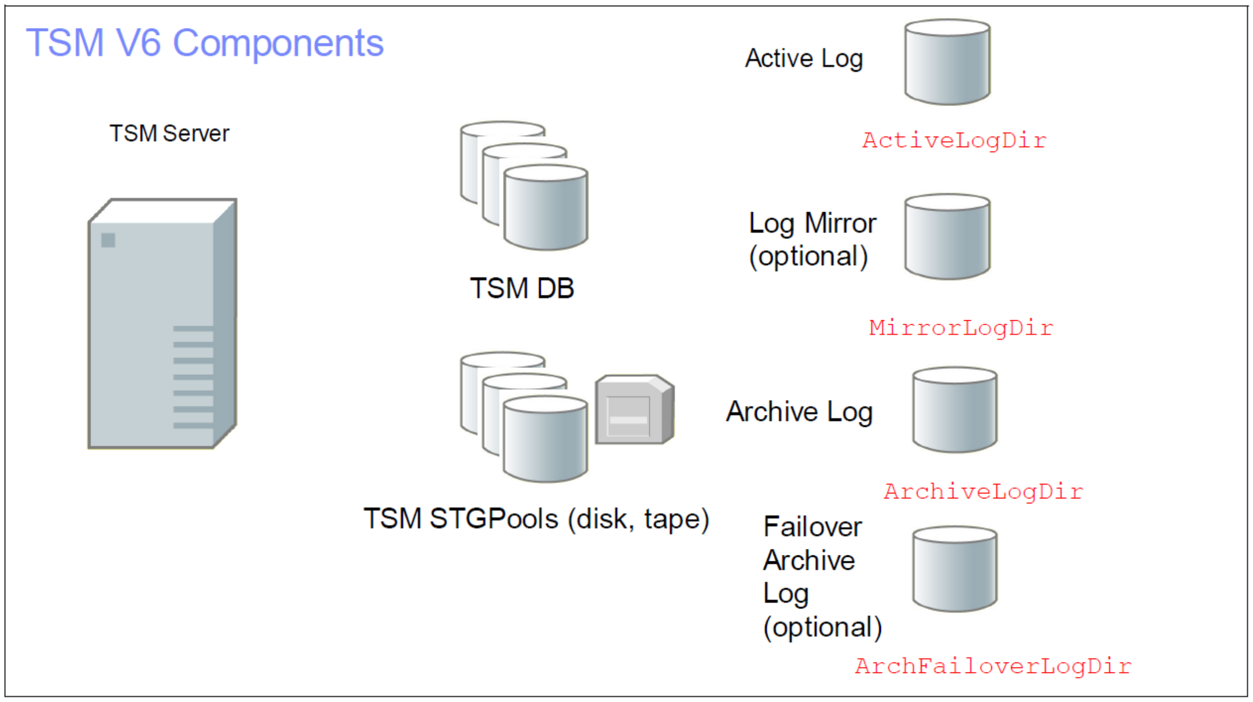
- Tivoli Storage Manager administration should not become more complex than it is with the current database. You do not want additional DBA skills to be required for administration of Tivoli Storage Manager and to complete the tasks you do already today.
- Conversion from the proprietary database to the new database should be automated and should not require an unreasonable amount of time.
- Performance should be comparable to or better than that with the proprietary database.
- Existing and supported storage agents must still be able to perform remote database operations.

Summarizing all the foregoing considerations, we identified these value propositions:

- Scalability: DB2 scales to much larger sizes and continues to be enhanced in this regard.
- Performance: There are many more tuning capabilities within DB2 than with the proprietary database.
- Recoverability: Many more tools and techniques are available for repairing or recovering DB2 databases than those that exist for the proprietary database.
- Monitoring and automation: A complete and robust implementation of SQL will now be available for customers to monitor and solicit information from Tivoli Storage Manager.
- Database license: There is no additional expense for a database license.

- Repair and recovery tools: Having a complete SQL implementation as well as the other features offered by DB2 eliminates the current techniques used to diagnose and repair the database, making the offline audit of the database obsolete.

Here we discuss the items to consider when configuring the DB2 database and log directories. Figure shows the Tivoli Storage Manager V6.1 components involved: database, active log directory, mirror log directory, archive log directory, and the archive failover directory.



Estimating database space requirements

Estimate database space requirements for a new server

The size of the database depends on the number of client files to be stored and the method by which the server manages them.

If you can estimate the maximum number of files that might be in server storage at any time, you can estimate the database size from the following information:

- Each stored version of a file requires about 600 to 1000 bytes of database space.
- Each cached file, copy storage pool file, and active-data pool file requires about 100 to 200 bytes of database space.
- Overhead can require up to 25% in additional space.

In the following examples, the computations are probable maximums. In addition, the numbers are not based on using file aggregation. In general, aggregation of small files reduces the required database space. Assume the following numbers for a Tivoli Storage Manager system:

- The size of the database depends on the number of client files to be stored and the method by which the server manages them.
- If you can estimate the maximum number of files that might be in server storage at any time, you can estimate the database size from the following information
- Each stored version of a file requires about 600 to 1000 bytes of database space.
- Each cached file, copy storage pool file, and active-data pool file requires about 100 to 200 bytes of database space.
- Overhead can require up to 50% in additional space.

Versions of files

The following considerations apply:

- Backed up files:

Up to 500,000 client files might be backed up. Storage policies call for keeping up to three copies of backed up files:

500.000 files x 3 copies = 1.500.000 files

- Archived files:

Up to 100,000 files might be archived copies of client files.

- Space-managed files:

Up to 200,000 files migrated from client workstations might be in server storage.

File aggregation does not affect space-managed files.

At 600 bytes per file, the space required for these files is: (1.500.000 + 100.000 + 200.000) x 600 = 1.0GB

Cached, copy storage pool, and active-data pool files

The following considerations apply:

- Cached copies:

Caching is enabled in a 5 GB disk storage pool. The pool's high and low migration thresholds are 90% and 70% respectively. Thus, 20% of the disk pool, or 1 GB, is occupied by cached files.

If the average file size is about 10 KB, about 100,000 files are in cache at any one time.

100.000 files x 200 bytes = 19 MB

- Copy storage pool files:

All primary storage pools are backed up to the copy storage pool:

(1.500.000 + 100.000 + 200.000) x 200 bytes = 343 MB

- Active-data pool files:

All the active client-backup data in primary storage pools is copied to the active-data pool.

Assume that 500,000 versions of the 1,500,000 backup files in the primary storage pool are active.

500.000 x 200 bytes = 95 MB

Therefore, cached files, copy storage pool files, and active-data pool files require about 0.5 GB of database space.

Overhead

About 1.5 GB is required for file versions, cached copies, copy storage pool files, and active-data pool files. Allow up to 50% additional space (or 0.7 GB) for overhead.

The database should then be approximately 2.2 GB at a minimum.

If you cannot estimate the numbers of files, you can roughly estimate the database size as from 1% to 5% of the required server storage space. For example, if you need 100 GB of server storage, your database should be between 1 GB and 5 GB.

During SQL queries of the server, intermediate results are stored in temporary tables that require space in the free portion of the database. Therefore, using SQL queries requires additional database space. The more complicated the queries, the greater the space that is required.

LOG configuration

You specify the logs used by the server with the ACTIVELOGDIR and the ARCHLOGDIR parameters to the DSMSERV [LOAD]FORMAT command. Both parameters are required. For the active log you can, in addition, specify the optional ACTIVELOGSIZE parameter. If you do not specify the active log size, it defaults to 2 GB. The active log directory specifies the directory in which the Tivoli Storage Manager server writes and stores active log files.

There is only one active log location. The name must be a fully qualified directory name. The directory must already exist, it must be empty, and it must be accessible by the user ID of the database manager.

The maximum number of characters is 175. In this context, active log is a Tivoli Storage Manager term, whereas DB2 uses logs and archive logs.

Optionally you can specify the ARCHFAILOVERLOGDIR and MIRRORLOGDIR with either of the format commands. ARCHFAILOVERLOGDIR specifies the directory to be used as an alternate storage location if the ARCHLOGDIR directory is full. MIRRORLOGDIR specifies the directory in which the server mirrors the active log (those files in the ACTIVELOGDIR directory). For both, the same restrictions apply as with the ARCHLOGDIR.

Active log directory

The following considerations apply:

- It contains current in-flight transaction data.
- I/O characteristic is sequential.
- Usage is required.
- Fixed-size non-circular log type is used.
- Roll-forward mode is supported only.
- Active log files are created in 512 MB sized files. The number of logs created is determined by ACTIVELOGSIZE / 512. If you specified an odd number with the ACTIVELOGSIZE, the value is rounded up to the next even number.
- If a transaction is not committed and all active log files are filled, the server halts.
- The default ActiveLogSize is 2 GB, maximum supported value is 128 GB.
- The ACTIVELOGDIR value can be changed in the dsmserv.opt file (requires server restart), and the initial value is taken from the format command.

Active log mirror directory

The following considerations apply:

- It contains mirrored copies of active transaction data.
- I/O characteristic is sequential.
- Use is optional but recommended.
- Mirror log files are created in 512 MB sized files.

- If a mirror log directory becomes full, a message is issued, and the server continues.
- The MIRRORLOGDIR value can be changed in the dsmserv.opt file (requires server restart), and the initial value is taken from the format command if specified.

Archive log directory

The following considerations apply:

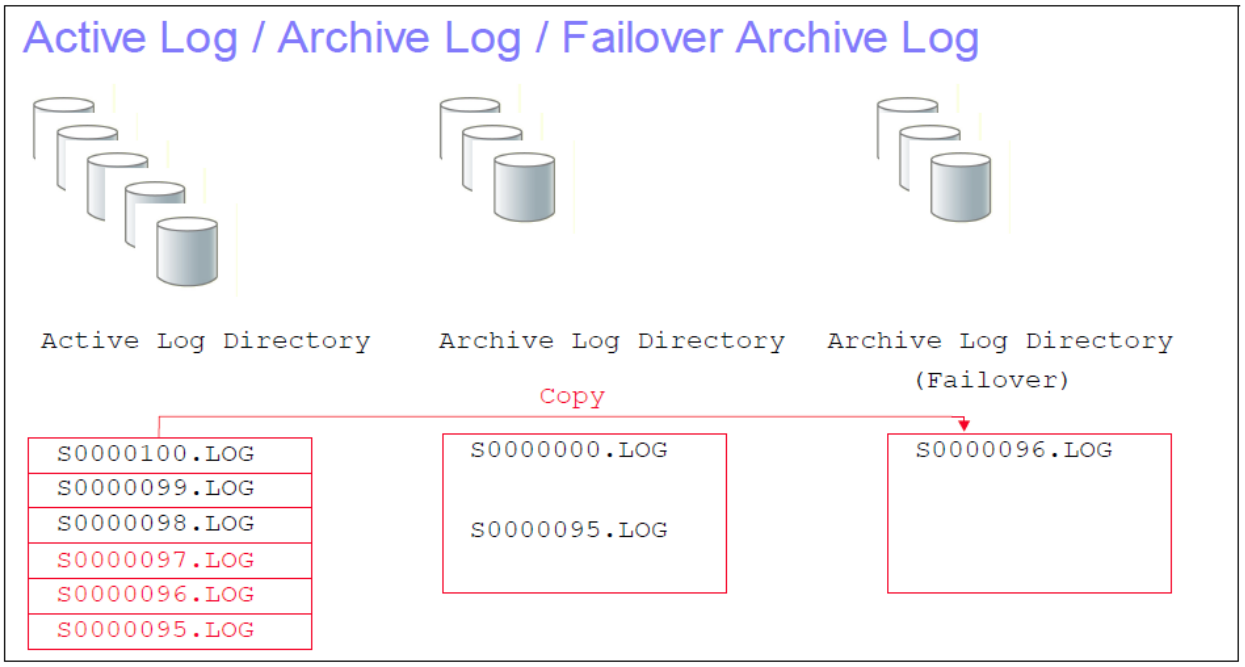
- It contains committed transaction data.
- I/O characteristic is sequential.
- Usage is required.
- It can have up to three full backups worth of space for archive logs.
- It is required to roll forward transactions after a database restore
- Log files older than two full backups ago are removed after DB backup.
- The ARCHIVELOGDIR value can be changed in the dsmserv.opt file (requires server restart), and the initial value is taken from the format command

Archive failover log directory

The following considerations apply:

- Used in case Archive Log Directory becomes full to hold archive logs.
- I/O characteristic is sequential.
- Use is optional.
- If in use, logs will be moved back to ACTIVELOGDIR location for a DB restore.
- Log files are removed after DB backup.
- The ARCHFAILOVERLOGDIR value can be changed in the dsmserv.opt file (requires server restart), and the initial value is taken from the format command if specified.

