z/OS Data De-De-De-De-duplication

Speaker Name Dr. H. Pat Artis
Speaker Company Performance Associates, Inc. (PA1)

Speaker Name Victor T. Peltz
Speaker Company IBM Corporation (IBM)

Date of Presentation Tuesday, August 3, 2010
Session Number 8021

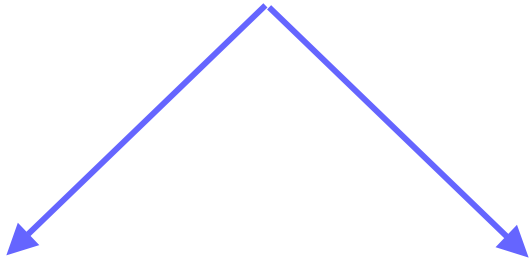
Topics

- Why de-duplication?
- Backup drivers
- How data de-duplication works
- An audacious hypothesis
- z/OS implementations
- Fixed and variable length segments
- A hypothetical model
- De-duplication workload characterization
- Performance metrics
- Critical decision factors
- Questions

Why De-duplication?

- Better use of disk capacity
  - Up to 25x reduction in physical disk space required
  - Reduces VTL Sprawl
- “Green” solution
- Lower Total Cost of Ownership
  - Environmental costs
  - Equipment purchase deferral or avoidance

Backup “Drivers”

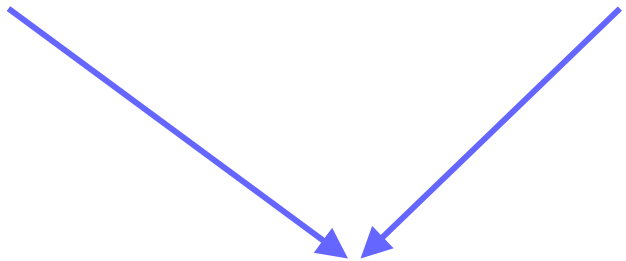


Improve performance

- Faster drives
- Multiplexing
- Networks
- VTL

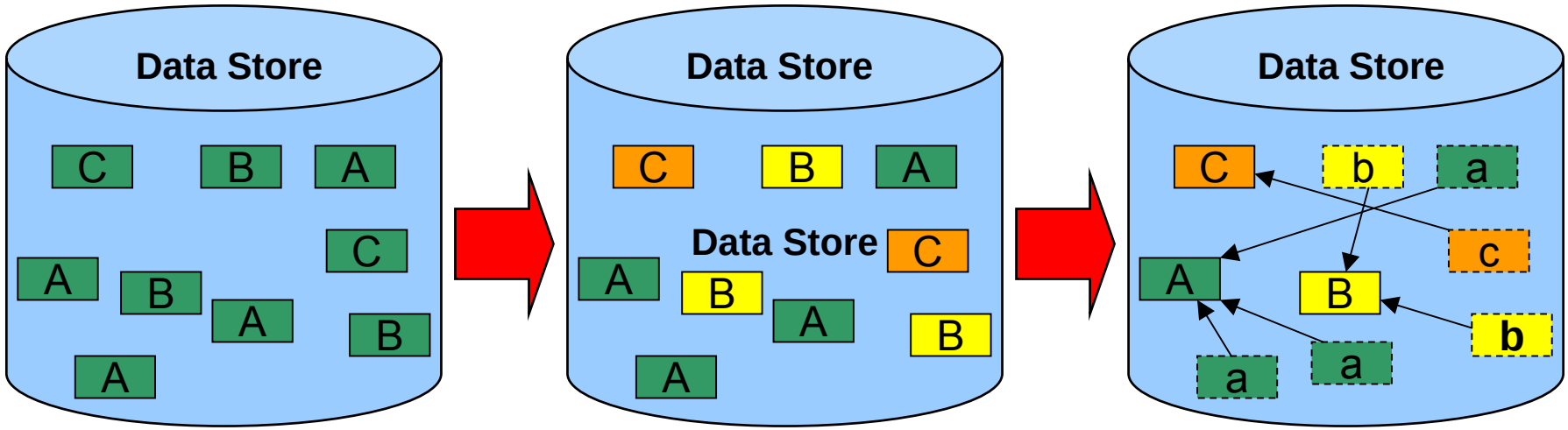
Increase capacity

- Incremental backups
- Compression
- Denser tape media
- Generation 1 De-Dupe



Most de-duplication solutions provide capacity improvements, while forfeiting performance

