



Support Technical Exchange



Data Deduplication and Tivoli Storage Manager

Randy Larson – larsonr@us.ibm.com
IBM Advanced Technical Support

August 2008

© 2008 IBM Corporation

Trademarks

The following are trademarks of the International Business Machines Corporation in the United States, other countries, or both.

Not all common law marks used by IBM are listed on this page. Failure of a mark to appear does not mean that IBM does not use the mark nor does it mean that the product is not actively marketed or is not significant within its relevant market.

Those trademarks followed by ® are registered trademarks of IBM in the United States; all others are trademarks or common law marks of IBM in the United States.

For a complete list of IBM Trademarks, see www.ibm.com/legal/copytrade.shtml:

*, AS/400®, e business (logo)®, DBE, ESCO, eServer, FICON, IBM®, IBM (logo)®, iSeries®, MVS, OS/390®, pSeries®, RS/6000®, S/30, VM/ESA®, VSE/ESA, WebSphere®, xSeries®, z/OS®, zSeries®, z/VM®, System i, System i5, System p, System p5, System x, System z, System z9®, BladeCenter®

The following are trademarks or registered trademarks of other companies.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency, which is now part of the Office of Government Commerce.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Disclaimer

- This presentation describes potential future enhancements to the IBM Tivoli Storage Manager family of products
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only
- Information in this presentation does not constitute a commitment to deliver the described enhancements or to do so in a particular timeframe
- IBM reserves the right to change product plans, features, and delivery schedules according to business needs and requirements
- This presentation uses the following designations regarding availability of potential product enhancements
 - Next Release Candidate: Candidate for delivery in the next release
 - Future Candidate: Candidate for delivery in future release

Topics

- Data deduplication technology
- Data reduction and deduplication in TSM



Support Technical Exchange



Data Deduplication Technology

August 2008

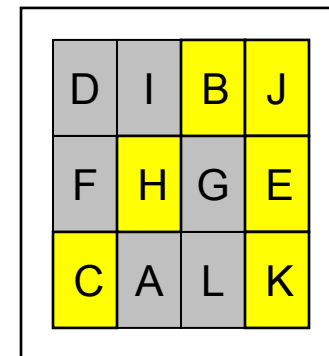
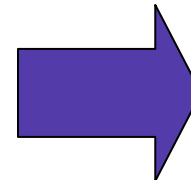
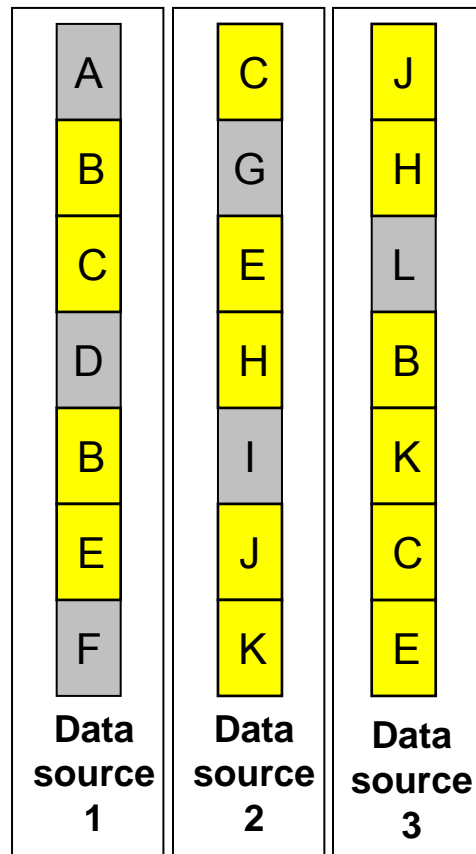
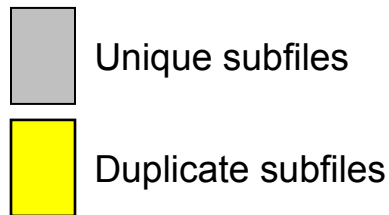
© 2008 IBM Corporation

Data Reduction Methods

Compression	<ul style="list-style-type: none">▪ Encoding of data to reduce size▪ Typically localized, such as to a single file, directory tree or storage volume
Single instance store (SIS)	<ul style="list-style-type: none">▪ A form of compression, usually applied to a large collection of files in a shared data store▪ Only one instance of a file is retained in the data store▪ Duplicate instances of the file reference the stored instance▪ Also known as redundant file elimination
Data deduplication	<ul style="list-style-type: none">▪ A form of compression, usually applied to a large collection of files in a shared data store▪ In contrast to SIS, deduplication often refers to elimination of redundant subfiles (also known as chunks, blocks, or extents)▪ Only one instance is stored for each common chunk▪ Duplicate instances of the chunk reference the stored instance

This terminology is not used consistently throughout the industry. In particular, the terms SIS and deduplication are sometimes used interchangeably.