The log file flow is illustrated in Figure below. When they are full, the log files are closed by DB2 and get copied to the archive log directory, transactions might still be active when the file gets archived. The server continues to copy full log files to the archive log directory until the directory becomes full, then copies will go to the failover archive log directory if defined. If even the failover archive log directory fills up, for example, because of unexpected workload, the active logs will retain in the active log directory. This can result in an out of log space condition and a server halt if the active log directory fills up, too.



3.7 Administration interfaces and administration authority

3.8 TSM Licenses

To check for license compliance, use the **query license** command

You can register licenses for server components. This includes:

1. Tivoli Storage Manager (base): **tsmbasic.lic**

- For basic backup-archive with the base client and server.

2. Tivoli Storage Manager Extended Edition: **tsmee.lic**

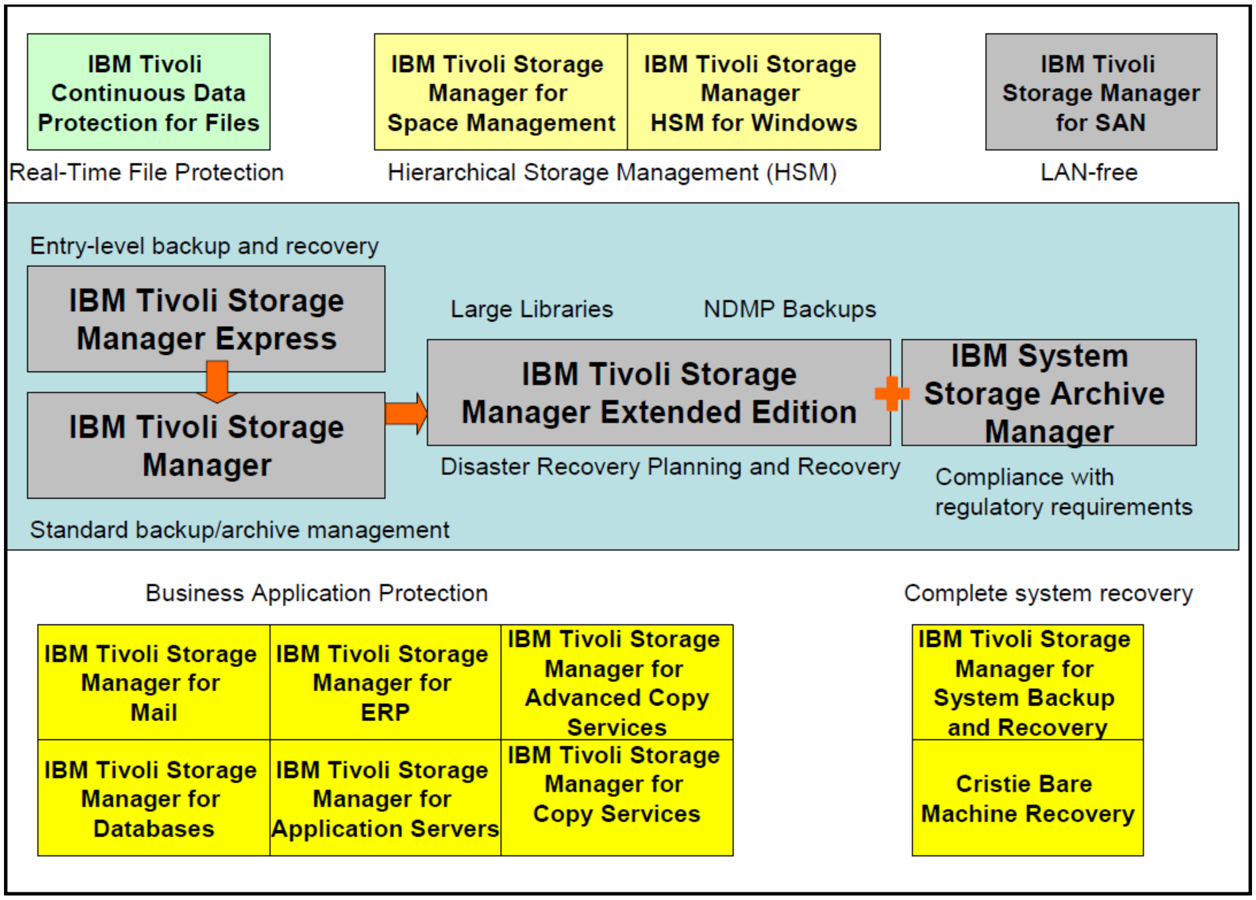
- For additional advanced functions, including:
- Server-free data movement
- Disaster recovery Manager
- Large libraries (greater than 3 drives or 40 slots)
- Tape Library Sharing over LAN
- NDMP backup and restore for NAS appliances

3. Tivoli Data Retention Protection: **dataret.lic**

- Meets additional requirements defined by the regulatory agencies for retention and disposition of data. It has additional functionality in:

- Data retention protection
- Event-based Retention Management
- Expiration/Deletion suspension (Deletion hold)

3.9 TSM Related Products


IBM Tivoli Continuous Data Protection for Files
Real-Time File Protection
IBM Tivoli Storage Manager for Space Management
IBM Tivoli Storage Manager HSM for Windows
Hierarchical Storage Management (HSM)
IBM Tivoli Storage Manager for SAN
LAN-free
Entry-level backup and recovery
IBM Tivoli Storage Manager Express
IBM Tivoli Storage Manager
Standard backup/archive management
Large Libraries
NDMP Backups
IBM Tivoli Storage Manager Extended Edition
Disaster Recovery Planning and Recovery
IBM System Storage Archive Manager
Compliance with regulatory requirements
Business Application Protection
IBM Tivoli Storage Manager for Mail
IBM Tivoli Storage Manager for ERP
IBM Tivoli Storage Manager for Advanced Copy Services
IBM Tivoli Storage Manager for Databases
IBM Tivoli Storage Manager for Application Servers
IBM Tivoli Storage Manager for Copy Services
Complete system recovery
IBM Tivoli Storage Manager for System Backup and Recovery
Cristie Bare Machine Recovery

IBM Tivoli Storage Manager can be integrated with a number of optional applications that together form a powerful integrated storage management solution. These include:

- BM Tivoli Storage Manager for Space Management
- IBM Tivoli Storage Manager for HSM for Windows
- IBM Tivoli Storage Manager for Storage Area Networks
- BM Tivoli Continuous Data Protection for Files
- IBM Tivoli Storage Manager for System Backup and Recovery
- IBM System Storage™ Archive Manager

Tivoli Storage Manager products line:

- **Storage FlashCopy Manager**

Integrated recovery management solution for IBM Storage systems

- **Storage Manager**

Automates data backup, restore and archive functions, centralizes storage management operations

- **Storage Manager Extended Edition**

Expands on backup, restore and archive abilities with disaster recovery functionality

- **Storage Manager FastBack**

Next-generation data protection and recovery in remote offices and data centers

- **Storage Manager FastBack Center**

Full-featured data protection and recovery suite for mid-sized businesses and remote offices

- **Storage Manager FastBack for Bare Machine Recovery**

Restore a Windows Server OS volume within an hour

- **Storage Manager FastBack for Microsoft Exchange**

Fast and easy recovery of individual e-mail objects, like messages and contacts

- **Storage Manager HSM for Windows**

Hierarchical storage management software for automatically migrating rarely used files

- **Storage Manager for Databases**

Secures Informix, Oracle and Microsoft SQL data

- **Storage Manager for Enterprise Resource Planning**

Protects vital SAP R/3 system data more efficiently, consistently and reliably

- **Storage Manager for Mail**

Secures Lotus Domino and Microsoft Exchange data

- **Storage Manager for Microsoft SharePoint**

Data protection for Microsoft SharePoint environments

- **Storage Manager for Space Management**

Software that moves inactive data to reclaim online disk space

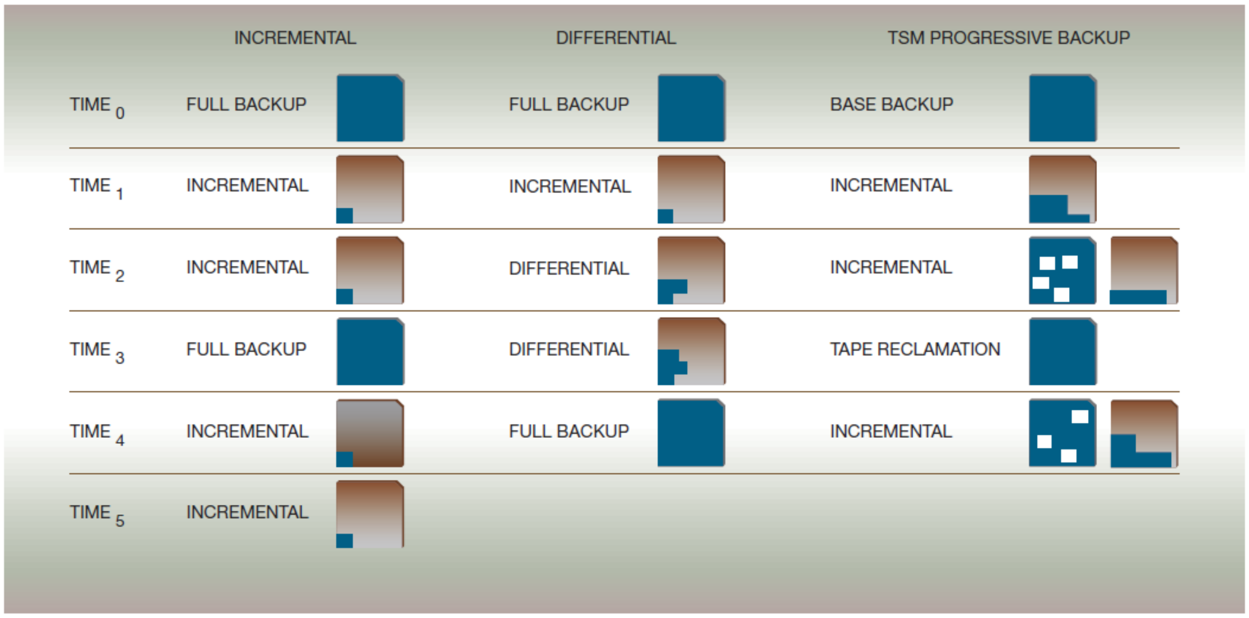
- **Storage Manager for Storage Area Networks**

Maximizes storage network connections for Tivoli Storage Manager servers and client computers

- **Storage Manager for System Backup and Recovery**

Offers system backup, restore, and reinstallation providing bare metal restore capabilities