How Data De-duplication (DDD) Works



1. Data chunks are evaluated to determine a unique signature for each

2. Signature values are compared to identify all duplicates

3. Duplicate data chunks are replaced with pointers to a single stored chunk, saving storage space

De-duplication Design Considerations

- Source-side vs. target-side
- In-band vs. out-of-band
- Method used for data chunking
- How redundant chunks are identified
- Avoiding false matches
- How redundant chunks are eliminated and tracked
- De-duplication ratios
- De-duplication design considerations

Where De-duplication is Performed

Approach	Advantages	Disadvantages
Source-side (client-side) De-duplication performed at the data source (e.g., by a backup client), before transfer to target location	• De-duplication before transmission conserves network bandwidth • Awareness of data usage and format may allow more effective data reduction	• De-duplication consumes CPU cycles on the file/ application server • Requires software deployment at source (and possibly target) endpoints • Depending on design, may be subject to security attack via spoofing
Target-side (server-side) De-duplication performed at the target (e.g., by backup software or storage appliance)	•No deployment of client software at endpoints •Possible use of direct comparison to confirm duplicates	• De-duplication consumes CPU cycles on the target server or storage device • Data may be discarded after being transmitted to the target

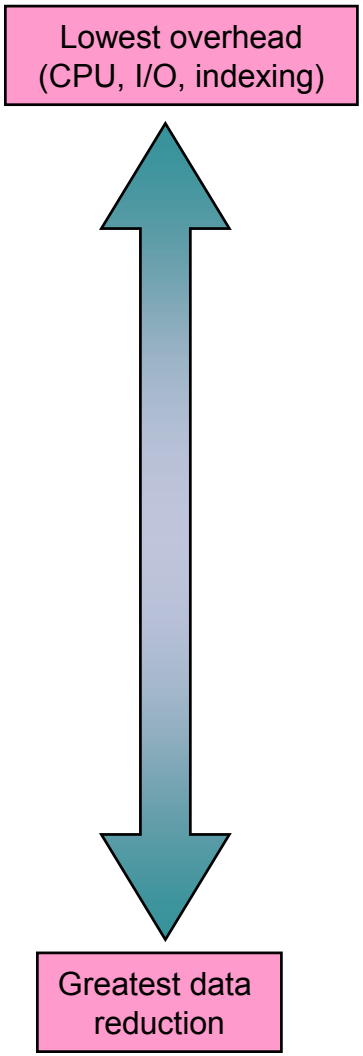
Note: Source-side and target-side de-duplication are not mutually exclusive

When De-duplication is Performed

Approach	Advantages	Disadvantages
In-band De-duplication performed during data processing on the source or target	•Immediate data reduction, minimizing disk storage requirement •No post-processing	•May be bottleneck for data ingestion (e.g., longer backup times) •Only one de-duplication process for each I/O stream •No de-duplication of legacy data on the target server
Out-of-band De-duplication performed after data ingestion at the target	•No impact to data ingestion •Potential for de-duplication of legacy data •Possibility for parallel data de-duplication processing	•Data must be processed twice (during ingestion and subsequent de-duplication) •Storage needed to retain data until de-duplication occurs

Note: In-band and out-of-band de-duplication are not mutually exclusive

Data Chunking Methods



Whole file chunking

- Each file is treated as a single chunk
- No detection of duplicate data at subfile level

Fixed-size chunking

- Chunk boundaries occur at fixed intervals, irrespective of data content
- Method is unable to detect duplicate data if there is an offset difference
  - Because redundant data has shifted due to insertion/deletion
  - Because redundant data is embedded within another file or contained in a composite structure

Variable-size chunking

- Rolling hash algorithm is used to determine chunk boundaries to achieve an expected average chunk size
- Can detect redundant data, irrespective of offset differences
- Often referred to as fingerprinting (e.g., Rabin fingerprinting)

Format-aware chunking

- In setting chunk boundaries, algorithm considers data format/structure
- Examples: awareness of backup stream formatting; awareness of PowerPoint slide boundaries; awareness of file boundaries within a composite

Identification of Redundant Chunks

- Unique identifier is determined for each chunk
- Identifiers are typically calculated using a hash function that outputs a digest based on the data in each chunk
  - MD5
  - SHA
- For each chunk, the identifier is compared against an index of identifiers to determine whether that chunk is already in the data store
- Selection of hash function involves tradeoffs between
  - Processing time to compute hash values
  - Index space required to store hash values
  - Risk of false matches

Hash Functions

Hash functions take a message of arbitrary length as input and output a fixed length digest of L bits. They are published algorithms, normally standardized as RFC.