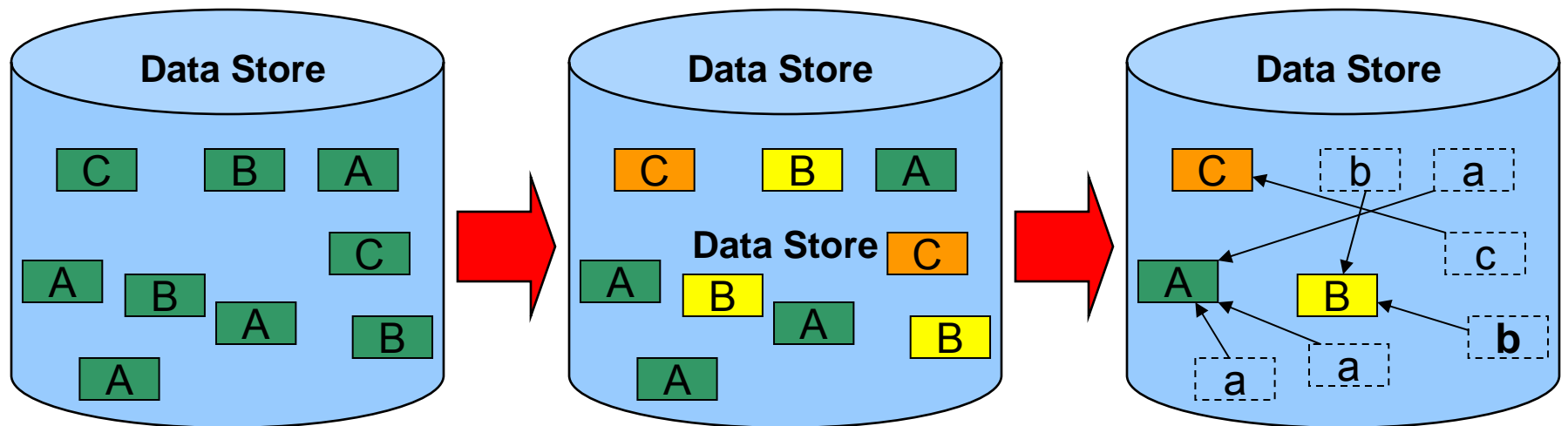
Deduplication Concept



Target data store (backup server)

Data at source locations (backup client machines)

How Deduplication 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

Data Deduplication Value Proposition

Potential advantages

- Reduced storage capacity required for a given amount of data
- Ability to store significantly more data on given amount of disk
- Restore from disk rather than tape may improve ability to meet recovery time objective (RTO)
- Network bandwidth savings (some implementations)
- Lower storage-management cost resulting from reduced storage resource requirements

Potential tradeoffs/limitations

- Significant CPU and I/O resources required for deduplication processing
- Deduplication might not be compatible with encryption
- Increased sensitivity to media failure because many files could be affected by loss of common chunk
- Deduplication may not be suitable for data on tape because increased fragmentation of data could greatly increase access time

Deduplication 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

Where Deduplication is Performed

Approach	Advantages	Disadvantages
<p>Source-side (client-side) Deduplication performed at the data source (e.g., by a backup client), before transfer to target location</p>	<ul style="list-style-type: none"> ▪ Deduplication before transmission conserves network bandwidth ▪ Awareness of data usage and format may allow more effective data reduction ▪ Processing at the source may facilitate scale-out 	<ul style="list-style-type: none"> ▪ Deduplication 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
<p>Target-side (server-side) Deduplication performed at the target (e.g., by backup software or storage appliance)</p>	<ul style="list-style-type: none"> ▪ No deployment of client software at endpoints ▪ Possible use of direct comparison to confirm duplicates 	<ul style="list-style-type: none"> ▪ Deduplication 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 deduplication are not mutually exclusive