4. Backup Methodology – Incremental forever



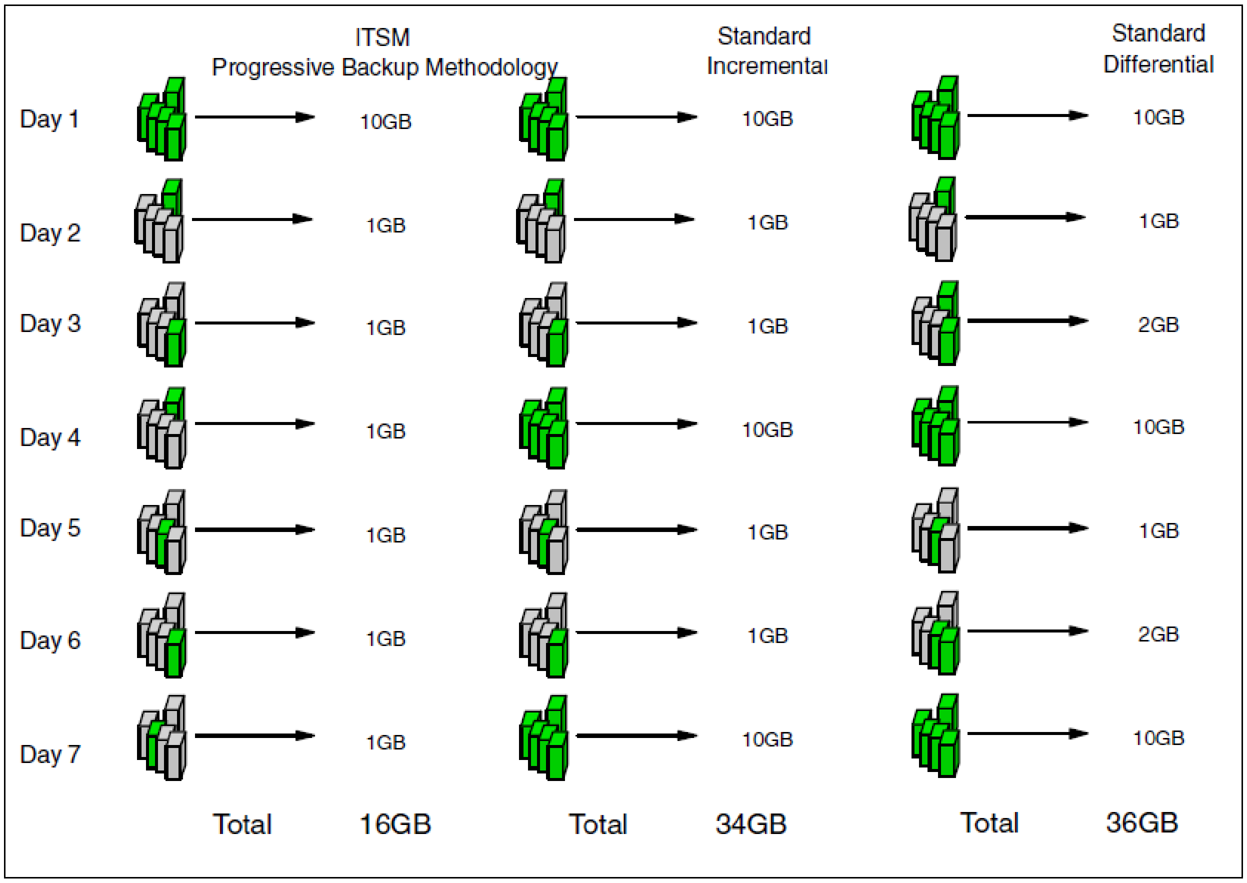
The figure illustrates the most common consolidation methods in use in comparison with the progressive (or incremental forever) methodology of TSM. Increasing points in time are displayed on the left as times T0 through T5. Each column in the figure represents a different backup technique, with use of tape for backup storage depicted as square tape cartridges. The dark areas on the cartridges represent used portions of tape, whereas lighter regions represent unused tape or regions of tape that are no longer valid because the file copies in these areas are no longer needed. Incremental backup processing is shown in the first column. The technique involves periodic full backup operations that copy all data (at times T0 and T3), interspersed with "incremental" backup operations that only copy data that have changed since the last full or incremental backup operation (times T1, T2, T4, and T5). Although relatively efficient for backup processing, the technique can require the highest number of tape mount operations when restoring data. A full restore operation needed shortly after time T2 but before time T3, for example, would have to restore the data from tapes created at times T0, T1, and T2.

Differential backup processing, column 2 in the figure, is similar to incremental backup processing. Periodic full backup operations (times T0 and T4) are interspersed by "differential" backups (times T1, T2, and T3), which copy all data that have changed since the last full backup operation. Restore operations are more efficient than with incremental processing because fewer volumes need to be mounted, but differential backup still requires unchanged data to be repetitively copied (or backed up) from the client to the server. A full restore operation after time T2 in the differential model would need data from created at times T0 and T2 (since the tape at time T2 also contains the data copied to the tape created at time T1).

Incremental and differential processing requires that all client data be resent periodically so that new tapes can be created and those written on in earlier operations can be made available for reuse. TSM backup processing, illustrated in column 3, is incremental in nature for every backup operation after the first full backup (T0). As changed file copies are sent to the server, earlier versions are no longer needed. These older copies represent logical empty spaces in the server tapes and are represented by white squares in the figure. Instead of resending all client data to consolidate space on tape, the TSM server automatically reclaims the tape volumes written in earlier operations by copying valid (unexpired) data to new volumes (shown at time T3). The emptied tapes are then made available for reuse. The reclamation and co-location process (described later) continually reorganizes data on tape volumes so that restore operations are optimized and a minimal amount of data needs to be sent from the client during backup operations.

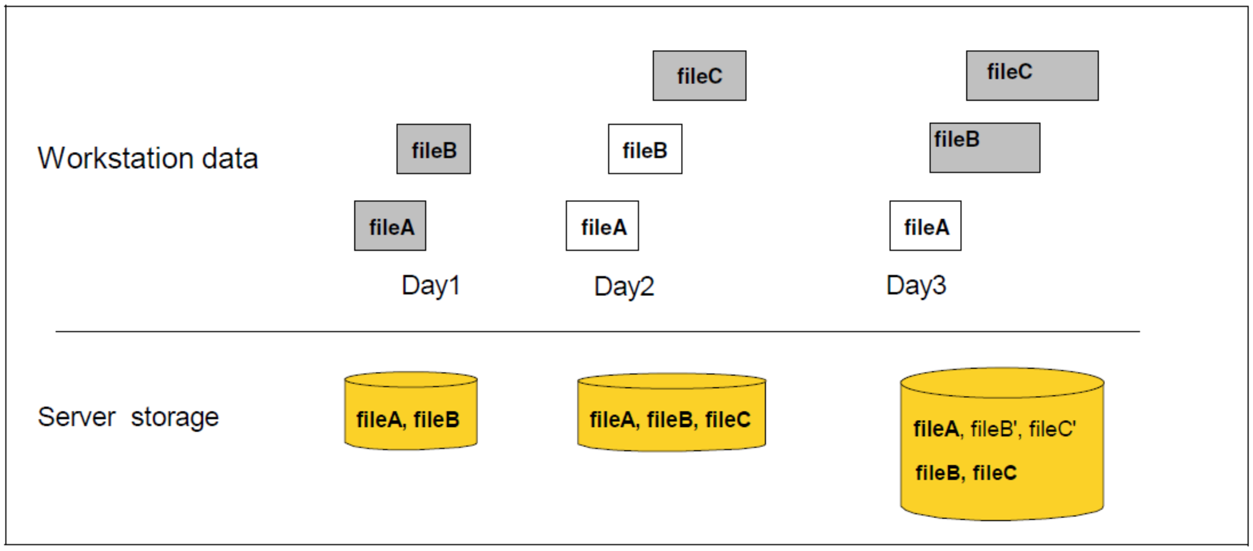
The adaptive subfile differencing technology of TSM extends the philosophy of sending only changed data for mobile user connections in extremely slow communications environments such as phone lines.

The differencing technique is applied to certain files (as specified by the user or the administrator) so that only changed portions of updated files are sent. After an initial backup of the entire file (the base), changes are calculated by comparing the base file with the current version of the file.1 If the changed portion is small enough (less than 40 percent of the total file size), only the changed data (a delta) are sent, along with information on the base file on which the changed portion is dependent.
The server tracks the relationship between file deltas and their dependent base versions so that the proper file image can be reconstructed correctly during a restore operation. TSM also ensures that the base version of a file exists when a delta is backed, up. If the base is not found (because of inadvertent deletions or damage caused by volume handling, for example), the server forces the client to resend the entire file.



5. Backup Types

5.1 Incremental backups



The incremental backup operation is a full scan of the client's file systems, which backs up all files and other information (and only those things) necessary to ensure that the Tivoli Storage Manager inventory matches the current state of the client's storage. The first time this operation is run on a new client, everything is backed up. On each subsequent incremental backup, only new and changed files are sent. During the incremental backup, the client queries the Tivoli Storage Manager server so that it knows what files are currently stored. The client uses this information to:

- Back up new files
- Back up files whose contents have changed
- Expire backup versions on the server for files that were deleted from the

Workstation

The Figure shows an example of a daily incremental backup. On Day 1, two files (fileA and fileB) exist on the client, and are therefore backed up. On Day 2, fileC is newly created, and therefore it is backed up. fileA and fileB have not changed so are not backed up. On Day 3, fileB and fileC have changed, so they are backed up, because they are the only file that have changed since the last backup. In our example, fileA has never changed, so Tivoli Storage Manager only stores one copy of this file. However, the changed files, fileB and fileC, have two copies stored in the server.

5.2 Incremental (Complete)

Full Drive Incremental Backup.

The first time you run a full incremental backup, Tivoli Storage Manager backs up all the files and directories on the drives you specify. This process can take a long time if the number of files is large, or if one or more very large files must be backed up. Subsequent full incremental backups will only back up new and changed files. This allows the backup server to maintain current versions of your files, without having to waste time or space by backing up files that already exist in server storage.

Depending on your storage management policies, the server might keep more than one version of your files in storage. The most recently backed up files are active backup versions. Older copies of your backed up files are inactive versions. However, if you delete a file from your workstation, the next full incremental backup will cause the active backup version of the file to become inactive. If you need to restore a file you have deleted, and if a full incremental backup has been run since you deleted the file, then you will need to restore an inactive version of the file (assuming that a version still exists on the server). The number of inactive versions maintained by the server and how long they are retained is governed by the management policies defined by your server administrator. The purpose of the active versions is to represent which files existed on your file system at the time of the backup

5.3 Incremental-by-Date

For a file system to be eligible for incremental-by-date backups, you must have performed at least one full incremental backup of that file system. Running an incremental backup of only a directory branch or individual file will not make the file system eligible for incremental-by-date backups.

To perform an incremental-by-date backup using the GUI, select the ***Incremental (date only)*** option from the *type of backup* pull-down menu or use the ***incrbydate*** option with the **incremental** command.

The client backs up only those files whose modification date and time is later than the date and time of the last incremental backup of the file system on which the file resides. Files added by the client after the last incremental backup, but with a modification date earlier than the last incremental backup, are not backed up.

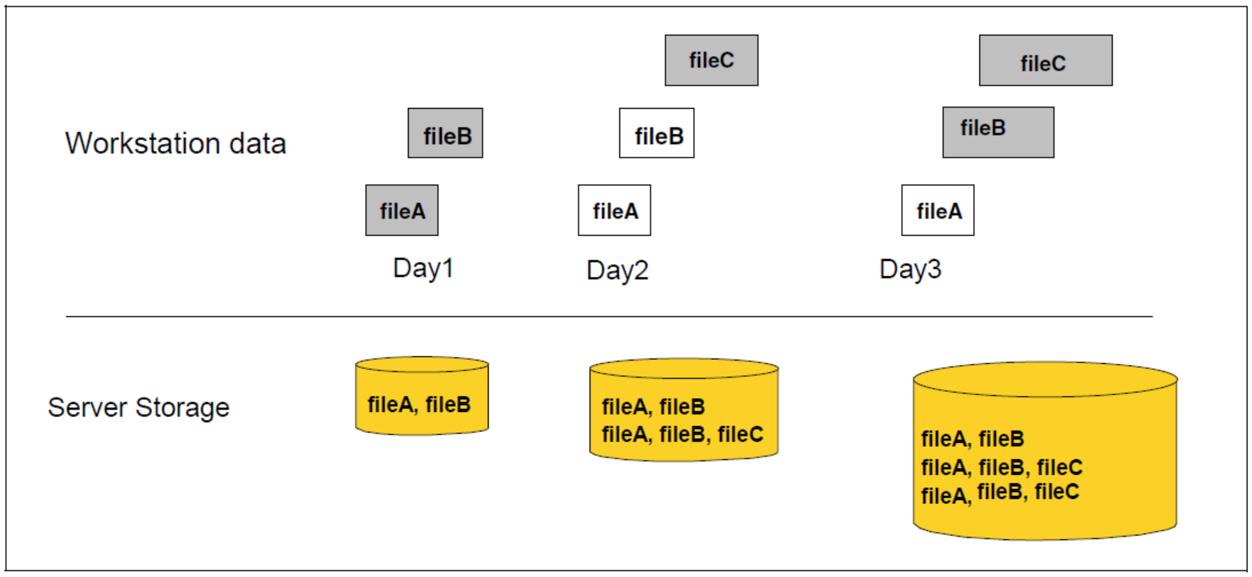
Files that were renamed after the last incremental backup, but otherwise remain unchanged, will not be backed up. Renaming a file does not change the modification date and time of the file. However, renaming a file does change the modification date of the directory in which it is located. In this case, the directory is backed up, but not the files it contains.

If you run an incremental-by-date backup of the whole file system, the server updates the date and time of the last incremental backup. If you perform an incremental-by-date backup on only part of a file system, the server does not update the date of the last full incremental backup. In this case, the next incremental-by-date backup will back up these files again.