Name	Output size L (bits)	Performance (cycles/byte) Intel Xeon: C / assembly*	Collision chance 50% (or greater) when these many chunks (or more) are generated **	Chance of one collision in a 40 PB archive*** (using 4KB / chunk)	Year of the standard
MD5	128	9.4 / 3.7	$2^{64} \approx 10^{20}$	$0.5*10^{-20}$	1992
SHA-1	160	25 / 8.3	$2^{80} \approx 10^{24}$	$0.5*10^{-28}$	1995
SHA-256	256	39 / 20.6	$2^{128} \approx 10^{40}$	$0.5*10^{-60}$	2002
SHA-512	512	135 / 40.2	$2^{256} \approx 10^{80}$	$0.5*10^{-140}$	2002
Whirlpool	512	112 / 36.5	$2^{256} \approx 10^{80}$	$0.5*10^{-140}$	2003

The chances of a hash collision is about the same as the chance of one person getting hit by lightning ….twice

Probability of collision is extremely low and can be reduced at the expense of performance by using a hash function that produces longer digest

False Matches

- Possibility exists that two different data chunks could hash to the same identifier (such an event is called a collision)
- Should a collision occur, the chunks could be falsely matched and data loss could result
- Collision probability can be calculated from the possible number of unique identifiers and the number of chunks in the data store
  - Longer digest ➔ More unique identifiers ➔ Lower probability of collisions
  - More chunks ➔ Higher probability of collisions
- Approaches to avoiding data loss due to collisions
  - Use a hash function that produces a long digest to increase the possible number of unique identifiers
  - Combine values from multiple hash functions
  - Combine hash value with other information about the chunk
  - Perform byte-wise comparison of chunks in the data store to confirm matches

Elimination of Redundant Chunks

- For each redundant chunk, the index is updated to reference the matching chunk
- Metadata is stored for the object indicating how to reconstruct the object from chunks, some of which may be shared with other objects
- Any space occupied by the redundant chunks can be de-allocated and reused

De-duplication Ratios

- Used to indicate compression achieved by de-duplication
- If de-duplication reduces 500 TB of data to 100 TB, ratio is 5:1
- De-duplication vendors claim ratios in the range 20:1 to 400:1
- Ratios reflect design tradeoffs involving performance and compression
- Actual compression ratios will be highly dependent on other variables
  - Data from each source: redundancy, change rate, retention
  - Number of data sources and redundancy of data among those sources
  - Backup methodology: incremental forever, full+incremental, full+differential
  - Whether data encryption occurs prior to de-duplication