When Deduplication is Performed

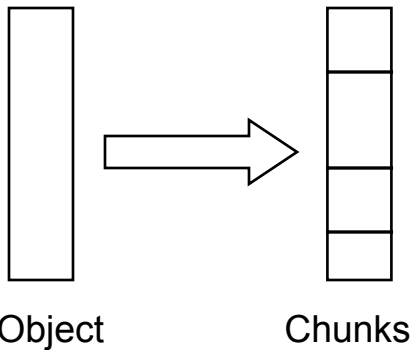
Approach	Advantages	Disadvantages
<p>In-band Deduplication performed during data transfer from source to target</p>	<ul style="list-style-type: none"> ▪ Immediate data reduction, minimizing disk storage requirement ▪ No post-processing 	<ul style="list-style-type: none"> ▪ May be bottleneck for data ingestion (e.g., longer backup times) ▪ Only one deduplication process for each I/O stream ▪ May not support deduplication of legacy data on the target server
<p>Out-of-band Deduplication performed after data ingestion at the target</p>	<ul style="list-style-type: none"> ▪ No impact to data ingestion ▪ Potential for deduplication of legacy data ▪ Possibility for parallel data deduplication processing 	<ul style="list-style-type: none"> ▪ Data must be processed twice (during ingestion and subsequent deduplication) ▪ Storage needed to retain data until deduplication occurs

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

Generalized Deduplication Processing

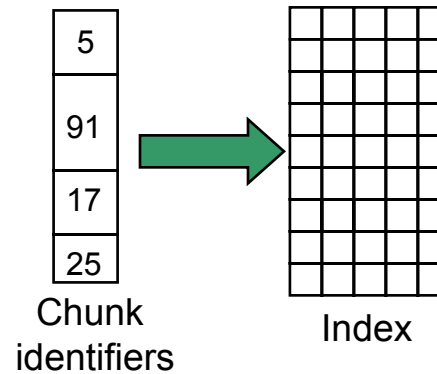
1. Chunk the object

- Divide object into logical segments called chunks



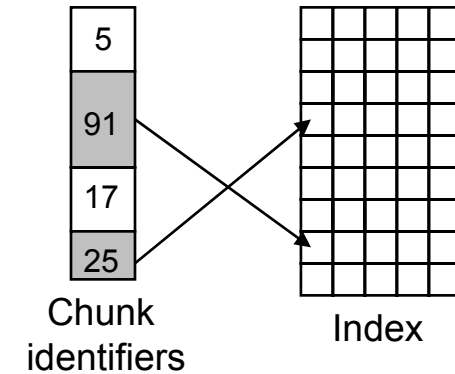
2. Identify duplicate chunks

- Hash each chunk to produce unique identifier
- Compare each chunk identifier with index to determine whether chunk is already stored



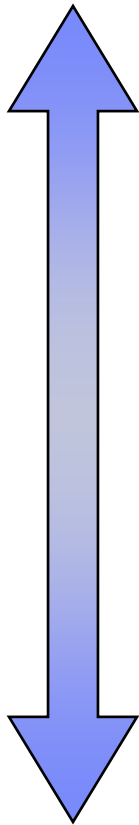
3. Eliminate redundant chunks

- Update index to reference matching chunks
- Deallocate space for redundant chunks



Data Chunking Methods

Lowest overhead
(CPU, I/O, indexing)



Greatest data
reduction

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 (message-digest algorithm)
 - SHA (secure hash algorithm)
- 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

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

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

* "Performance analysis and parallel implementation of dedicated hash functions", Proc. of EUROCRYPT 2002, pp 165-180, 2002.

** The probability of one collision out of k chunks is $p \approx 1 - e^{-(k^2)/2^N}$, where $N=2^L$; when $p=0.5$, we get $k \approx N^{1/2} = 2^{L/2}$ (from birthday paradox).