5.4 Incremental (without Journal)

Incremental (without journal) is a incremental backup without using the journal database. If you installed the journal engine service and it is running, then by default the incremental command will automatically perform a journal-based backup on selected file systems which are being monitored by the journal engine service. This option lets you perform a traditional full incremental backup, instead of the default journal-based backup.

5.5 Always (Selective) Backups



Use a selective backup when you want to back up specific files or directories regardless of whether a current copy of those files exists on the server. Incremental backups are generally part of an automated system to back up entire file systems. In contrast, selective backups allow you to manually select a set of files to back up regardless of whether they have changed since your last incremental backup. Selective backup still applies file versioning.

Unlike Incremental backups, a selective backup:

- Does not cause the server to update the date and time of the last incremental.
- Backs up directory and file entries even if their size, modification timestamp, or permissions have not changed.

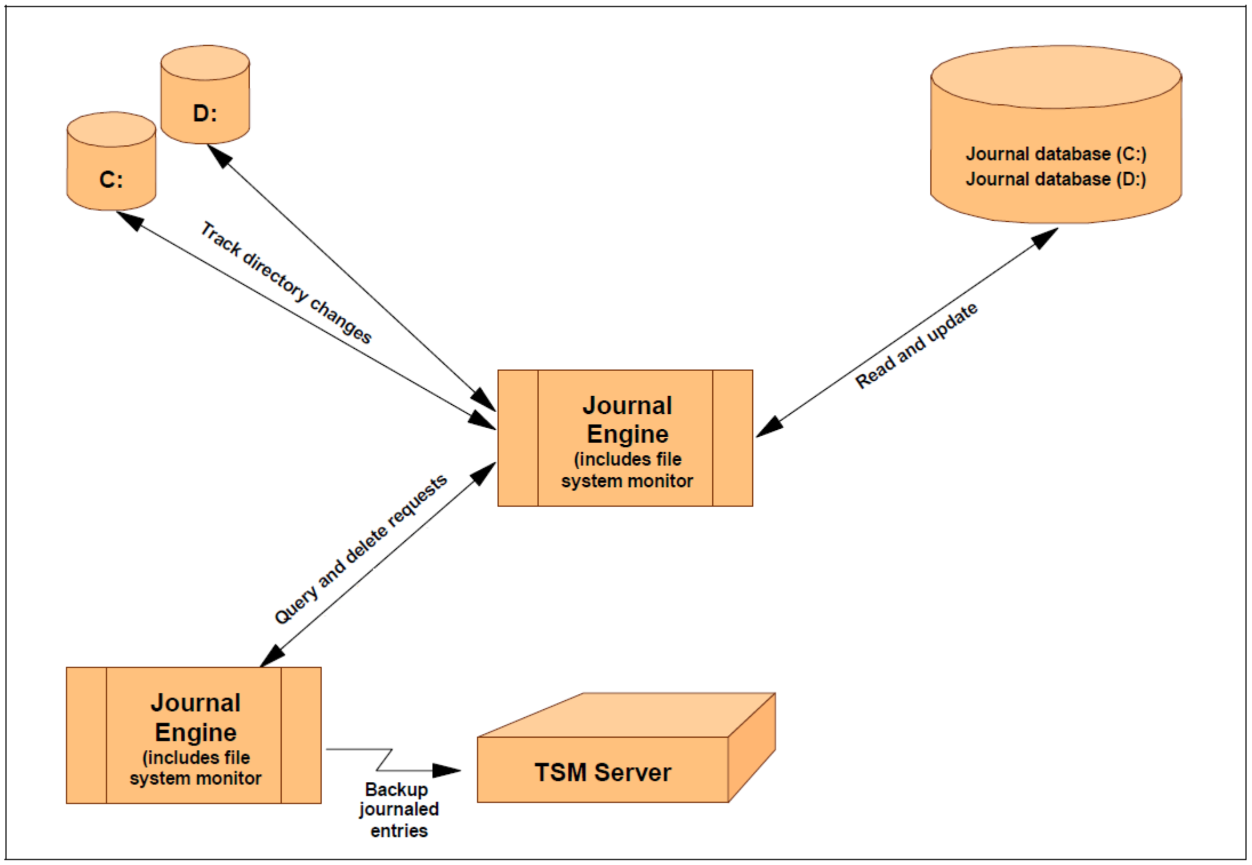
5.6 Journal-based backups

Journal-based backup provides an alternative to traditional progressive incremental backup, which under certain circumstances may dramatically increase overall backup performance.

As the name already implies, journal-based backups have no effect on archive processing. The main difference between journal-based backup and progressive incremental backup is the method in which the list of backup candidate objects is derived.

A *backup candidate list* specifies objects for a particular file system that are to be backed up, expired, or updated on the Tivoli Storage Manager server by a backup-archive client.

The progressive incremental backup operation derives the backup candidate list by building and comparing the list of active previously backed-up objects stored on the Tivoli Storage Manager server with the list of objects currently residing on the local file system.



The server list is obtained over the network and the local list is obtained by scanning the local file system. Objects that exist in the local list but do not exist in the server list are added as backup candidates to the candidate list.

Objects that exist in both lists but differ in some way (such as attributes, policy, and size) are also added as backup candidates unless only the Tivoli Storage Manager database attributes differ, in which case they are added as attribute update candidates.

Objects that exist in the server list but not in the local list are added as expiration candidates.

The Tivoli Storage Manager backup-archive client obtains the backup candidate list by contacting the Journal Based Backup Daemon. This is a local background process that manages and maintains a journal database of change activity for each file system being journaled.

Journal database entries are generated by real-time file system change activity, and they specify objects to back up or expire. (Attribute update actions are not currently supported.) The type of change activity that generates journal entries is configurable by the user and may consist of any combination of these changes:

- Objects created, deleted, or renamed on the file system
- Size changes
- Modification time/date changes
- Access time/date changes
- Attribute changes
- Security (ACL) changes

Once a journal entry has been processed successfully, the Tivoli Storage Manager backup-archive client notifies the Journal Based Backup Daemon to remove the journal entry from the journal database.

If the Journal Engine Service is installed and running, then by default the **incremental** command performs a journal-based backup on any journaled file systems.

Advantages of journal-based backup

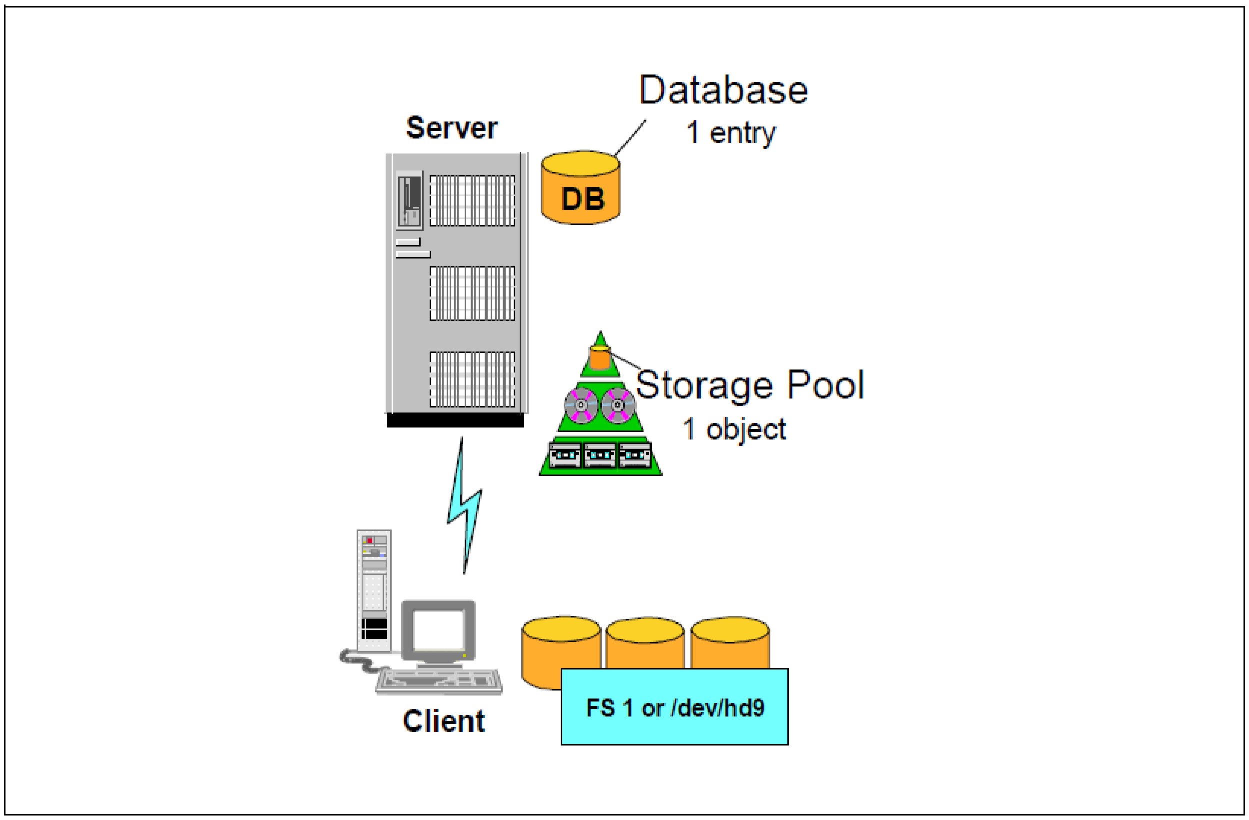
Journal-based backup can improve incremental backup performance in most environments.

With journal-based backup, the client does not scan the local file system or obtain information from the server to determine which files to process. As such, journal-based backup reduces network traffic between the client and server. The backup-archive client, as always, still sends data (files) to the Tivoli Storage Manager server and, as has always been the case, the Tivoli Storage Manager server stores file details and location in the Tivoli Storage Manager database.

As the backup-archive client does not carry out the initial metadata conversation, the backup-archive client does not have to sit idle. The backup-archive client can begin sending the files to the Tivoli Storage Manager server as soon as the journal-based backup is initiated. This means faster backup times and less backup-archive client idle time.

5.7 Image and Logical volume backups

Tivoli Storage Manager enables you to back up a file system or raw logical volume as a single object from your client machine. The Tivoli Storage Manager client accomplishes this by dynamically loading and image plug-in utility that sends the object to the server using the Tivoli Storage Manager API. This capability is currently available for the **AIX, HP-UX, Solaris, Linux, and Windows** clients and can be used on a logical volume whether or not there is an associated file system. This will ensure a clean backup. On Windows clients, an additional service is provided with the Tivoli Storage Manager client. Figure shows the operation of the image backup and restore operation as a single object.