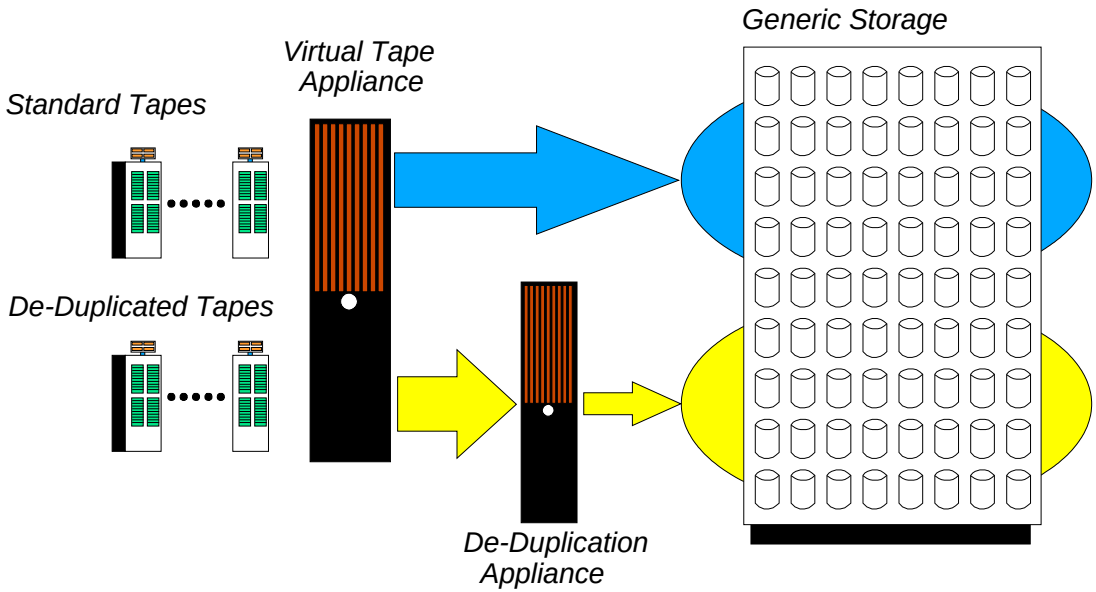
De-duplication Design Considerations

- Redundant Data Elimination
  - The grain of redundancy, 8KB, 1 MB or …
  - The more granular, the better the de-duplication ratios
  - The more granular, the higher the processing requirements
- Performance
  - Performance battles with Capacity
  - Performance is challenged/curtailed by disk I/O
- Capacity
  - Backup to disk has to cope with 100's of TBytes
  - All designs can grow in capacity… but many do so at the cost of performance

An Audacious Hypothesis

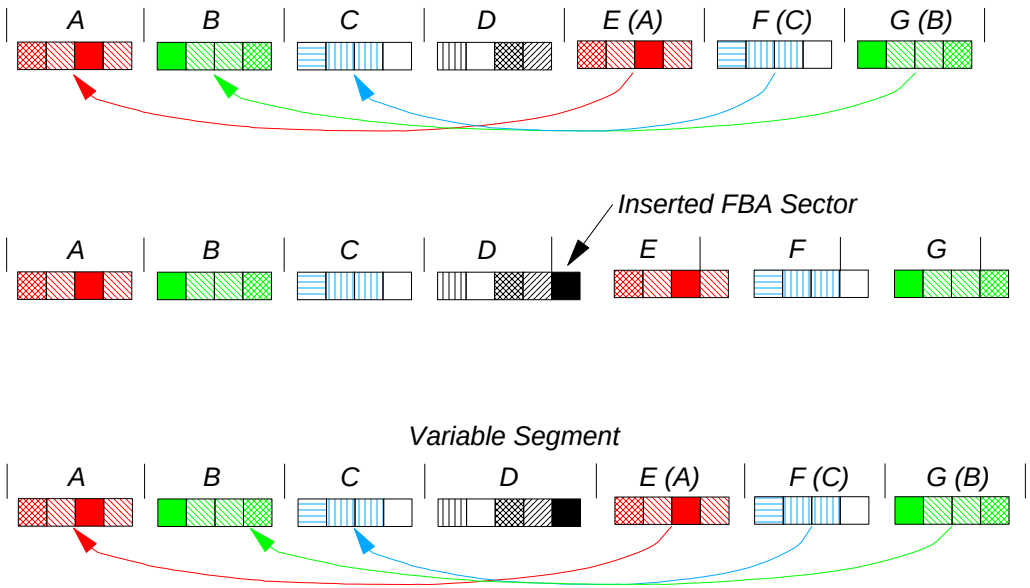
- There is compelling anecdotal evidence for email applications
- Why should this work for z/OS?
  - For a 10 TB store managed with 4K segments, there would be $2.6x10^9$ segments in the store
  - Each 4K segment could be considered as a 32,768 bit integer
  - Each segment could take on 2^{32768} values
  - $2^{32768} >>> 2.6x10^9!!!$
- z/OS backup applications

z/OS Implementations



- Current implementations are based on the EMC DLM and the IBM 7720 tape virtualizations engines
- Data Domain and ProtecTIER appliances are added to these cache only virtual tape subsystems to perform de-duplication

Fixed and Variable Length Segments



- The insertion of a single sector can defeat a fixed length segment algorithm
- The majority of z/OS data is aligned on fixed boundaries
- Hence, we will use a fixed boundary discussion to facilitate the discussion of de-duplication algorithms and to introduce performance metrics for the appliances

