Daten-DeDuplizierung (abgekürzt „DeDupe“) bezeichnet verschiedene Verfahren, um automatisiert redundante Files, Bytes oder Datenblöcken zu entfernen. Die eliminierten Daten werden durch Pointer ersetzt. Ziel ist es, die Menge zu speichernder Daten entweder bereits an der Quelle (Source) oder am Ziel (Target) zu reduzieren. Das hierzu nötige Repository (Metadaten-Index) zur Identifikation von nicht-redundanten Daten liefert im Zusammenspiel mit der eigentlichen DeDuplizierung-Engine (Software, Algorithmen) die Grundlage zu Implementierung entsprechender Lösungen. Es kommen in der Regel sog. Pattern-Matching- (Delta-based Suchmuster) oder hashbasierte Technologien zur Anwendung.

Die DeDuplizierung kann derzeit sowohl über (Backup-) Software als auch mittels dedizierte Hardware (Appliance, Array, Virtual-Tape-System) umgesetzt werden.

Grundsätzliche Begriffe und verwendete Techniken:

Beim **Single Instancing** werden ganze Objekte (z.B. Files) nur einmal abgespeichert. Nachteil: Auch wenn sich nur ein Zeichen in einer Datei ändert, wird der gesamte Inhalt doppelt gespeichert. Der Einsparungseffekt ist hier also geringer als beim Entfernen von Block-basierten Redundanzen. Beispiel: Von 20 Kopien eines Word Dokuments enthält jedes Doppel nur eine unterschiedliche Seite; einem Datenreduktionssystem auf Dateiebene erscheinen die Kopien als 20 unterschiedliche Files. Das Single-Instancing-Verfahren wird häufig in Verbindung mit inkrementeller oder differentieller Datensicherung eingesetzt.

DeDupe auf Block-Level ist die Form des Single Instancing auf der Ebene unterhalb von Dateien (Sub-File-Level). Der Begriff „DeDuplikation“ wird deshalb meist in diesem Zusammenhang verwendet. Bei DeDupe auf Blockebene hingegen lassen sich alle redundanten Daten entfernen und weniger Speicherplatz auf der Festplatte wird benötigt.

Ein leistungsstarkes DeDuplizierungs-Verfahren beruht auf **Blocks mit variabler Länge.** Die Methode analysiert eine Sequenz von Daten und teilt sie in verschiedene Blöcke von variabler Länge auf. Wird ein redundanter Block erkannt, legt das System einen auf den Originalblock verweisenden Pointer ab (Metadaten), statt den Block erneut zu speichern. Da der Pointer deutlich weniger Platz in Anspruch nimmt als der Block selbst, wird Speicherplatz einspart. Bei Backup-Jobs in denen dieselben Blöcke immer wieder vorkommen, kann so das 10- bis 50-fache an Daten auf dem entsprechenden Speichersystem gespeichert werden. Welches ist für welche Anwendung das jeweils effizienteste Verfahren und was der Algorithmus mit der höchsten bzw. sichersten DeDupe-Rate? Bevor wir uns dieser Fragestellung widmen, soll noch kurz auf den **Unterschied** zwischen **DeDuplizierung** und **Komprimierung** hingewiesen werden.

DeDupe ist eine Weiterentwicklung von Kompressions-Algorithmen, die für Files benutzt wurden. Ausgangspunkt war das sog. „Check Sum Verfahren“ aus der Kryptografie, um verschlüsselte Signaturen zu erzeugen (SHA0,1, MD5 etc.), ist also prinzipiell keine neue Technik. Jedoch wurden die mathematischen Verfahren bei DeDupe fort entwickelt und für Block-, Byte-, Bit-Ebene und File-Level erweitert. Bei der Datenkomprimierung wird die Darstellung von Daten so modifiziert, dass weniger Platz dafür benötigt wird. Eine Sequenz gleicher Gruppen von Zeichen z.B. lässt sich durch eines dieser Zeichen oder Zeichengruppen mit dem jeweiligen Wiederholungsfaktor ersetzen. Ebenso wie bei der DeDuplizierung speichert die Komprimierung die redundante Daten nicht weiter ab, liefert allerdings im Ergebnis eine geringerer Effizienz (realistische Werte liegen bei 1:2 - abhängig vom Datentyp). Die Deduplizierung sollte idealerweise durch eine lokale Komprimierung ergänzt werden, wodurch die Datengröße noch einmal halbiert wird und damit eine erhebliche Datenreduzierung erreichen wird.

Wie arbeitet DeDuplizierung?