Logical volume backup has the advantages of improved backup and restore speeds and conserving server resource because the entire backup is treated as a single object and individual files are not processed during backup or restore. Similar to the standard incremental and selective backup filtering options, you can include specific logical volumes, assign a Management Class to image objects, and exclude file systems or a raw device from being backed up by specifying in the client include/exclude list as in the following examples:

```
INCLUDE.IMAGE /.../*
EXCLUDE.IMAGE /dev/hd5
```

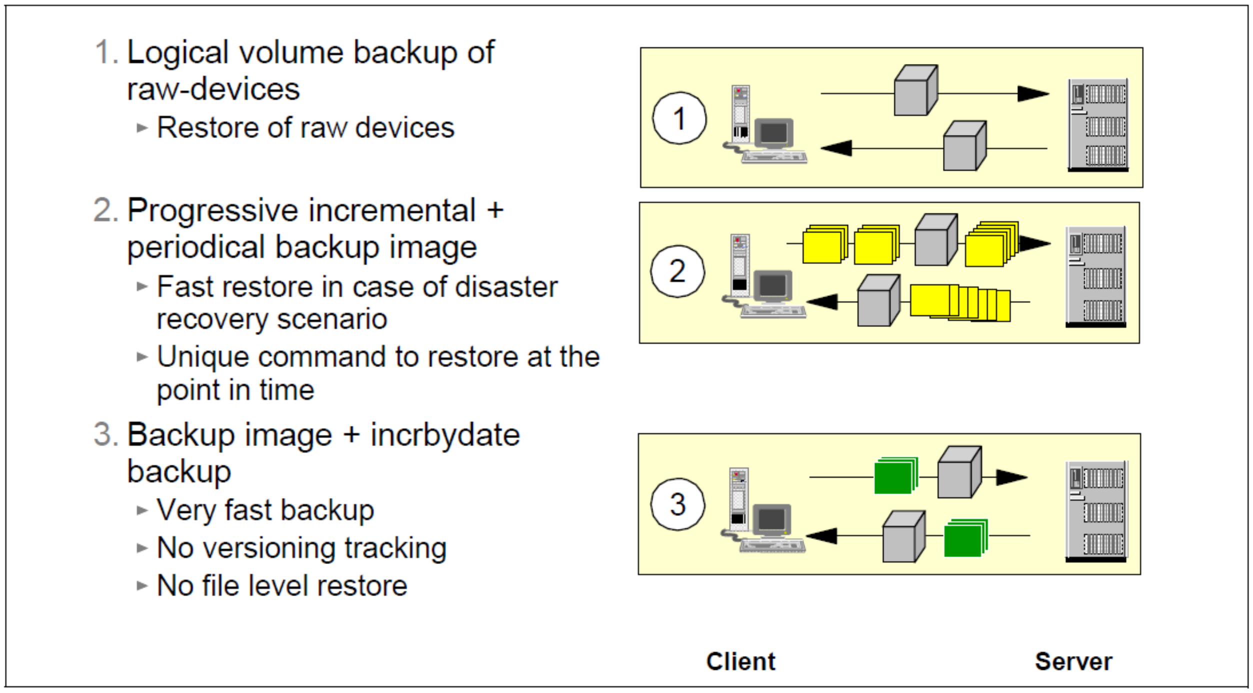
Setting the Copy Serialization parameter in your Management Class as Static (or Shared Static), directs that the file system (if one exists) will be unmounted first and remounted automatically as read-only before the backup . When the image backup is completed, the file system is remounted as it was originally. If the Copy Serialization parameter in your Management Class is Dynamic (or Shared Dynamic), the client performs the backup of the file system even if it is in use. This is not recommended, as it will probably lead to an inconsistent backup image. The automatic mounting and unmounting operations are not applicable for raw devices image backup because there is no associated file system.

You can consider full logical volume backup (mode=selective) on its own or in combination with progressive backup operations based on your requirements, as in the following:

- Perform regular logical volume backup for raw devices used for application managed data spaces. Examples are offline backup of raw logical volumes used in database applications. Note that there is no incremental option in raw logical volume backup.

- Perform a combination of progressive and occasional image backups of your file system. This provides fast recovery as well as file level restore capability.

- Perform a combination of daily incremental-by-last-image-date backup with periodic image backup. The MODE option in the backup image command determines whether the backup is a full file system image backup (selective) or an incremental-by-last-image-date backup (incremental). This provides fast backup and recovery of the entire file system. File level restore is limited to files changed since the last full image backup. It is important to note that for incremental-by-last-image-date backup to work, it must not have any previous incremental backup of the file system using the **incremental** command



1. Logical volume backup of raw-devices
   - Restore of raw devices
2. Progressive incremental + periodical backup image
   - Fast restore in case of disaster recovery scenario
   - Unique command to restore at the point in time
3. Backup image + incrbydate backup
   - Very fast backup
   - No versioning tracking
   - No file level restore

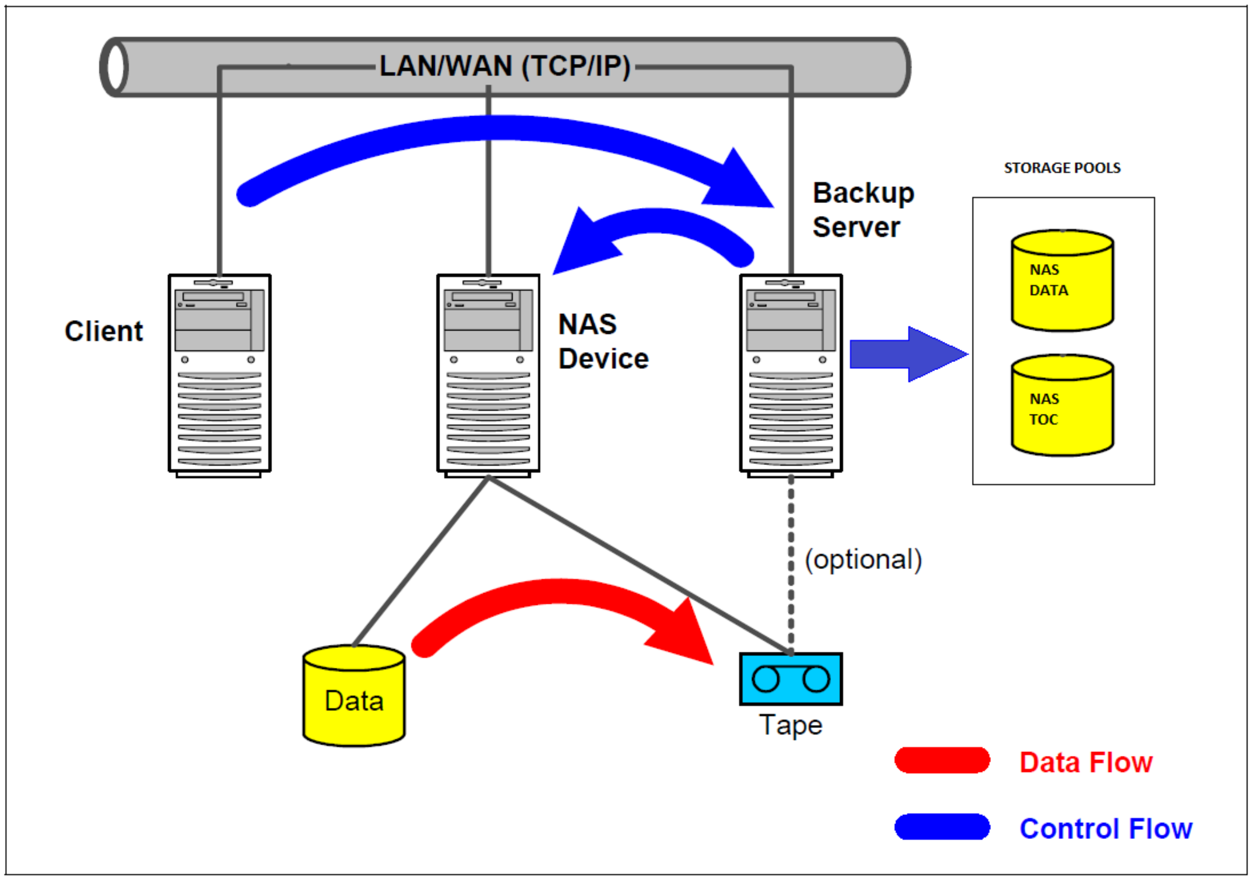
Only a UNIX root user can perform the backup and restore image operations. When you restore from an image backup, the entire previous contents of the logical volume or file system will be overwritten.

On the Windows client platforms, a Logical Volume Storage Agent (LVSA) is available that can take a snapshot of the volume while it is online. Optionally, only occupied blocks can be copied. If the snapshot option is used (rather than static), then any blocks that change during the backup process are first kept unaltered in an Original Block File. In this way the client sends a consistent image of the volume as it was at the start of the snapshot process to the Tivoli Storage Manager server.

In summary, an image backup provides the following benefits:

- Provide a quicker backup and restore than a file-by-file backup, as there is no overhead involved in creating individual files
- Conserve resources on the server during backups, because only one entry is required for the image
- Provide a point-in-time picture of your file system, which is useful if your enterprise needs to recall that information
- Restore a corrupt file system or raw logical volume, restoring data to the same state it was when the last logical volume backup was performed

5.8 NAS File system backups (NDMP)



IBM Tivoli Storage Manager Extended Edition can perform Network Data Management Protocol (NDMP) backups. NDMP is an industry-standard protocol that enables a network storage-management application to control the backup and recovery of an NDMP-compliant file server without installing third-party software on that server. This provides backup and recovery support for NAS file servers from Network Appliances. NAS file servers often require a unique approach to providing backup and recovery services, because these file servers are not typically intended to run third-party software.

The NAS file server does not require Tivoli Storage Manager software to be installed. Instead, the Tivoli Storage Manager server uses NDMP to connect to the NAS file server to initiate, control, and monitor a file system backup or restore operation as shown in the Figure. The implementation of the NDMP server protocol enables the NAS file servers to be backup-ready and enables higher-performance backup to tape devices without moving the data over the LAN. The tape devices have to be under the direct control of the NAS filer, which means that they have to be directly attached or connected through a supported SAN environment.

An NDMP backup is usually an image backup because the NAS filer performs the backup as an entity without informing the Tivoli Storage Manager about the content. So the Tivoli Storage Manager server administers only one image object that has been backed up. Additionally, Tivoli Storage Manager can create a table of contents (TOC) during backup and stores this TOC afterwards in a dedicated storage pool. This enables Tivoli Storage Manager to perform a single file restore from a NAS backup image.

Each time a single file restore from an NAS image backup is performed, Tivoli Storage Manager will load the TOC from the dedicated storage pool into a temporary database table. This table will be deleted after a specified amount of time that can be configured through the parameter TOCLOADRETENTION. Even without the TOC you can restore single files from a NAS backup image. In that special case, the exact information about the single file and the image it resides in must be provided for restore.

Although an NDMP backup is usually started and controlled by a Tivoli Storage Manager server, the Tivoli Storage Manager Web client can also initiate and control an NDMP backup or restore, respectively. Using a TOC, the Tivoli Storage Manager Web client provides file-level access to the TOC so that it becomes browsable.

Collecting file-level information requires additional processing time, network resources, storage pool space, and possibly a mount point during the backup. You must set up policy so that the Tivoli Storage Manager server stores the TOC in a different storage pool from the one where the backup image is stored. It is important to allocate adequate storage pool space for storing TOCs. To avoid mount delays, use random access storage pools (DISK device class).

5.9 Active and inactive file versions

One of the most important concepts in Tivoli Storage Manager data management is the difference between an active backup version and an inactive backup version.

Assume that a new file is created on your workstation. The next time you run a backup operation (say, Monday at 9 p.m.), Tivoli Storage Manager server backs up this file. This copy of the file is known as the ACTIVE version, since it is the most recent version of the file. When you next run an incremental backup (say, Tuesday at 9 p.m.), Tivoli Storage Manager uses this ACTIVE version already stored to check back with your workstation to determine whether the file has changed since the last backup. If it has, it is backed up again.

This version now becomes the ACTIVE version and the copy from Monday becomes an INACTIVE version. The most recent backed-up version of the file is always the ACTIVE version, *as long as the file itself still exists on the original client*. Tivoli Storage Manager will keep storing a new ACTIVE version and inactivating the previous active version, up to the limit of the total number of versions defined to be retained in the management class. Once this limit is exceeded, the oldest INACTIVE version is deleted from Tivoli Storage Manager storage and will no longer be able to be restored.

This process of maintaining the ACTIVE and INACTIVE versions (up to the management class limit) continues indefinitely, until and unless the file is deleted from the original client. If this occurs, when the next incremental backup is run, the server detects that the file no longer exists on the client. All stored versions of the file now automatically become INACTIVE, and some of the oldest versions of the file may also be deleted. This would happen if the management class setting for number of versions of a deleted file to retain is less than the number of versions of an existing file to retain.

Therefore, an ACTIVE file version is stored along with INACTIVE versions as long as the file is still resident on the client system. If the file is deleted, then only INACTIVE version(s) of the file will exist in server storage. If there is no ACTIVE file version in Tivoli Storage Manager storage, it means that the file has been deleted from the client machine, so the only copy is now in Tivoli Storage Manager storage.