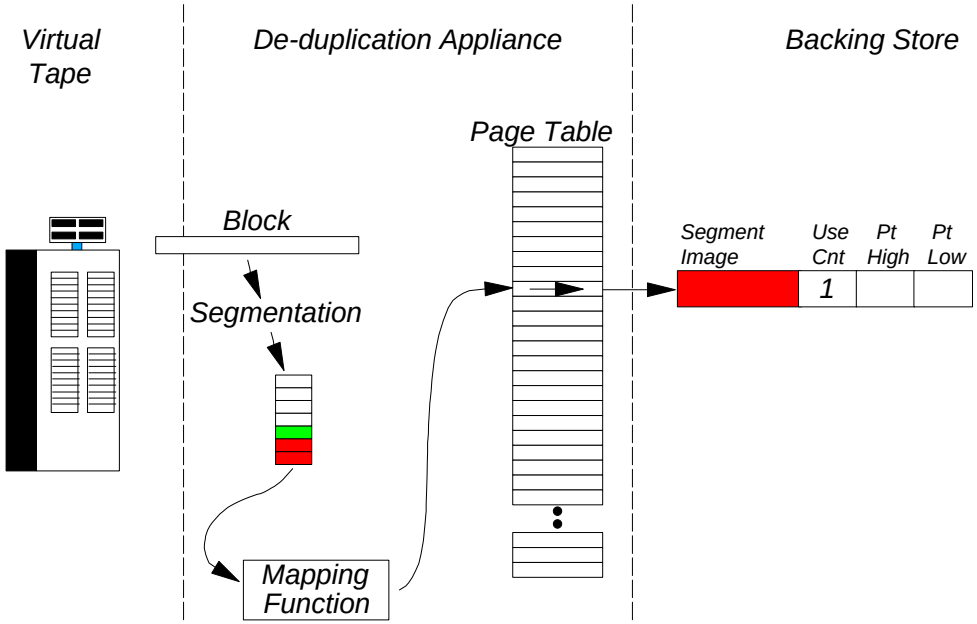
A Hypothetical Model

Full Block Processing

Application Aware Processing

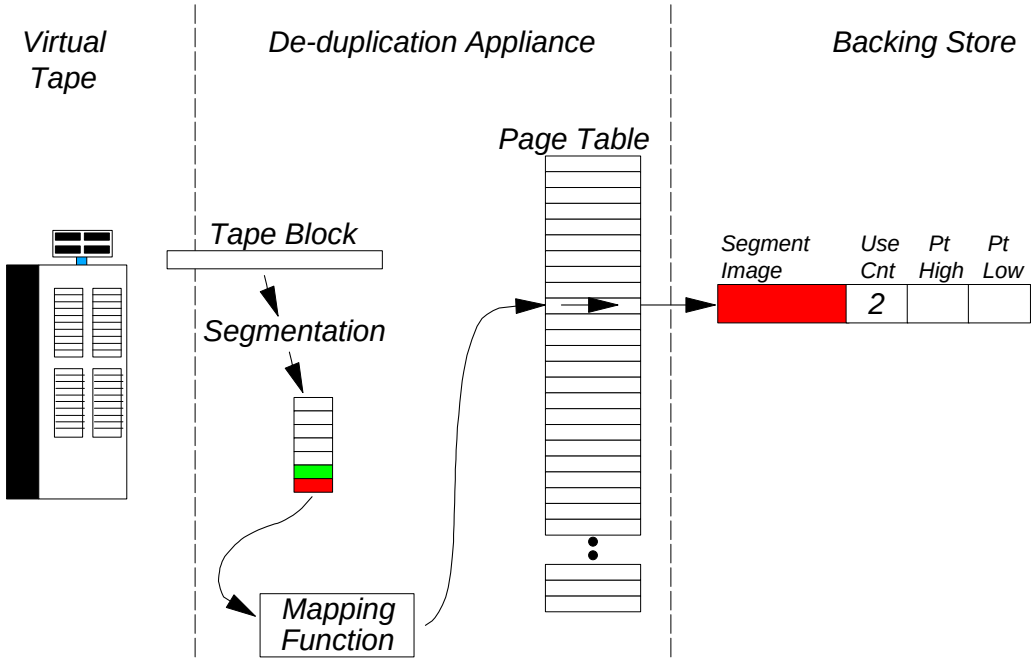
- Transform a file from a series of blocks to a series of pointers to segments
- A working model of de-duplication can be proposed based on segmentation with paging where each file is an address space
- Rather than the paging datasets being transient, they are archive data structures with pages shared (de-duplicated) between the address space
- Many functions can be used to map pages to an in memory 4 to 8 GB page table
- Like z/OS, page datasets may be added on the fly to address capacity and/or improve performance