*** The probability of one hard-drive bit-error is about 10^{-14} .

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

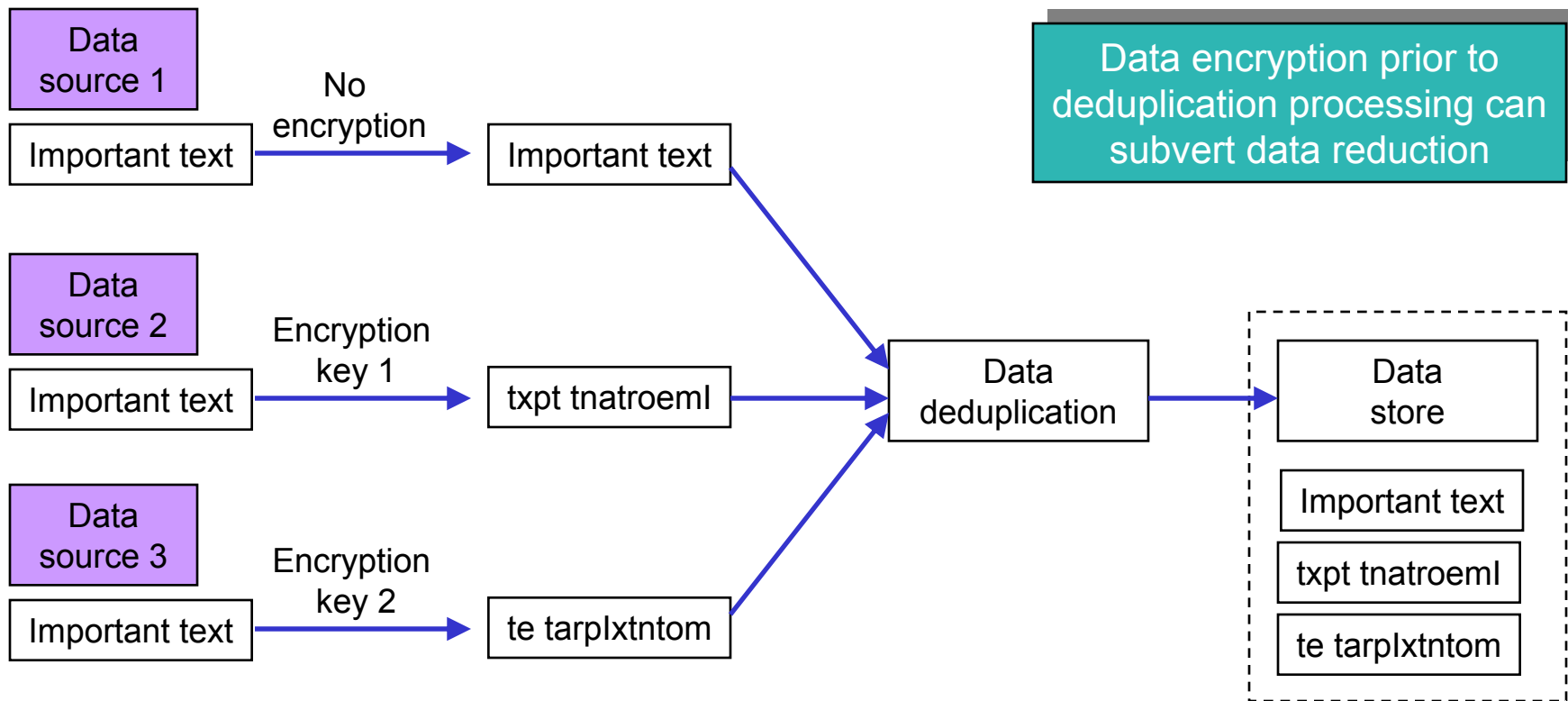
Elimination of Redundant Chunks

- For each redundant chunk, the index is updated to reference the matching chunk
- Index is updated with metadata 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 deallocated and reused
- Deduplication index is critical
 - Integrity
 - Performance
 - Scalability
 - Protection

Deduplication Ratios

- Used to indicate compression achieved by deduplication
- If deduplication reduces 500 TB of data to 100 TB, ratio is 5:1
- Deduplication vendors claim ratios in the range 2:1 to 500:1
- 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 deduplication
- Beware of hype