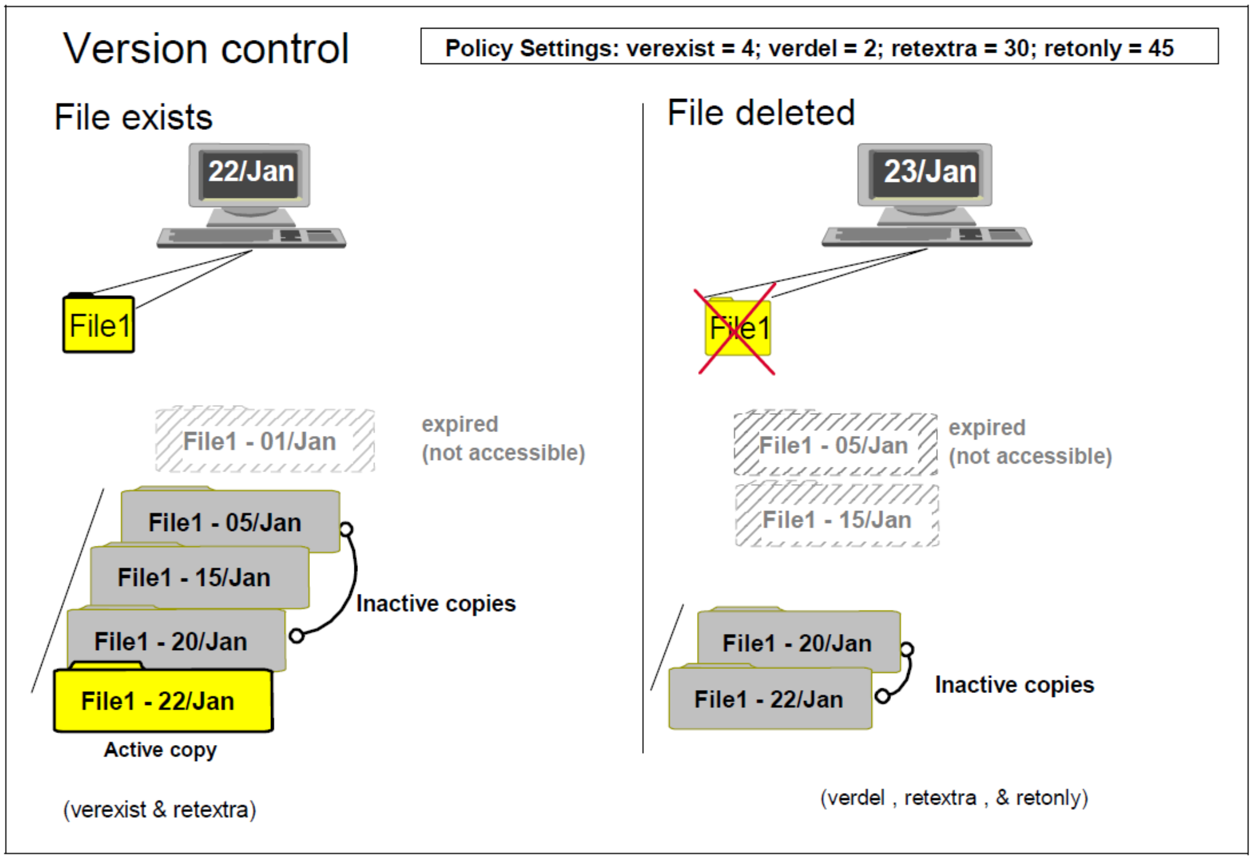
Tivoli Storage Manager controls the retention of its ACTIVE and INACTIVE versions of a file that exists on a client machine by using two criteria defined in the Management Class:

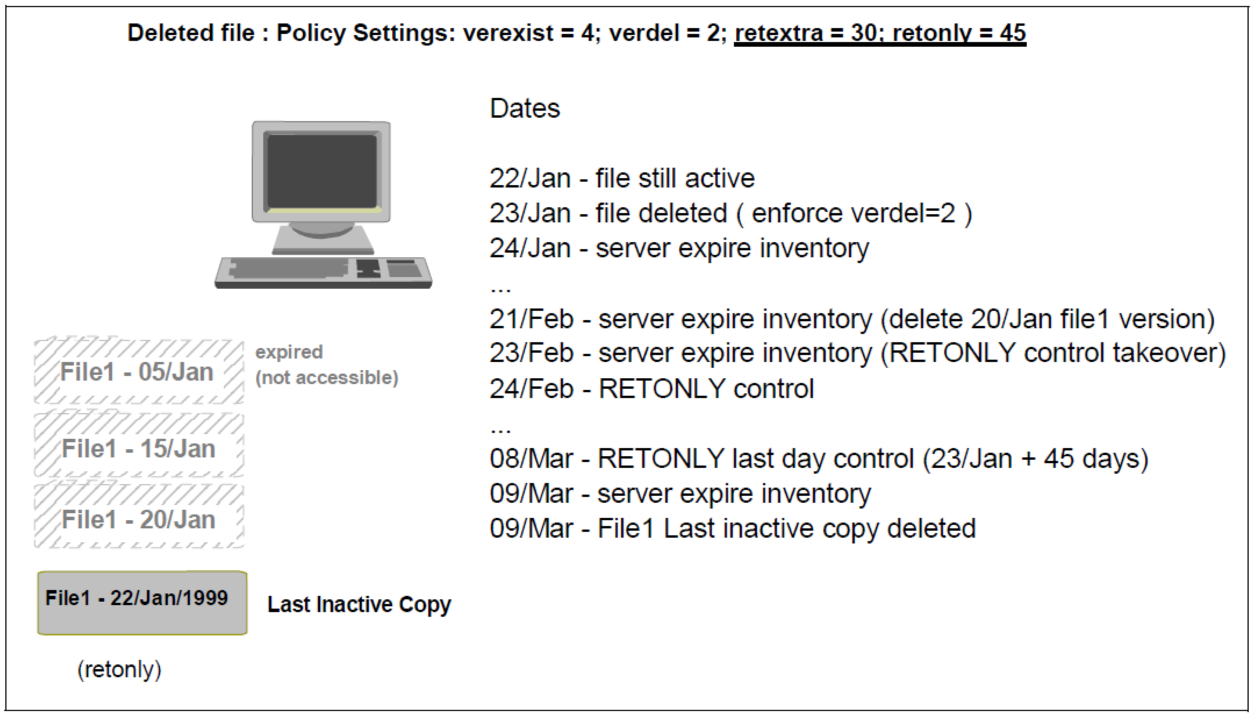
- **How many versions:** The parameter that controls the number of backup versions is called VEREXIST. This may be set at a specific number or to UNLIMITED.
- **How long to keep:** The RETEXTRA parameter controls how much time must elapse before an INACTIVE file version is considered expired. This parameter controls how long to retain all remaining inactive files and may be set at a specific number of days or to NOLIMIT, meaning they will never be expired.

Important: An ACTIVE file version is never expired. Even if you never change a particular file after the first incremental backup, Tivoli Storage Manager will keep this file version indefinitely.

For a file deleted on a client machine, Tivoli Storage Manager uses different criteria:

- **How many versions:** The parameter that controls the number of inactive backup versions is called VERDELETED. This number is normally less than or equal to the number you have for VEREXISTS.
- **How long to keep files:** The RETEXTRA parameter controls how much time must elapse before an INACTIVE file version is considered expired. This parameter controls how long to retain all remaining inactive files except for the last one and may be set at a specific number of days or to NOLIMIT, meaning they will never be expired.
- **How long to retain the last version:** The RETONLY parameter controls the last inactive copy of a file. As files get expired by RETEXTRA, you can configure Tivoli Storage Manager to manage the last inactive copy differently, so that you can keep that file for a longer period of time. It may be set at a specific number of days or to NOLIMIT, meaning they will never be expired. Typically, configure RETONLY to be either the same value or longer than RETEXTRA because it functions as a grace period before expiring the file.

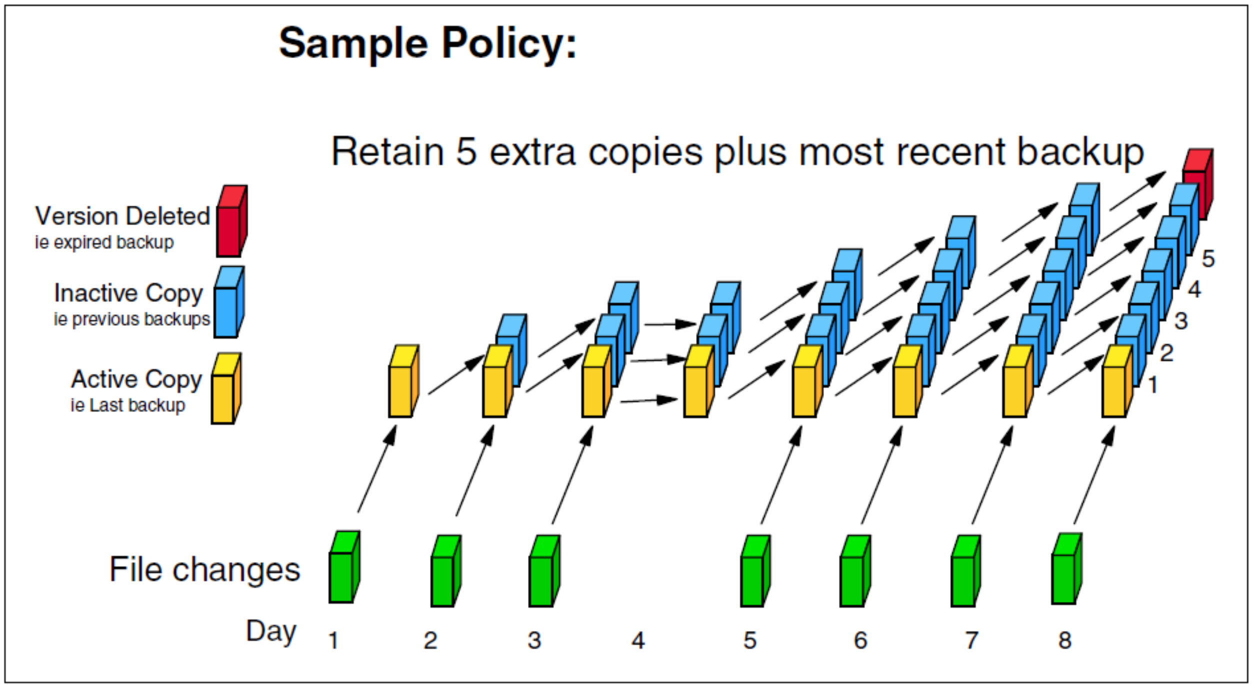




Deleted file : Policy Settings: verexist = 4; verdel = 2; retextra = 30; retonly = 45

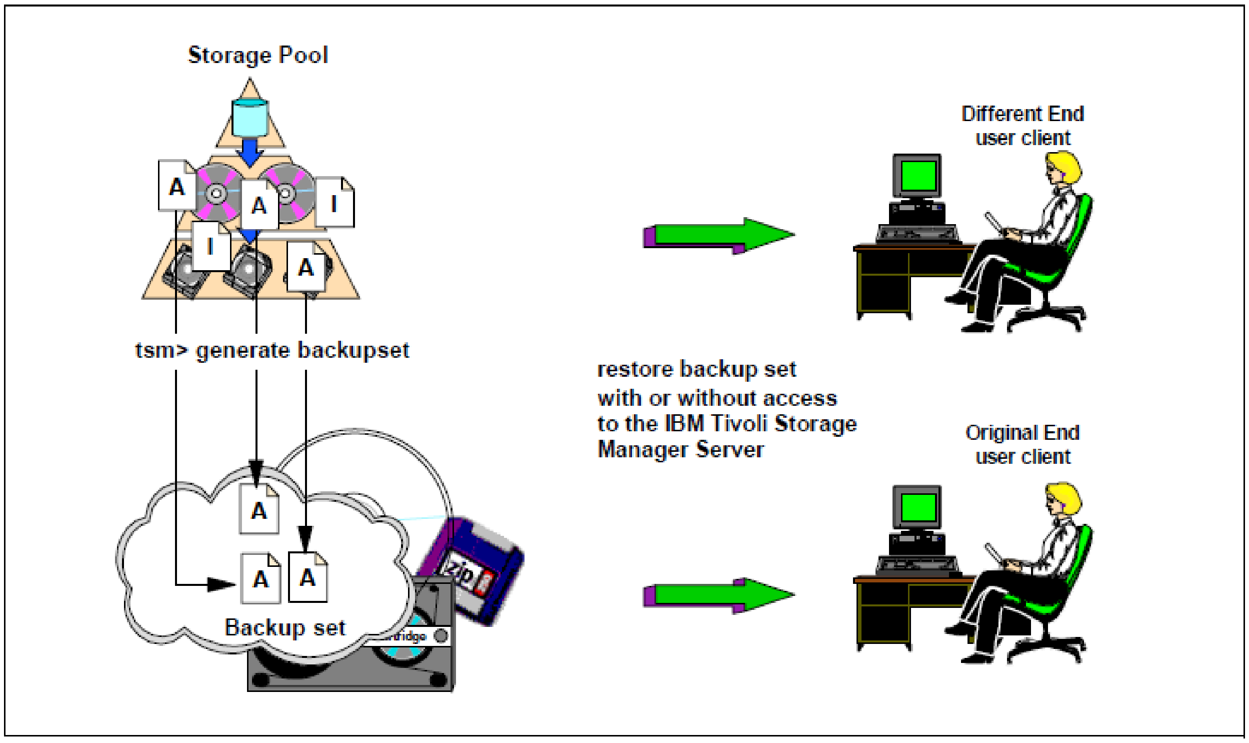
Dates

22/Jan - file still active
23/Jan - file deleted (enforce verdel=2)
24/Jan - server expire inventory
...
21/Feb - server expire inventory (delete 20/Jan file1 version)
23/Feb - server expire inventory (RETONLY control takeover)
24/Feb - RETONLY control
...
08/Mar - RETONLY last day control (23/Jan + 45 days)
09/Mar - server expire inventory
09/Mar - File1 Last inactive copy deleted