Initial Page Table Entry



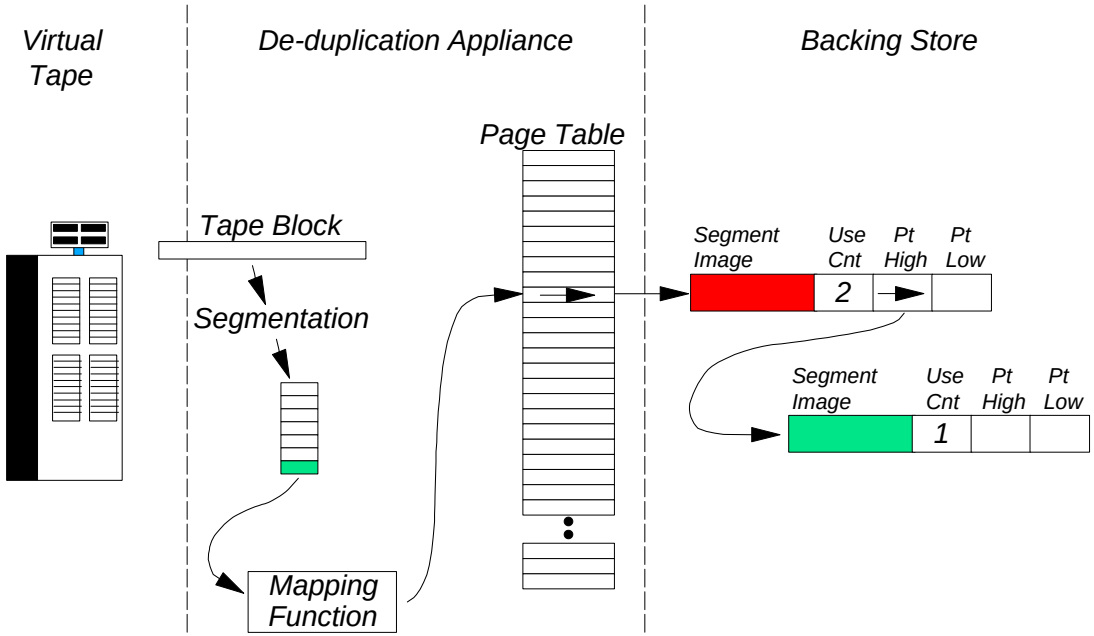
- The first red segment maps to an open entry. A pointer is established and the segment is written to the backing store with a use count of 1. Each page entry includes pointers and counters
- Once a segment has been written to a page dataset, it will never change

Processing a Duplicate Segment



- However, it can be deleted if the use count goes to 0
- When the second red (identical) segment is processed, the use count of the segment in the page dataset is incremented to 2

Managing Collisions



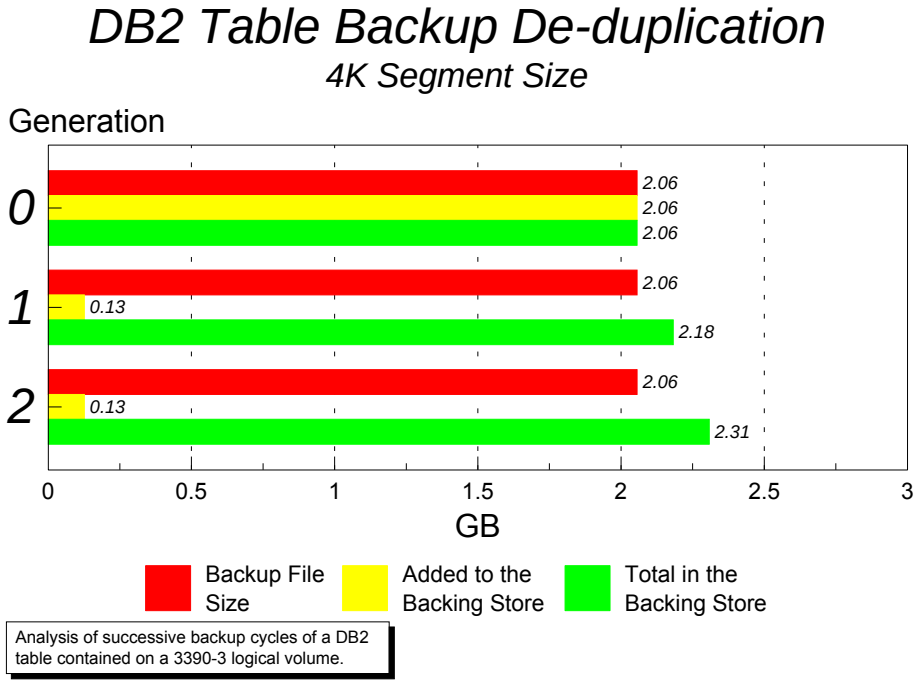
- Assume that the third green segment maps to the same page table entry as the prior two red segments
- While this is unlikely, the algorithm must recognize the collision to maintain data integrity

