# Optimizing for Recovery

Dave Cannon
IBM Tivoli Storage Management Development
Oxford University TSM Symposium 2003

Tivoli. software

e business software





## Setting Expectations

- This session describes TSM functions and methods for optimizing recovery of client data
- Most of the techniques described in this presentation must be implemented before recovery is needed, so plan ahead
- This is not a performance tuning session
- We won't cover disaster recovery or bare-metal restore
- There is not a magic, one-size-fits-all solution for optimal recovery

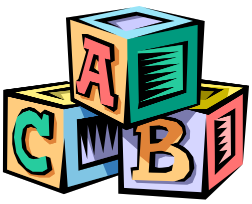
# Topics

- General principles
- Exploiting disk storage
- Sequential-access storage
- Special situations
- Potential future enhancements






# General Principles



# Tradeoffs

**Restore performance**

**Other considerations**

- Cost of storage
  - Capacity requirements
  - Device characteristics
- Backup constraints
  - Backup window
  - Network bandwidth
  - Impact to client
- Efficient data transfer within storage hierarchy
- Policy goals
  - Number of versions
  - Retention time

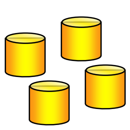





# Prioritizing Data

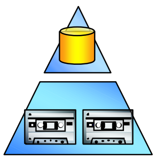
- Since there are tradeoffs (costs) associated with fast recovery, prioritize data to optimize recovery at least for most critical data
- Consider using different storage pool hierarchies and methods for different nodes, data types, or objects
  - Disk usually allows fastest recovery for small-medium files (<100MB)
  - High-performance tape may be optimal for large objects (databases or file system images)

Critical Files



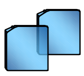
Disk only

Less-critical Files



Disk and tape

Databases



High-performance tape

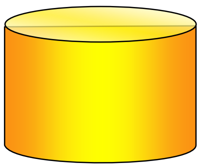
# Prioritizing Recovery Scenarios

- Single file or multiple files stored together
  - Tracking of objects by TSM database optimizes restore
- Multiple files (such as a directory) backed up at different times
  - Fragmentation of data in sequential-access storage pools may make this challenging
  - Presentation will emphasize approaches to optimize for this scenario
- Entire file system
  - Effects of fragmentation can be mitigated with image backup/restore
- Databases or other applications with large objects
  - Restore of large applications is typically fast
  - To optimize, see publications for your application protection product






# Exploiting Disk Storage



# Disk Storage Basics

- Disk storage can provide optimal restore for small/medium files
  - No delays for mounting of volumes
  - Fast positioning to data within volume
  - Simultaneous volume access from multiple sessions
- Random-access disk storage pools defined with device class of DISK
- Sequential-access disk storage pools defined using device class with DEVTYPE=FILE



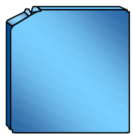



# Random vs. Sequential Disk

| | **Random-access** | **Sequential-access** |
|---|---|---|
| Space allocation/tracking | Random 4KB blocks | Sequential within volume |
| TSM caching | Optional (backup overhead) | Not supported |
| Space recovery (no cache) | When file is deleted/moved | When volume is reclaimed |
| Recovery of cache space | When space is needed | Not applicable |
| Aggregate reconstruction | Not supported | During volume reclamation |
| Concurrent volume access | Yes | Not supported |
| Multi-session client restore | One session for all volumes | One session per volume |
| Target for LAN-free backup | Not supported | Yes, via SANergy |
| Can be used for copy pools | No | Yes |
| Migration/stg pool backup | By node and file space | By volume |
| Parallel migration processes | Yes | No |
| Migration to collocated pool | Very efficient | Moderately efficient |
| Pools can span file systems | Yes | No |
| Database regression | Must audit all volumes | Reuse delay avoids audit |

# Sequential-access Storage

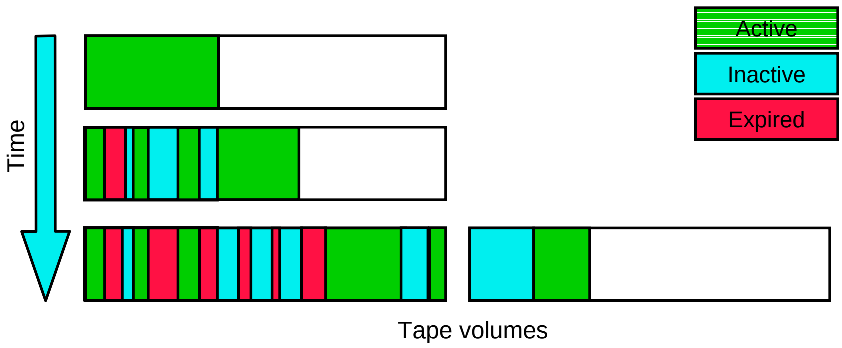
- Optimizing restore from sequential-access storage such as tape can be a challenge
  - Mounting of storage volumes
  - Positioning to data to be restored
  - Fragmentation
- Techniques described in this section can be applied to sequential disk storage, but are most beneficial for other sequential storage media








