Point-in-Time Copy:
Yesterday, Today and Tomorrow

Alain Azagury
Michael E. Factor
Julian Satran
IBM Research Lab in Haifa
William Micka
IBM Storage Systems Group

10th NASA Goddard Conference
on Mass Storage Systems and Technologies

April 15 - 18, 2002

Agenda

- Point-in-time copy definition
- Motivation
- Classes of implementation
- Survey of current solutions
- File system point-in-time copy
- IBM’s ESS FlashCopy
- Future trends

Definition

"A **fully usable copy** of a defined collection of data that contains an image of the data as it appeared at a **single point-in-time**. The copy is considered to have logically occurred at that point-in-time, but implementations may perform part or all of the copy at other times […] as long as the result is a consistent copy of the data as it appeared at that point-in-time. Implementations may restrict point-in-time copies to be **read-only** or may permit subsequent writes to the copy."

The Storage Networking Industry Association (SNIA)

Why Point-in-Time Copies?

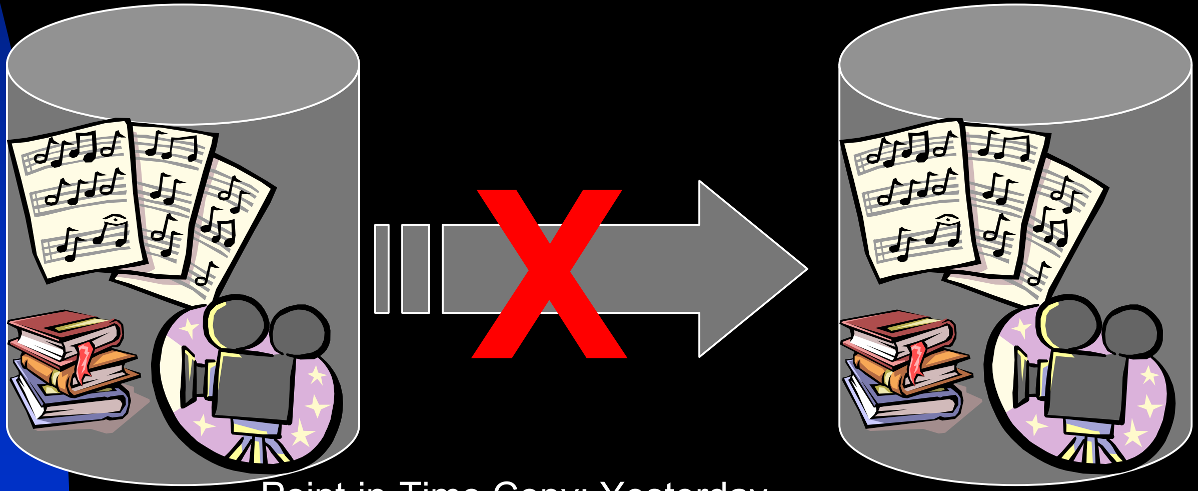
- Non-disruptive backup
  - Probably the most common reason
- Checkpointing
  - Safeguard against failures
- Data mining
  - Scan a consistent copy of the data without impacting production application
- Testing
  - E.g. Y2K

Classes of Implementations

- Split mirror
- Changed block
- Concurrent

Split Mirror

- A mirror of the data is constructed prior to the point-in-time copy
- The point-in-time copy is made by “splitting” the mirror



Split Mirror Characteristics

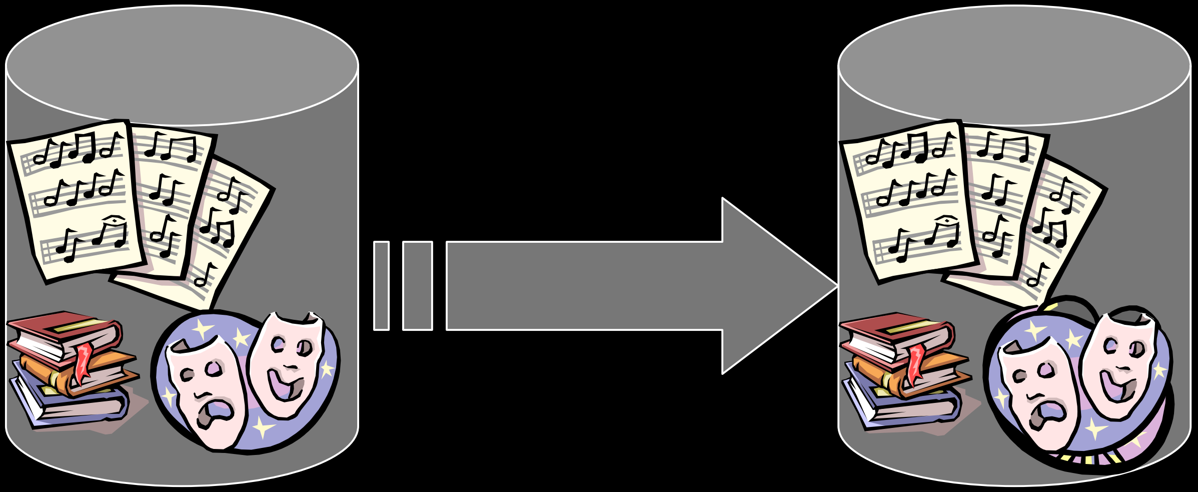
- Advantages
  - Point-in-time copy executes very quickly
  - Physical copy provides additional protection
- Disadvantages
  - Requires advanced planning
  - Space for copy needs to be pre-allocated
  - Performance penalty of mirroring