De-duplication Workload Characterization

Global Uniqueness: characterizes the expected value of the proportion of segments that are not duplicates of a prior segment written to the file or any prior (or future) generation of the file or any other file. For those familiar with tape data compression, it is common to say that a file will compress to 70% (for example) of its original size. If data has a global uniqueness of 95%, that means that only 5% of the blocks would have previously occurred in the file or any other file. That is, the file would de-duplicate to 95% of its native size

Generational Difference: characterizes the difference between two successive generations of the same file. For example, if 5% of the rows in a database changed on a daily basis, the file would exhibit a 5% generational difference

Analysis of DB2 Table Backups



100% global uniqueness

6% generational difference

- **File Size**: the number of GBs written
- **Added to Backing Store**: unique data written to backing store
- **Total in Backing Store**: total space occupied in the backing store at the end of each generation

Performance Metrics

- **Peak Write Data Rate**: maximum MB/Sec rate sustainable for a few minutes or an hour
- **Sustained Write Data Rate**: MB/Sec rate that can be maintained for 6 to 8 hours
- **Sustained Read Data Rate**: maximum MB/Sec that can be read continuously from the subsystem. Unlike normal sequential, no HDD prestage benefits can be expected
- **Mount Time**: is the delay from the time a mount was issued until the first byte of data is passed to host address space
- **De-Duplication Efficiency**: the ratio of the aggregate size of the files written to the de-duplication to the aggregate space occupied in the backing store

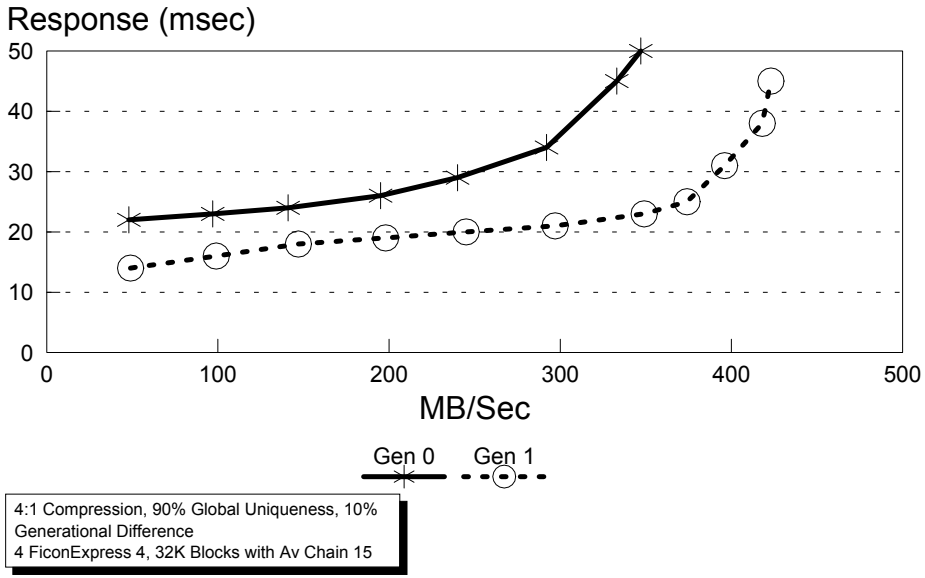
Performance Considerations

- **Concurrency**: since the number of active virtual tapes is effectively the multiprogramming level, performance degradation can be expected with increasing concurrency
- **Garbage Collection**: When a tape is scratched, the following functions must be performed:
  - The use count of every segment referenced by the tape must be decremented
  - In the event that the use count goes to zero, the space occupied by the page can be released
  - Garbage collection may occur on demand or it may be performed periodically as a housekeeping task
- Depending on the ***robustness*** of the appliance, garbage collection may or may not present performance problems