# Fragmentation of Data in Sequential Storage Pools




- Data required for a restore can be fragmented
  - Within a volume
  - Across multiple volumes
- Fragmentation increases restore time

# Reducing Fragmentation of Sequential Media

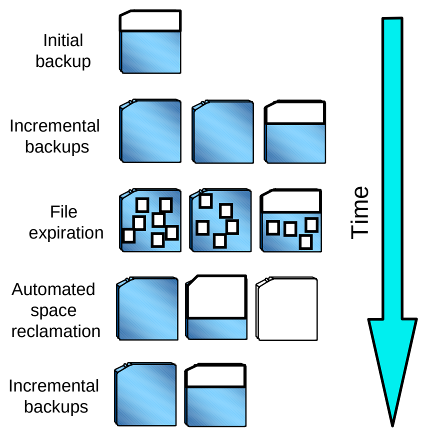
- Expiration and reclamation
- Collocation
- Limit number of versions
- File-system image backup and restore
- Full backups




IBM Software Group | Tivoli software

# Expiration and Reclamation

- Expiration deletes files per policy
- Reclamation consolidates data in sequential storage pools leading to
  - Reduced storage requirement
  - Improved restore performance
- Perform expiration and reclamation regularly to reduce fragmentation



## Collocation

- Minimizes number of sequential-access volumes used for a client
  - Node
  - File space
  - Directory tree (using virtual mount point and collocation by file space)
- Improves restore performance by reducing volume mounts
- Can minimize volume contention during concurrent restores of data for multiple nodes or file spaces
- Reduces benefits of multi-session restore operations
- Increases number of volumes required
- Can increase processing for data transfer to collocated media






## Limit Number of Versions

- Older versions expire earlier allowing reclamation to consolidate remaining data
- Restore operations locate data faster because there are fewer versions to skip
- Tailor policies for critical data to retain necessary versions while also considering impact to restore

# File-system Image Backup and Restore

- Optimizes restore of entire file system by eliminating fragmentation of data to be restored
- Can be combined with incremental backups to allow restore of file system to arbitrary point in time
  - Incremental backups may be fragmented which extends restore time
  - Perform image backups often enough to minimize impact of fragmentation (before more than 20% of data has changed)
- If available, snapshot technology minimizes impact to client applications






# Full Backups

- Periodic full selective backup collocates data, which can greatly improve restore time
- Consider using a different node for the full backup
  - Avoids versioning files that have not changed
  - To restore to point in time using two nodes
    - Restore from last full backup previous to required point in time
    - Restore files that changed since full backup
- Could be used for critical file spaces or directory trees

## Minimizing the Effects of Fragmentation

- Move Nodedata command
- Multi-session restore
- Parallel restore operations
- Separate node to optimize restore of active data

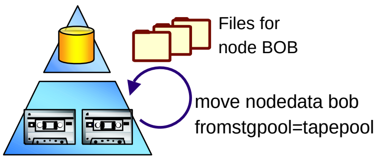




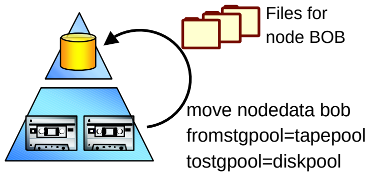

## Move Nodedata Command

- Moves files for specified node, file space, and data type residing in a specified sequential-access storage pool
- Prepares for rapid client restore by
  - Consolidating node data within a sequential-access pool
  - Moving data to disk for fast access
- Reconstruction option removes unused space within aggregates
- Requires lead time before restore