Deduplication and Encryption



1. Three data sources have the same text file

2. After encryption, text files do not match

3. Deduplication processing does not detect redundancy

4. Text files are stored without data reduction



Support Technical Exchange

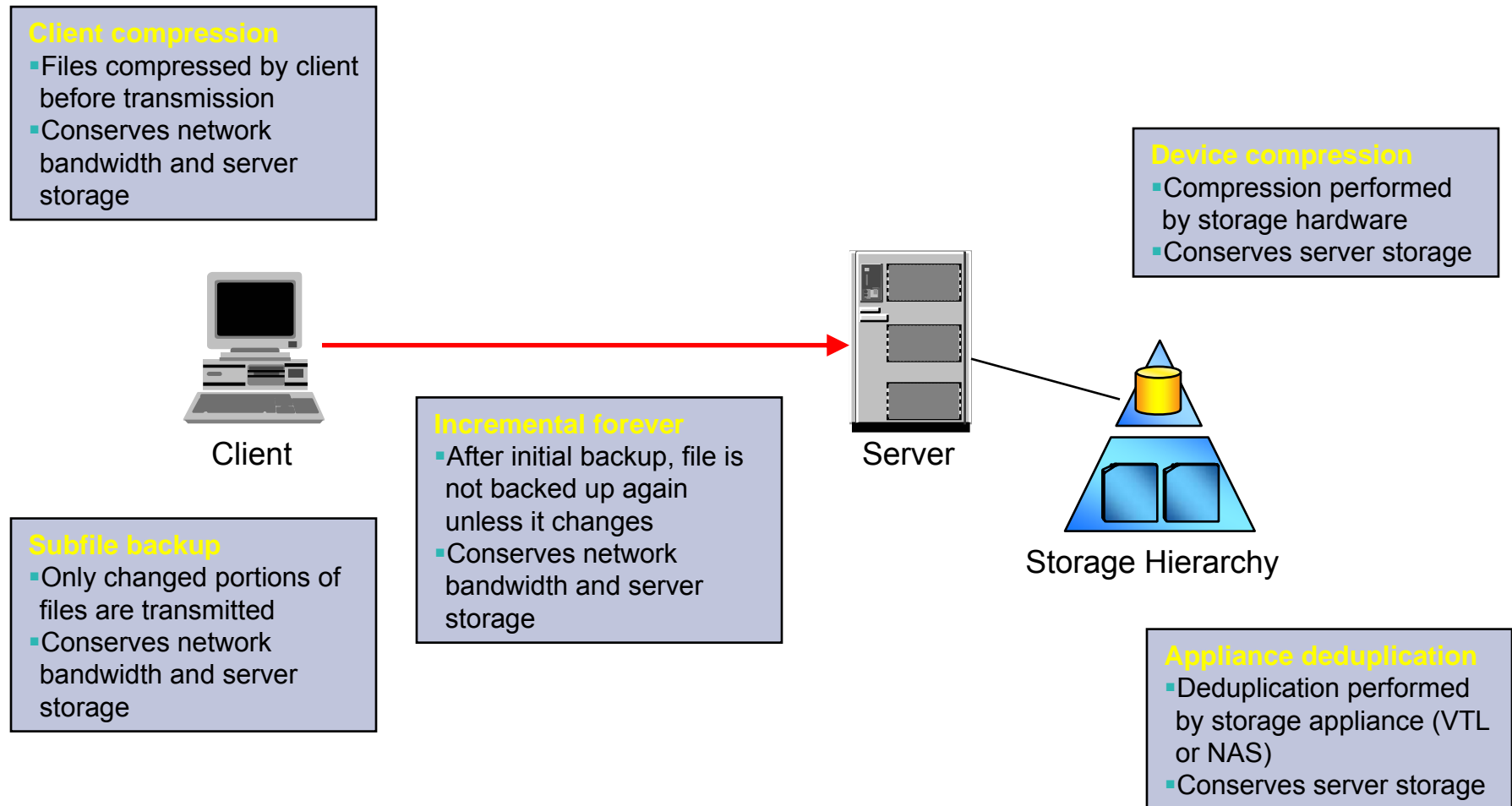


Data Reduction & Deduplication with Tivoli Storage Manager

August 2008

© 2008 IBM Corporation

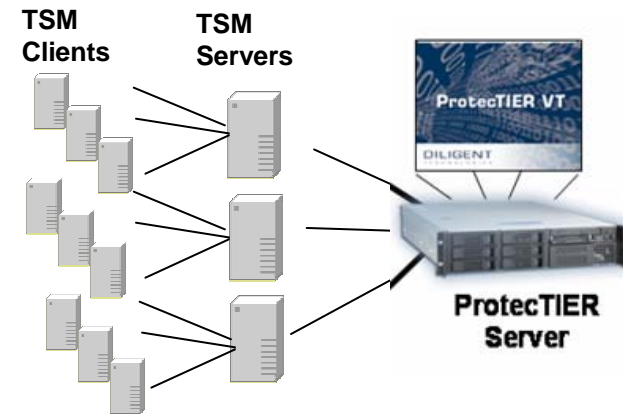
Data Reduction with TSM Today



The Information Avalanche Continues...