Hypothetical Performance

Tape Data De-Duplication

Data Rate and Response

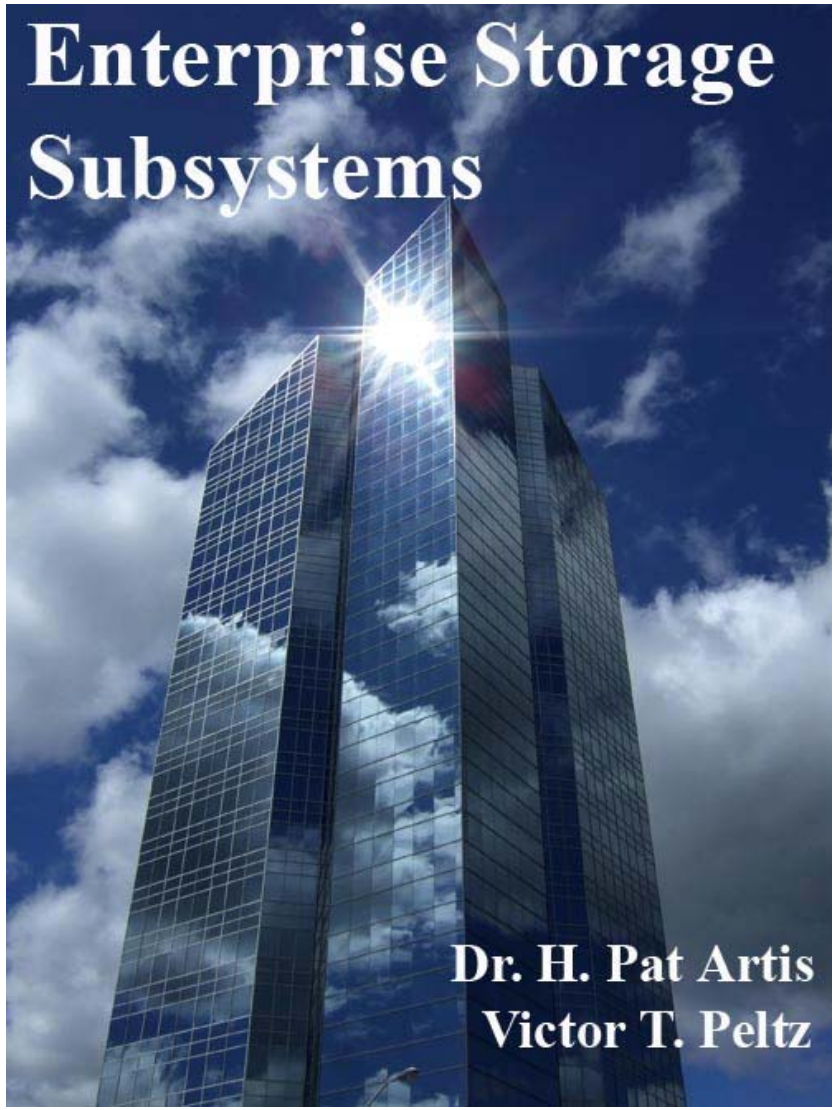


- For generation 0, the file size is reduced as a function of **global uniqueness** and the **compression ratio**
- For generation 1, only 10% of the data (**generational difference**) need be written to the backing store
- Subsequent generations can enjoy performance benefits as a result of reduced I/O to the page datasets
- Standard ***PAI/O Driver for Tape*** experiment

Critical Decision Factors

- The value of the storage saved. This value includes both hardware and environmental considerations
- The per TB licensing fee of the de-duplication appliance
- Suitability of the enterprises data
- Performance considerations
- Transparency of the implementation

Coming in 2011



www.enterprisestoragesubsystems.com

Questions