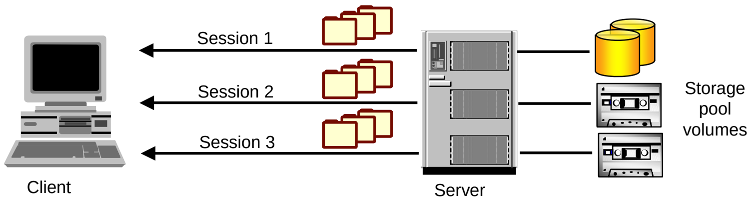
Consolidation




Movement to Disk



# Multi-session Restore

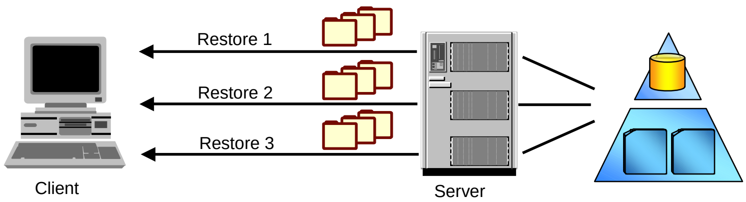



- Restore uses multiple client-server sessions for a single file space
- Provides improved throughput
- Limited by
  - Number of sequential-access volumes with data to be restored plus one session for random-access disk
  - Mount points
  - Client's resourceutilization option
- Most effective when data is spread over multiple sequential volumes
- Works only for no-query restore (unrestricted wildcard)



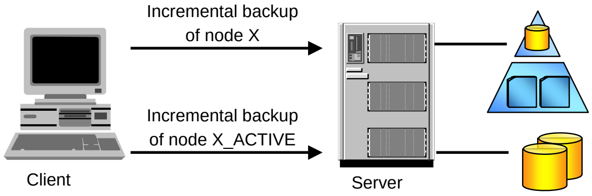

IBM Software Group | Tivoli software

# Parallel Restore Operations




- Divide data to be restored and use separate restore operation for each data subset
  - File spaces
  - Directory trees
- May cause volume contention unless
  - Data can be restored from random-access disk OR
  - Data required for each restore is located on separate sequential volume (e.g., collocation by file space)

# Separate Node to Optimize Restore of Active Data




**Node X**

- For restore of inactive versions
- Progressive incremental backup
- Data stored in disk-tape hierarchy
- Policy allows active/inactive versions as required

**Node X_ACTIVE**

- For optimal restore of active versions
- Progressive incremental backup
- Data stored in FILE storage pool to allow aggregate reconstruction
- Policy allows only active version (or possibly active plus small number of inactive versions)

IBM Software Group | Tivoli software

# Special Situations

## Using NDMP to Protect and Recover NAS Data

- High-performance, scalable backup of NAS file servers using Network Data Management Protocol (NDMP)
  - Full file-system image
  - Differential file-system image (files that have changed since last full backup)
- NDMP-controlled restore
  - Full file-system image
  - Full file-system image plus one differential image
  - Selected files and/or directories using the image table of contents
- Data flow for backup/restore is LAN-free and outboard of TSM client/server






## Classic vs. No-query Restore

- Classic restore processing
  1. Client queries server for information about files to be restored
  2. Server sends file information to client
  3. Client sorts files by storage location
  4. Client sends restore request for each file in optimal restore order
  5. Server sends each specified file to client
- No-query restore (NQR) processing
  1. Client sends restore specification to server
  2. Server sends matching files to client

## Classic vs. No-query Restore (continued)

- Classic vs. NQR tradeoffs
  - NQR reduces client-server interaction and client memory requirement
  - NQR allows multi-session restores, but classic restore does not
  - NQR is usually faster for restoring entire file systems or large directories
  - Classic restore may be faster for restoring directories if data is highly fragmented across sequential volumes with only a small amount of data on each volume
- NQR is automatically used if both of the following apply
  - The file specification is an unrestricted wildcard (/home/mydata/*)
  - None of the following are used: inactive, latest, pick, fromdate, todate
- To force classic restore, use ?* in the file specification (home/mydata/?*)






## LTO Tape

- Apply techniques as for other sequential media
- IC33920
  - TSM server uses read rather than locate to improve performance for positioning to fragmented data
  - Not available on Windows
- Recent Windows LTO device driver (5.0.6.4) can improve performance significantly for Gen 1 drives
- Use LTO Gen 2 rather than Gen 1
  - Faster streaming
  - Improved locate performance
  - Improved I/O for Windows
  - Note that larger capacity of Gen 2 media may reduce potential benefit of multi-session restore

# Potential Future Enhancements

- Collocation groups
- Disk storage enhancements
- Collocation of active data
- Restore of individual files from file system image
- Increased tape block size for Windows