DeDupe funktioniert mit Algorithmen, die nach doppelten Datenobjekten – Chunks oder Blöcken - suchen und zu diesem Zweck eine eindeutige Identifikation über den Hash-Wert der Objekte bilden. Dazu muss ein Katalog aller bisherigen Werte erzeugt werden (Metadaten-Index). Als nächstes vergleicht man die Hash-Werte und speichert bei erstmaligem Auftreten eines Objektes dieses dann ab (ein Chunk ist in etwa wie der menschliche Fingerabdruck, also einmalig). Die Chunk-Inhalte werden nun miteinander verglichen und nur dann gesichert, wenn sie als Inhalt bislang noch nicht identifiziert wurden. Das bedeutet: Bei redundanten Objekten wird nur ein Pointer zum ersten gespeicherten Objekt erzeugt.

Es wurde über die Möglichkeit von Datenverlusten bei DeDupe durch Hash-Kollisionen berichtet und auf Grund von fehlerhaften Kalkulationen bei den ersten Hash-Code-Implementierungen auch vereinzelt in der Praxis beobachtet. Eine Kollision bedeutet, dass zwei überprüfte Datensätze dieselbe Identifikationsnummer erhalten – also denselben Hash-Code erzeugen – obwohl sie aus jeweils unterschiedliche Bitfolgen bestehen. Dies ist möglich (als Funktion der Datenmenge bzw. der Hash-Implementierung), allerdings verfügen moderne Hash-Codes über eine ausgefeilte Integritätsprüfung und die Fehlerrate ist laut Aussagen von Anbietern damit beispielsweise geringer als die mögliche Bitfehler-Rate bei heutigen LTO-Tapes.

Anforderungen an DeDuplizierungslösungen

In modernen Speichersystemen werden immer öfter verschiedene Standardprotokolle verwendet, z. B. NFS, CIFS, Block, iSCSI, FC und VTL (siehe auch „Unified Storage“), da diverse Anwendungen unterschiedliche Protokolle zur selben Zeit benötigen. Benutzer-Startverzeichnisse können sich auf einem NAS-Server befinden; der Exchange-Server sollte in Blöcken ausgeführt werden und für Sicherung wird eine VTL bevorzugt.

Im Laufe eines Lebenszyklus werden dieselben Daten möglicherweise mit allen Protokollen gespeichert. Eine Präsentation, die in einem Startverzeichnis beginnt, kann per E-Mail gesendet oder vom E-Mail-Server in Blockformat gespeichert und dann von einer E-Mail-Archivierung auf einem NAS-Server abgespeichert werden. Diese kann vom Startverzeichnis, E-Mail-Server und von der Archivierungsanwendung in der VTL gesichert werden. Bei der DeDuplizierung sollten somit unabhängig von der Art der Speicherung möglichst alle redundante Daten gefunden werden.

Verschiedene DeDuplizierungs-Verfahren

Wo wird DeDupliziert? Beim DeDuplizieren an der Quelle - **Source-based DeDupe** - wird der Hash-Wert am Backup-Client erzeugt und verglichen.

Vorteil: Es werden weniger Daten durch das Netzwerk transferiert. Nachteil: Der Deduplizierungs-Index kann je nach Implementierung über das Netzwerk verteilt sein und ist schwieriger zu verwalten; daneben sind Verfügbarkeitsaspekte zu beachten. Das Auslesen von deduplizierten Daten kostet normalerweise hier etwas mehr Zeit (potentielle Performance-Probleme), was für die Online-Arrays problematisch sein kann. Auch gilt es zu beachten: Wenn während des Backups die DeDuplizierung auf dem Server läuft, werden zwar weniger Daten über die Backup-Verbindung übertragen, jedoch muss Software auf allen gesicherten Hosts verwaltet werden. Der Backup-Prozess selbst verlangsamt sich, da die DeDuplizierung systembedingt einen Overhead erzeugt; andere auf dem Server laufende Anwendungen können davon auch betroffen werden.

Grundsätzlich stellt die DeDuplizierung des Primärspeichers derzeit eine sehr interessante Option dar, speziell im Bereich des Virtual-Machine-Managements, da hier viele redundante Daten entstehen, die zeitnah gesichert werden sollten. Source-based DeDupe empfiehlt sich deshalb für Unternehmen, die erst kürzlich Investitionen in die Storage-Hardware getätigt haben oder keine weitere separate Hardwareplattform für Backup und DeDupe verwalten wollen.

Bei der **DeDuplikation am Ziel (Target)** müssen die Daten zuerst durch das Netzwerk, bevor eine Reduzierung erreicht wird.