How can I reduce storage requirements for my recovery data?

- IBM announced the acquisition of Diligent Technologies on April 18, 2008
- Diligent has market leading Virtual Tape Library (VTL) solutions with:
 - Enterprise-class de-duplication software technology
 - Outstanding inline performance
 - High capacity and scalability
 - Significant User Acceptance



- ✓ Offload processing from TSM
- ✓ Simplified hierarchy configuration
- ✓ Integration of deduplication into TSM backup process
- ✓ Usage by many TSM servers

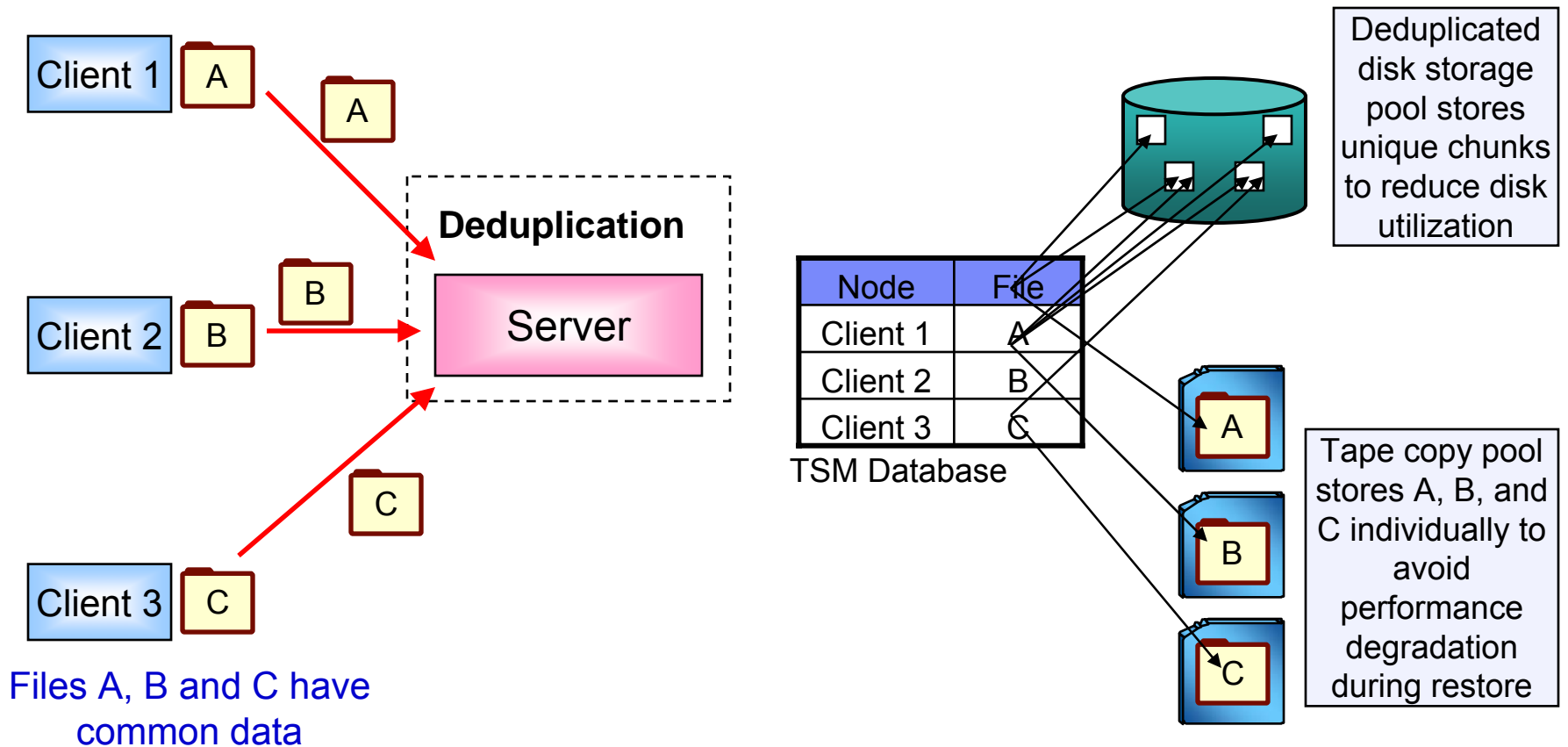
A large number of Diligent customers today use TSM