Sample Policy:

5.10 Backup set



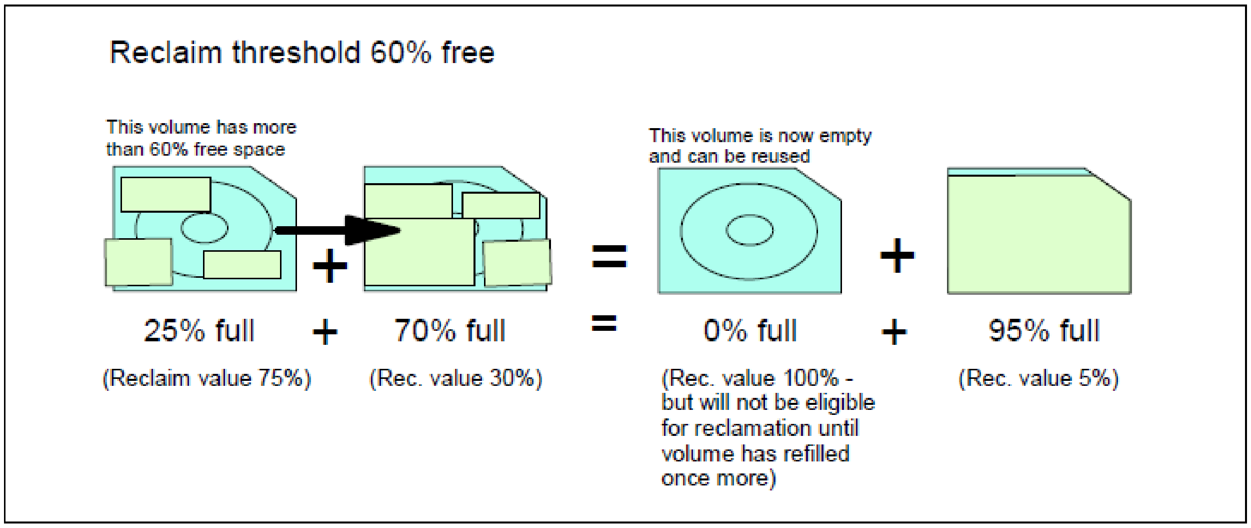
You can generate a copy of a client's most recent backup from the Tivoli Storage Manager server onto sequential media. This is accomplished using the **generate backupset** command, which copies all active file versions of the fileset from server storage onto the media. This copy of the backup, also called a backup set or a portable backup, is self-contained and can be used independent from Tivoli Storage Manager to restore the client's data from a locally attached device that can also read this media, such as a CD-ROM.

This technique provides the Tivoli Storage Manager client with a rapid recovery, which is achieved without needing access to either the Tivoli Storage Manager server or the network. You can also transfer the backup set from one server to another by generating the backup set on the source server, then transporting the backup set volume and defining it to the destination server, assuming that both servers have the same media type, as shown in the figure. The same node name is required to be registered on both servers.

6. TSM processes - expiration, reclamation, migration, collocation

6.1 Expiration

6.2 Reclamation


Reclaim threshold 60% free
This volume has more than 60% free space
This volume is now empty and can be reused
25% full
+
70% full
=
0% full
+
95% full
(Reclaim value 75%)
(Rec. value 30%)
(Rec. value 100% - but will not be eligible for reclamation until volume has refilled once more)
(Rec. value 5%)

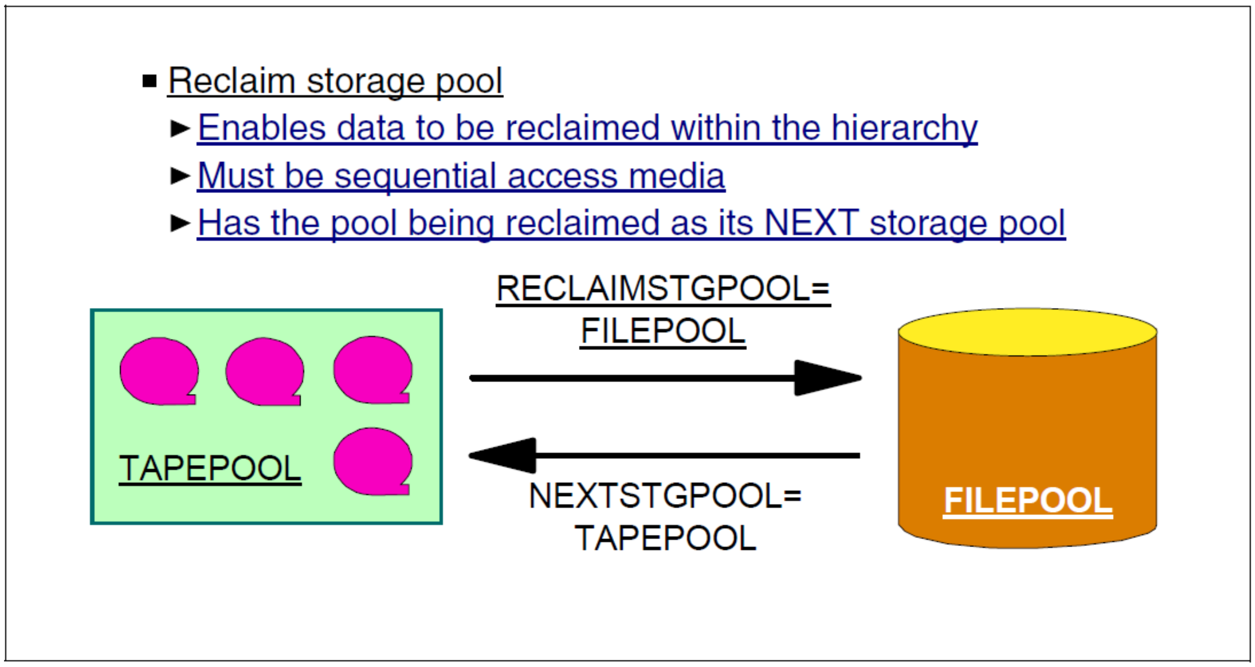
Reclamation is a server process that consolidates data and free space on tape (or optical) volumes in sequential storage pools. Over time, versions of backed-up files expire, or perhaps files are deleted from client file systems. It is common for tape volumes to contain files that will expire on different dates. When the expiration process occurs, expired and deleted files are marked as no longer required, and the tape volumes on which these files are stored now have empty space where the files physically resided.

Over time, as more and more files are expired from a tape volume, its active data spaces become fragmented by the increasing empty space. Fragmentation on tapes or optical disks causes increased read times due to the need to skip over the empty spaces. Similarly, restores take longer. The total number of volumes required is also higher than it needs to be. It is not possible to go back and rewrite new data in the empty spaces — sequential media can only be written from the beginning to the end. Once a sequential volume has been completely written one time, Tivoli Storage Manager will not write to it again, until it becomes totally empty.

The amount of empty space on a sequential volume is presented as the percentage of *reclaimable* space on the volume. Rather than wait until a volume becomes completely empty (which may never happen if data on the volume is active on the client), the empty space can be "reclaimed". Reclamation means moving the active data to another volume in the same storage pool. Only volumes that have a status of “scratch” or “filling” can be used to store the moved data (similarly, onsite volumes can only be reclaimed after they become “full”). When all the data is moved from the original tape to another, the original becomes empty, and is typically returned to a status of “scratch”. A scratch tape can be re-used for any function for which the server requires a volume.

The reclamation threshold is set on sequential storage pools via the **REClaim** parameter. The default value for a new storage pool is 60 percent, which can be over-ridden when the pool is created, or subsequently changed. When the reclamation value is reached for a particular sequential volume (that is, when the percentage of reclaimable data on the tape reaches the RECLAIM value), a reclamation process starts automatically for that volume. Having a reclamation process start automatically may not be desirable — the reclamation process is very device-intensive, and normally requires at least two available drives in the library (one to read, one to write). Other critical Tivoli Storage Manager operations (for example, a client restore) might be delayed or refused if all drives were engaged in reclamation. It is usually more convenient and efficient to schedule reclamation.

To prevent reclamation of volumes from happening automatically, set the value of **REClaim** on the storage pool to 100 percent. Then, at a suitable time, the reclamation process can be initiated using the **reclaim stgpool** command on Tivoli Storage Manager V5.3 or updating the storage pool to change the **REClaim** value. If reclamation is scheduled using the latter method, you must remember to reset the value after a suitable period. When setting the value of **REClaim** to start reclamation, we recommend that you specify a value of 50 percent or greater so that files stored on two volumes can be combined onto a single output volume.



- Reclaim storage pool
  - Enables data to be reclaimed within the hierarchy
  - Must be sequential access media
  - Has the pool being reclaimed as its NEXT storage pool

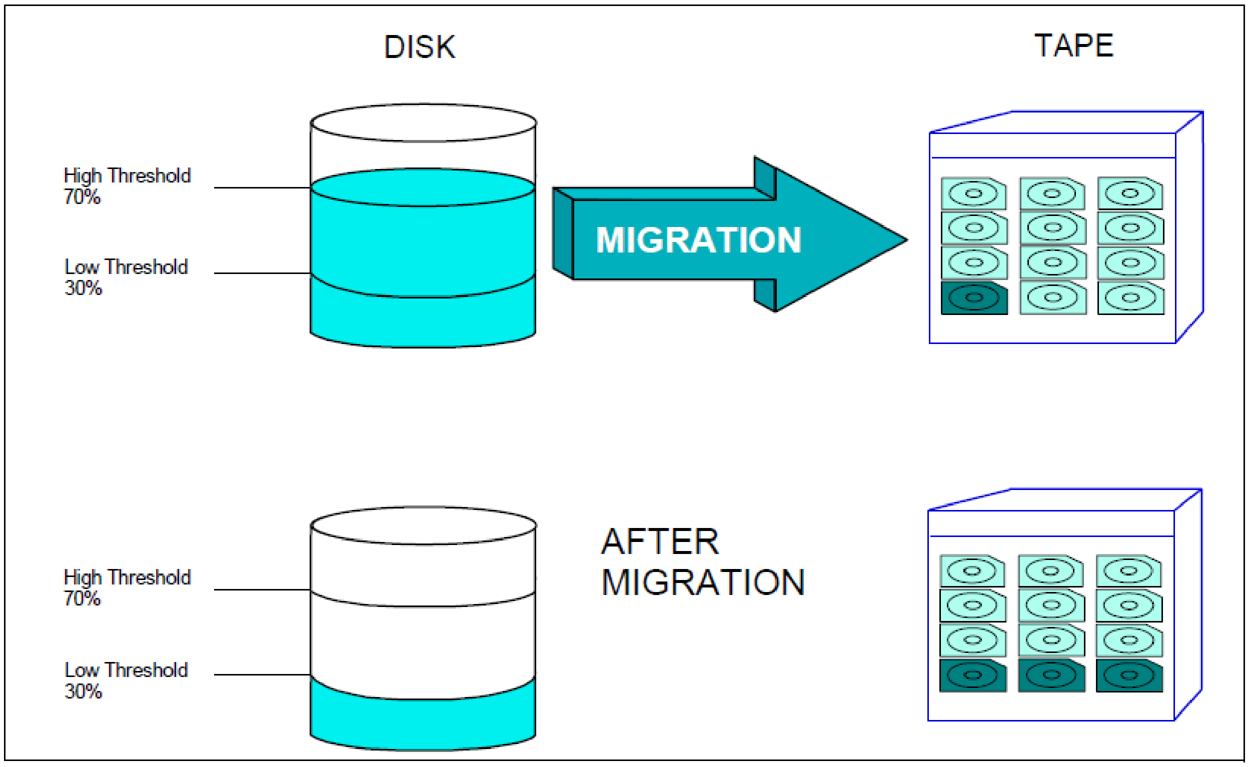
RECLAIMSTGPOOL=
FILEPOOL
TAPEPOOL
NEXTSTGPOOL=
TAPEPOOL
FILEPOOL

Reclamation requires two or more drives to work most efficiently. Nevertheless, reclamation can be performed on a single drive by specifying the **RECLAIMSTGpool** parameter. **RECLAIMSTGpool** allows another storage pool to be used as the holding area for the sequential volume being consolidated. The storage pool specified as the reclaim storage pool must be a primary sequential storage pool.