Split Mirror Variant

- Resynchronizing split mirrored copies

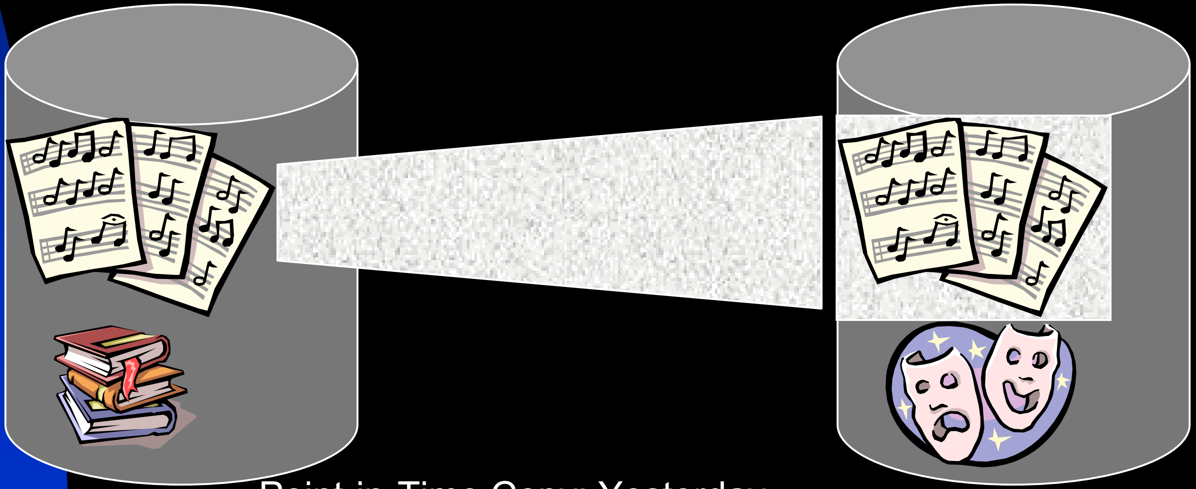
Split Mirror Variant

- Resynchronizing split mirrored copies



Changed Block

- Shares the physical copy of the data until the data is written
- Requires setting up a "table" to keep track of modified records
  - Fits naturally in log-structured arrays



Changed Block Characteristics

- Advantages
  - No advanced set up is required prior to executing a point-in-time copy
  - Amount of space required is a function only of the amount of data modified
- Disadvantages
  - Requires time to set up the table
  - No physically separated copy

Concurrent

- Similar to “changed block”
- However, always physically copies the data (in the background)

Additional Potential Limitations

- Some implementations put additional limitations on the copy, e.g.,
  - Read-only
  - Only sequential reads
  - Resilience to failures
  - …

Block vs. File

- Block copy advantages
  - Reduces load on the server and on the storage network
- File advantages
  - Finer granularity control

Split Mirrored Implementations

- Examples
  - EMC's TimeFinder
  - Hitachi's ShadowImage
- EMC's Timefinder
  - Originally a split-mirror implementation
  - Supports incremental resynchronization of copies
  - Latest version supports "changed block" implementation for faster set-up time

Log Structured Changed Block Solutions

- Examples
  - IBM's RAMAC Virtual Array (RVA)
  - StorageTek's Shared Virtual Array
- Volume implementation
  - Represented by a set of tables that eventually point to the set of tracks that comprise the volume
- Point-in-time copy setup implementation details
  1) Decrease the reference count of the target tracks
  2) Copy the "track" table from the source to the target
  3) Increase the reference count of the source volume tracks

File Level Implementations

- Most implementations leverage the file system "inode" implementation
  - Snapshot points initially to same data blocks as the source
  - Uses copy-on-write technique to guarantee two copies semantics
- Network Appliance Inc.
  - Combines "snapshot" with "Snapmirror/SnapRestore" utility
  - Modified blocks are mirrored in a remote location
- Caveat: snapshots are "read-only"
  - Metadata is also read-only!
  - Access control of the replica cannot be changed!

IBM's ESS FlashCopy

- A *concurrent* point-in-time copy
  - Utilizes *copy-on-write* bitmap
- Provides instant availability for read and write data on both the source and target
- For zSeries, can specify that only a portion of the volume be copied
  - *Sparse volume*

IBM's ESS FlashCopy Performance

- Time required for the invocation of the copy

# of FlashCopy Volumes	Dss small VTOC	Dss large VTOC	TSO invoked
1	6 sec	8 sec	1.2 sec
256	48 sec	66 sec	18 sec

- Impact on application response time
  - Less than 3% impact on I/O rate for 256 volumes running a cache standard workload, no background copy
  - Less than 7% with background copy

Future Trends

- Improving Today’s Point-in-time Copy
  - Towards instantaneous point-in-time copies
    - Efficient management of the cache
    - Efficient data structures

Future Trends *(cont.)*

- Point-in-time copy and Object Based Storage
  - Relegates space management to the storage subsystem
  - File-level point-in-time copy can be made without moving (meta)data from the storage controller to the file server
  - (Incremental) point-in-time copy can be made with minimal space (and time) overhead and encompasses any set of objects (not necessarily a volume or a large portion of a volume)