Native Data Deduplication in TSM

- TSM's incremental forever methodology greatly reduces data redundancy as compared to traditional methodologies based on periodic full backups
- Consequently, there is less potential for data reduction via deduplication in TSM as compared to other backup products
- Nevertheless, deduplication is an important function to TSM because it will allow more data objects to be stored on a given amount of disk for fast access
- Native deduplication is a key product enhancement in TSM

Next Release Candidate

TSM Deduplication Overview



Allows more objects to be stored on disk for fast access

Next Release Candidate

Design Points for Initial TSM Deduplication Solution

Design point	Comments
Server-side	<ul style="list-style-type: none"> ▪ Avoids need for deployment of client software ▪ Effective for all types of stored data
Out-of-band	<ul style="list-style-type: none"> ▪ Allows deduplication of legacy data in addition to new data ▪ Minimizes impact to backup windows ▪ Concurrent processes to identify duplicate data
Index maintained in TSM server database (DB2)	<ul style="list-style-type: none"> ▪ Transactional integrity, scalability, performance, disaster protection
Variable-size chunking	<ul style="list-style-type: none"> ▪ Rabin fingerprinting with awareness of TSM data format
SHA-generated identifiers for detection of duplicate chunks	<ul style="list-style-type: none"> ▪ Probably SHA-1 or SHA-256 ▪ Longer identifiers of SHA-256 would reduce collision probability, at the expense of increased processing and database space usage
Average chunk size to be determined	<ul style="list-style-type: none"> ▪ Larger chunks require less database overhead ▪ Larger chunks reduce the total number of chunks required for given amount of data and therefore reduce collision probability ▪ Smaller chunks improve deduplication ratio
Space occupied by redundant chunks will be recovered during reclamation	<ul style="list-style-type: none"> ▪ Allows coordinated recovery of space occupied by deleted objects and redundant chunks

Next Release Candidate

Expected Deduplication Behavior

- Disk storage requirement reduced via optional data deduplication for FILE storage pools
- Deduplication processing performed on TSM server and tracked in database
- Reduced redundancy for
 - Identical objects from same or different client nodes (even if file names are different)
 - Common data chunks (subfiles, extents) in objects from same or different nodes
- Post-ingestion (out-of-band) detection of duplicate data on TSM server to minimize impact to backup windows
- Space occupied by duplicate data will be removed during reclamation processing
- Allowed for all data types: backup, archive, HSM, TDP, API applications
- Transparent client access to deduplicated objects

Next Release Candidate

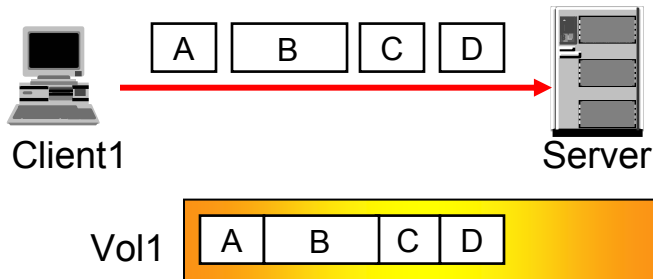
Expected Deduplication Behavior

- Deployment of new clients or API applications not required (but use of next release client or higher may improve deduplication ratio)
- Legacy data stored in or moved to enabled FILE storage pools can be deduplicated
- Data migrated or copied to tape will be reduplicated to avoid excessive mounting and positioning during subsequent access
- Ability to control number, duration and scheduling of CPU-intensive background processes for identification of duplicate data
- Reporting of space savings in deduplicated storage pools
- Deduplication processing will skip client-encrypted objects, but should work with storage-device encryption
- Native TSM implementation, with no dependency on specific hardware