Vorteil: Alle Komponenten zur Verwaltung dieser Systeme sind zentralisiert und damit leichter zu administrieren. Wenn Sie am Backup-Ziel deduplizieren, wird eine größere Menge an Daten über die Verbindung übertragen; jede gängige Backup-Software kann in der Regel verwendet werden. Die Leistung ist höher, da das Hardwaresystem speziell für DeDupe optimiert wurde. Zielsysteme (Targets) sind für diesen Zweck optimierte Disk-Arrays mit schnellem Halbleiter-Cache und Solid-State-Disks (SSDs). Moderne Lösungen lassen sich meist als Virtual Tape Library (VTL) konfigurieren, stellen sich der Backup-Software also wie reale Tape-Bibliotheken dar. Sie erlauben aber

natürlich weiterhin den Anschluss von realen Tape Drives für Archiv- oder Desaster-Recovery-Zwecke. Klassische VTLs mit Datensicherung auf Tape anstelle Disk verwenden typischerweise Standard-Komprimierungsverfahren anstatt DeDuplikation.

Nachteil: Beim Post-Processing (siehe auch „Wann soll dedupliziert werden“) können größere Investitionen in schnelle SAS- oder FC-Disk-Arrays nötig sein. Des Weiteren fallen je nach Anbieter bzw. Lizenzpolitik zusätzliche Software-Lizenzkosten für entsprechende Sicherungskapazitäten an.

Target-DeDupe ist also für Organisationen interessant, die derzeit über keine weiteren leistungsfähigen Kapazitäten für D2D-Backup und DeDupe verfügen oder eine Backup-Software einsetzen, die DeDupe nicht gut unterstützt. Hier ist die Anschaffung einer dedizierten Disk-Backup DeDupe-Lösung (Array und/oder Appliance) eine Alternative.

Rein **Software-basierte DeDuplizierungslösungen** erfordern eine sorgfältige Planung, da sich aus ihr Folgen für die Serverbelastung, Festplattenkapazitäten und den Netzwerkdurchsatz ergeben können. Andererseits bietet die Software per se den Vorteil von Hardware-Unabhängigkeiten: In der Regel lassen sich auch die bestehende Backup-Prozesse unverändert weiterführen und es wird kein zusätzlicher neuer Management-Layer neben der bereits verwendeten Backup-Software erzeugt.

Wann soll dedupliziert werden? Post-Processing bedeutet, dass die Daten erst dedupliziert werden, nachdem sie auf das Speicherarray geschrieben wurden. Es ist deshalb nötig, entsprechende Speicherkapazität plus weiteren Platz für die deduplizierte Daten vorzuhalten.

Vorteil: Die Prozesse können ohne Beeinträchtigung des Backup-Vorgangs offline durchgeführt werden und die hierfür nötige CPU-Belastung ist in der Regel sehr gering. Nachteil: Sollte die für eine nachträgliche DeDuplizierung benötigte Zeit bis zum nächsten Backup nicht reichen, werden weitere Sicherungsläufe davon negativ betroffen; somit müssen natürlich auch die verbundenen Prozesse für weitergehende Desaster-Recovery-Mechanismen (Datenreplizierung) bis zum erfolgreichen Beenden des De-Dupe-Prozesses beeinträchtigt (Latency).

Inline-DeDupe hingegen ermöglicht die DeDuplizierung „on the fly“, d.h. schreibt Daten direkt auf das Speichermedium.

Vorteil: Die Kapazitätsreduzierung arbeitet sofort, da die DeDupe-Algorithmen in „Realtime“durchgeführt werden. Nachteil: Es müssen genügend leistungsfähige und skalierbare (Anforderungen steigen mit der Zeit) Server- plus CPU-Ressourcen für DeDupe zur Verfügung stehen. Dank Multicore-CPUs und optimierter Algorithmen ist die Inline-DeDupe inzwischen aus Performancesicht ein empfehlenswertes Verfahren (die Leistung skaliert linear mit der Hardware-Performance)

DeDuplizierung in Verbindung mit Datenreplikation

Bei der Replikation werden duplizierte Daten von einer Quelle an ein Ziel übertragen und es werden normalerweise alle Backup-Daten repliziert. Dazu ist allerdings ein leistungsfähiges Netz erforderlich. Beim DeDupe sucht das Quellsystem im Replikationsstrom nach duplizierten Blöcken. Wenn bereits ein Block an das Zielsystem übertragen wurde, ist dieser nicht mehr zu übertragen, sondern nur ein entsprechender Verweis darauf. Da der Pointer deutlich kleiner als der Block ist, wird zur Replikation geringere Netzbandbreite benötigt.