When defining the reclaim storage pool, you must also define the **NEXTstgpool** parameter pointing back to the pool being reclaimed. Thus the reclaimed data is migrated back to the original storage pool.

6.3 Migration

Migration helps control the amount of free space within a storage pool, and can be used to move data to its final (or more permanent) location. High and low migration thresholds can be defined for each primary storage pool in the hierarchy. The thresholds tell Tivoli Storage Manager when to move data from one storage pool to another. The default values for a newly created storage pool are 90 percent (**HIghmig**) and 70 percent (**LOwmig**). Copy storage pools do not have migration thresholds.



At some point in time, you may wish to force migration from a disk pool to a tape pool. Rather than just wait until a disk pool reaches its high migration threshold, a migration process can be initiated at any time. One reason for manually initiating a migration process is that migration during client backup may reduce performance because of the additional I/O. If there are a large number of clients backing up simultaneously, it is a best practice to give them an empty disk pool large enough to contain a typical incremental backup, so that migration is not triggered.

There are two ways to initiate a migration:

- Update the storage pool, modifying the values of **HIghmig**, **LOwmig**, or both.
- Use the **migrate stgpool** command after TSM 5.3

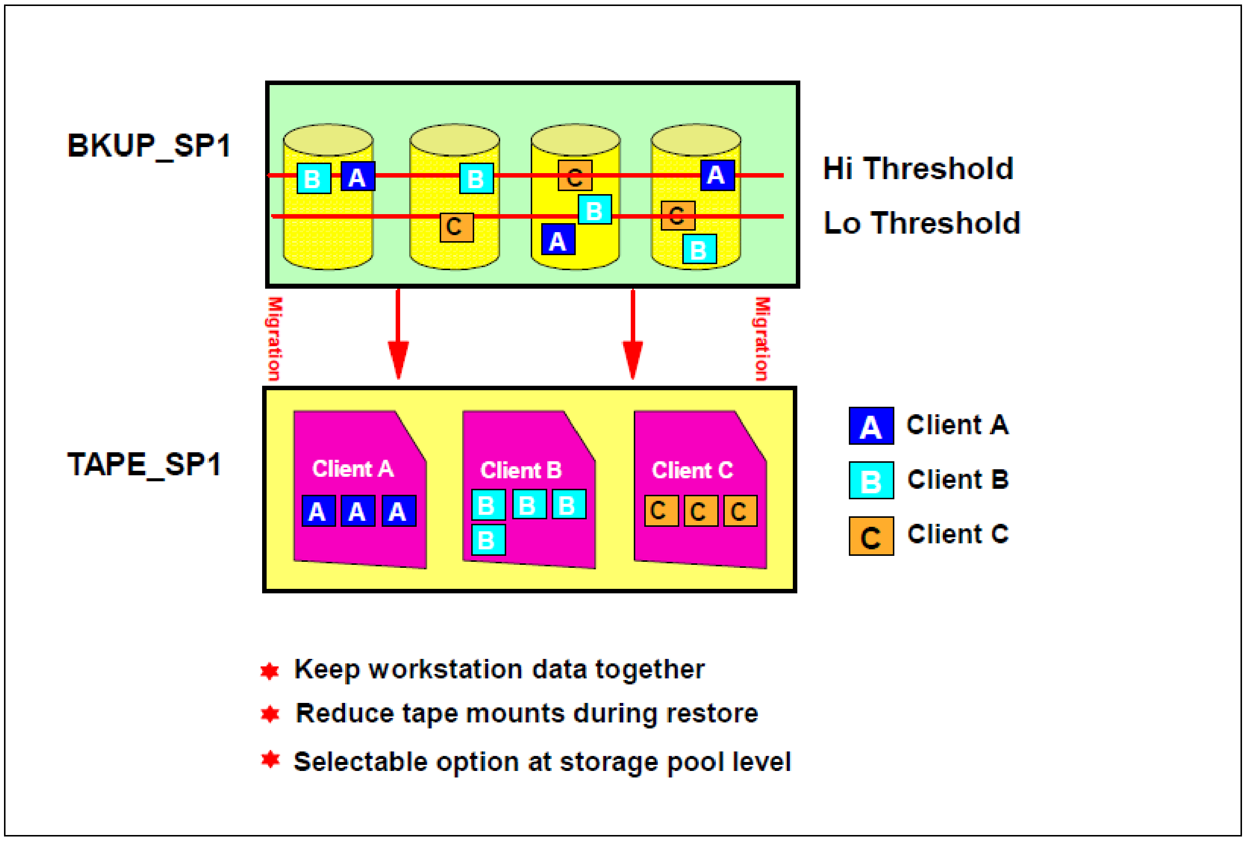
Storage pool attribute Maxsize

The storage pool attribute **MAXSIze**, specifies the maximum size of a file that can be stored in the storage pool. If compression is used, the uncompressed size of the file is used for comparison. If a file is too large for a storage pool, it will go straight to the next storage pool defined in the hierarchy. The next storage pool must have volumes (and tape drives if a tape storage pool) available, otherwise the backup of the file will fail.

Limiting the file size for a storage pool is another way to improve performance — the server does not waste time writing large files to the first (usually disk) pool, only to trigger an immediate migration when the storage pool is filled. LTO drives have good streaming performance; setting a **MAXSIze** value on a disk pool can make use of that performance.

MAXSIze is not required. If you do not assign a **MAXSIze**, the default is NOLIMIT, meaning that the server will attempt to store any sized file in that storage pool. If a file entering the storage pool causes it to exceed its high threshold, migration will automatically occur to move data to the next storage pool.

6.4 Collocation



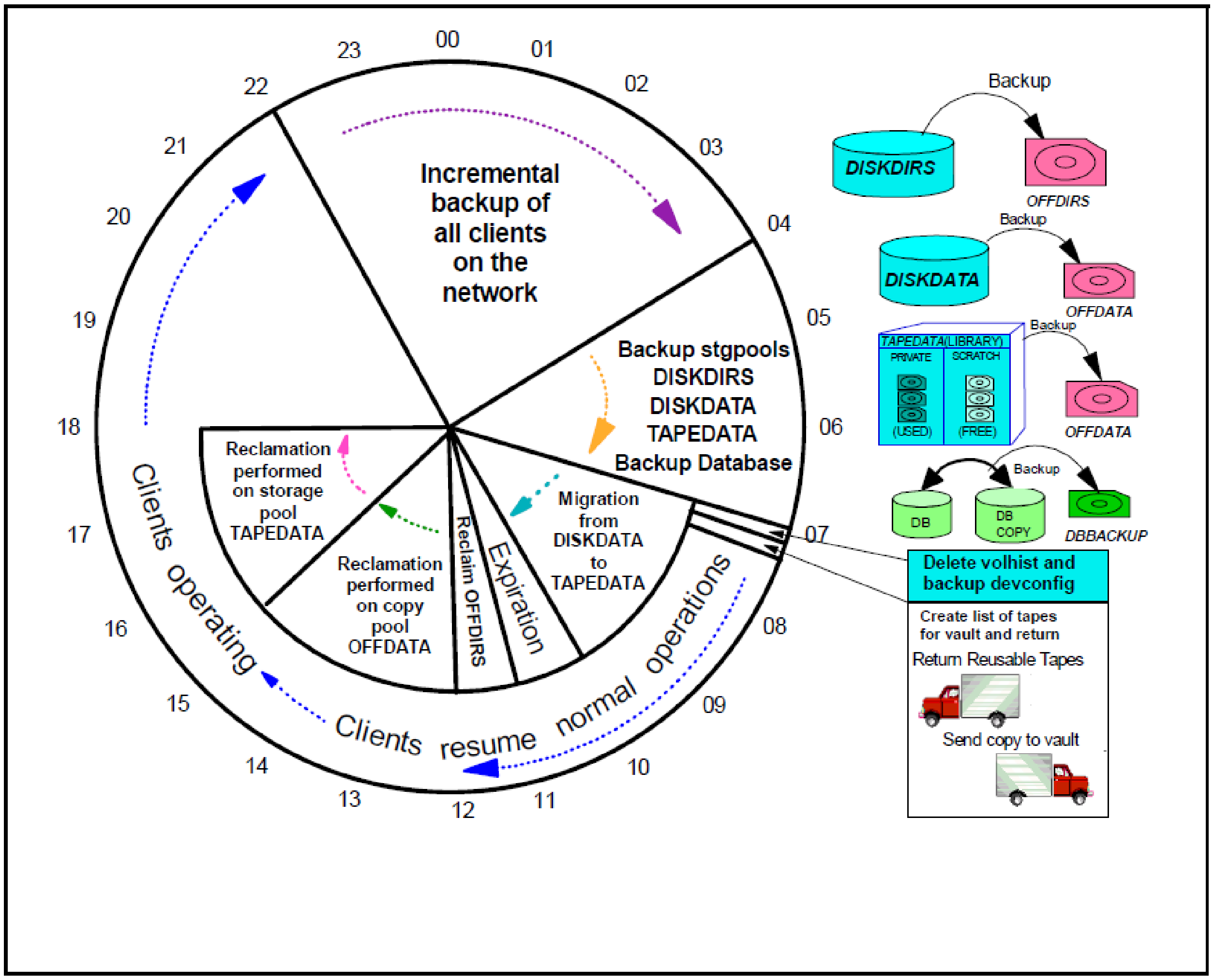
When collocation is enabled for a storage pool, the server attempts to keep all files belonging to a client node, a client’s filespace, or a group of nodes, on the smallest number of sequential volumes. Using collocation reduces the number of volume mount operations required when restoring or retrieving many files from the storage pool, and also improves performance for these operations.

A number of different options are available for the **COLlocate** parameter to control the granularity of collocation:

- The default for Tivoli Storage Manager V5.3 is to collocate by group. Data for a group of nodes is placed on as few volumes as possible. When **COLlocate** is set to “Group”, you must also create a collocation group (**define collocgroup**) and give it members (**define collocmember**). If you do not define and collocation groups or members, the default behavior is to collocate by node (see below).
- To collocate data for each node in the same storage pool, set **COLlocate** to “Node”. The server will put data for each node on as few volumes as possible. To maintain compatibility with older versions of Tivoli Storage Manager, **COLlocate** can be set to “Yes”, which has the same behavior as “Node”.
- The finest granularity for the **COLlocate** parameter is “Filespace”. If collocation by filespace is defined, the server attempts to put data for one filespace of one node on one volume. If a node has multiple filespaces, the server attempts to place data for each filespace on different sequential volumes in the storage pool.

When collocation is disabled (that is **COLlocate** is set to “No”), the server attempts to use all available space on each volume before selecting a new volume. While this method provides better utilization of individual volumes, user files can become scattered across many volumes. Complete restoration of a client may require many volume mounts.

7. Daily Administration cycle



00
01
02
03
04
05
06
07
08
09
10
11
12
13
14
15
16
17
18
19
20
21
22
23
Incremental backup of all clients on the network
Backup stgpools DISKDIRS DISKDATA TAPEDATA Backup Database
Migration from DISKDATA to TAPEDATA
Expiration
Reclaim OFFDIRS
Reclamation performed on copy pool OFFDATA
Reclamation performed on storage pool TAPEDATA
Clients operating
Clients resume normal operations
Backup
DISKDIRS
OFFDIRS
Backup
DISKDATA
OFFDATA
Backup
TAPEDATA(LIBRARY)
PRIVATE
SCRATCH
(USED)
(FREE)
OFFDATA
Backup
DB
DB COPY
DBBACKUP
Delete volhist and backup devconfig
Create list of tapes for vault and return
Return Reusable Tapes
Send copy to vault

8. Q&A