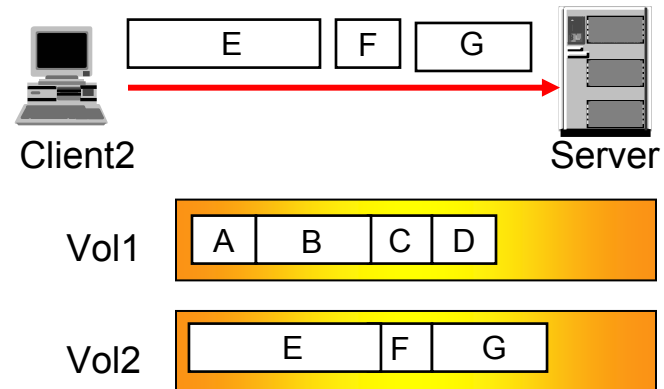
Next Release Candidate

Deduplication Example

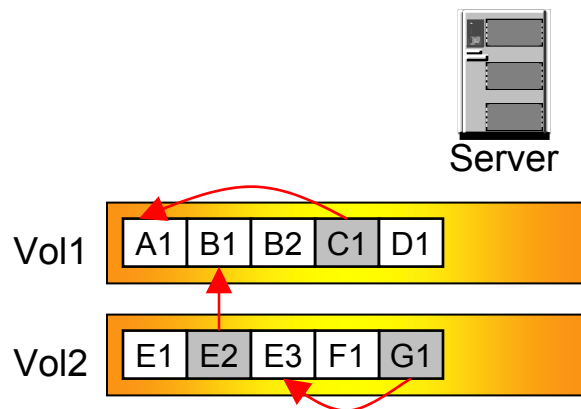
1. Client1 backs up files A, B, C and D. Files A and C have different names, but the same data.



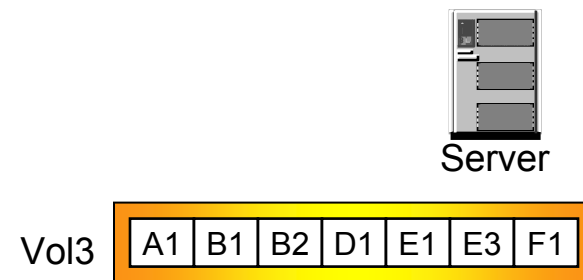
2. Client2 backs up files E, F and G. File E has data in common with files B and G.



3. Server process “chunks” the data and identifies duplicate chunks C1, E2 and G1.



4. Reclamation processing recovers space occupied by duplicate chunks.



Next Release Candidate

Comparison of TSM Data Reduction Methods

	Client compression	Incremental forever	Subfile backup	Deduplication
How data reduction is achieved	Client compresses files	Client only sends changed files	Client only sends changed subfiles	Server eliminates redundant data chunks
Conserves storage pool space?	Yes	Yes	Yes	Yes
Conserves network bandwidth?	Yes	Yes	Yes	No
Data supported	Backup, archive, HSM, API	Backup	Backup (Windows only)	Backup, archive, HSM, API
Scope of data reduction	Redundant data within same file on client node	Files that do not change between backups	Subfiles that do not change between backups	Redundant data from any files in storage pool
Avoids storing identical files renamed, copied, or relocated on client node?	No	No	No	Yes
Removes redundant data for files from different client nodes?	No	No	No	Yes

Next Release Candidate

Considerations for Use of TSM Deduplication

- Consider deduplication if
 - Data recovery would improve by storing more data objects on limited amount of disk
 - Data will remain on disk for extended period of time
 - Much redundancy in data stored by TSM (e.g., for common operating-system or project files)
 - TSM server CPU and disk I/O resources are available for intensive processing to identify duplicate chunks

- Deduplication might not be indicated for
 - Mission-critical data, whose recovery could be delayed by accessing chunks that are not stored contiguously
 - TSM servers that do not have sufficient resources
 - Data that will soon be migrated to tape

Next Release Candidate

Potential Follow-on Enhancements

- The initial TSM deduplication solution is designed to allow extensibility
- Depending on business priorities, possible future extensions to this solution could include
 - Option to perform inline deduplication during data ingestion (to achieve immediate compression)
 - Client-side deduplication (to distribute processing and conserve network bandwidth)
 - Option to control which hash function is used (tradeoff between performance and probability of false match)
 - Deduplication support for random-access disk or tape storage pools
 - Policies to control deduplication based on node, filespace, file size, or other criteria

Future Candidates

Summary

- Data deduplication can reduce storage requirements, allowing more data to be retained on disk for fast access
- Deduplication involves tradeoffs relating to degree of compression, performance, risk of data loss and compatibility with encryption
- TSM's incremental forever method avoids periodic full backups, reducing the potential for additional data reduction via deduplication
- Server-side deduplication is a key enhancement in TSM