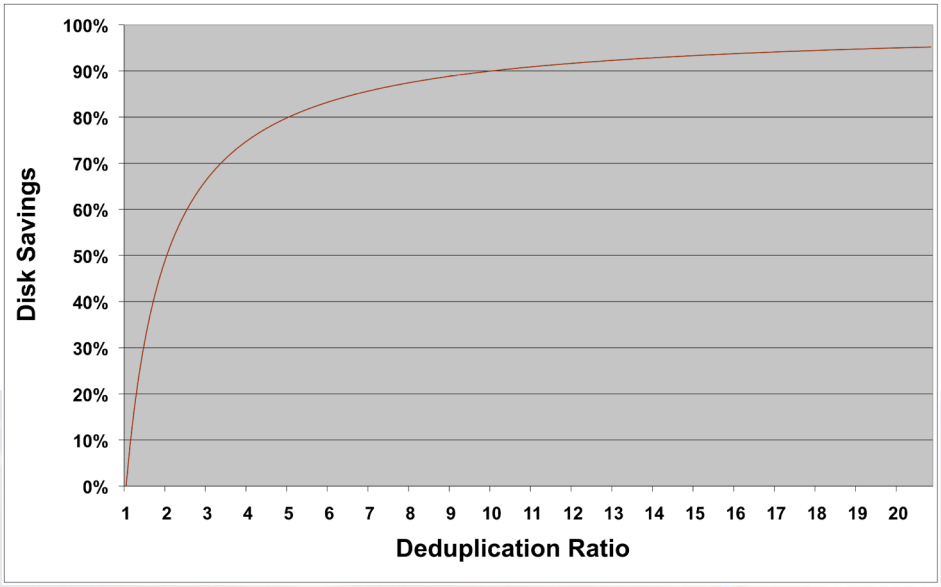
Welche Applikationen sind geeignet?

Virtualisierte Umgebungen mit sehr vielen virtuellen Maschinen sind gute DeDupe-Kandidaten. Zu beachten ist: Wenn Daten der VMs durch Source-DeDupe bereits reduziert worden sind, müssen diese vor dem Backup jedoch zur Verfügung stehen; dies kann einen Mehraufwand für die Administration bedeuten und Automatisierungstools sind nötig. Als effektiv hat sich die Sicherung auf schnelle Disk-basierte Backupsysteme (integrierte Arrays / Appliances) mit DeDupe-Funktionalitäten herausgestellt. Folgende Vorgehensweise ist prinzipiell sinnvoll (Skripting nötig): 1. Mittels konsistenten Snapshots VMDKs in das D2D-Backup-/DeDupe-System kopieren. 2. Diese Sicherungsdaten in das zweite System am Desaster-Recovery (DR-)Standort replizieren.

Wenn DeDupe für das Backup eingesetzt wird, sind alle gängigen Anwendungen wie Mail, Datenbanken, File-Services sowie alle gängigen Backup-Lösungen unterstützt. Bei der Deduplizierung mit variabler Blocklänge lassen sich in der Regel für nahezu alle Anwendungen im Backup-Strom redundante Blöcke finden. Für bestimmte Dateitypen entstehen beim ersten Deduplizierungslauf nur geringe Vorteile, weil die Anwendung selbst im Vorfeld Redundanzen beseitigt. Werden allerdings mehrere Backup-Läufe durchgeführt – oder nachdem Änderungen vorgenommen wurden – kann DeDupe sehr wirksam sein (Beispiel: MultiMedia Apps).

Erfolgt die Source-basierte DeDuplikation in den Außenstellen von Unternehmen, dann reduziert sich das Volumen der zentral zu sichernden Daten und es müssen deutlich weniger Daten transferiert werden, was wiederum eine geringere Netzbandbreite zwischen Außenstelle und Zentrale erfordert und damit reduzierte Kosten nach sich zieht.

In Verbindung mit geeigneten WAN-Optimierungsverfahren (siehe z.B. Riverbed-Lösungen) verbessert dies die Planbarkeit und Durchführung von DR-Prozessen und der Backup-Effizienz durch die Zentralisierung und Automation der Verfahren über weniger Hardware und Verwaltungsaufwand, geringere Fehlerquoten beim Backup und höhere Datenintegrität. Verschlüsselte Daten lassen sich natürlich nicht sinnvoll Deduplizieren, d.h. zuerst eine Reduktion der Redundanz und dann die Verschlüsselung durchführen.



<u>Anlage A:</u> Graph DeDupe - Effizienz (DeDuplizierungs-Ratio und Diskeinsparung beim Backup)

Quelle: Quantum Corp., US

Autor:

Norbert E. Deuschle, Business Consulting & Research

Für das Storage Consortium

Starnberg, im August 2011