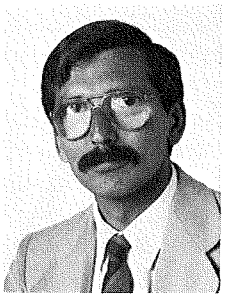
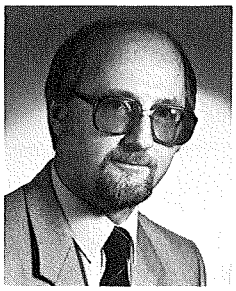
H.-G. Stork und W. Stucky, Universität Karlsruhe

Zur Anwendung von Datenkomprimierungsverfahren, speziell des „Frankenstein-Lidzba-Verfahrens"



Stork, Dr. Hans-Georg, Promotion in Informatik an der TH Darmstadt. Jetzt wissenschaftlicher Angestellter und Lehrbeauftragter am Institut für Angewandte Informatik und Formale Beschreibungsverfahren der Universität Karlsruhe.



Stucky, Dr. Wolffried, Promotion in Mathematik an der Universität Saarbrücken. Jetzt Ordinarius für Angewandte Informatik/Fakultät für Wirtschaftswissenschaften der Universität Karlsruhe. Forschungsgebiete: Datenbank- und betriebliche Informationssysteme.

Datenkomprimierung hat ein breites und sinnvolles Anwendungsfeld. Ihr ökonomischer Nutzen ist beträchtlich. Das „Frankenstein-Lidzba-Verfahren" ergibt nach unserer Erkenntnis für typische kommerzielle Daten einen erheblich besseren Effekt als traditionell übliche Verfahren.

1 Einleitung

Im Rahmen vieler kommerzieller Rechner-Anwendungen gewinnt die Verwendung von Datenkomprimierungs-Verfahren beständig an Bedeutung. Nur mit ihrer Hilfe ist es häufig möglich, der immer stärker wachsenden Datenflut Herr zu werden. Im vorliegenden Artikel diskutieren wir derartige Verfahren und richten unser Augenmerk speziell auf ein seit kurzem für den Einsatz unter BS2000 und OS/MVS erhältliches Produkt. Dieses Produkt basiert auf dem von R. Lidzba und H.-U. Wiebach entwickelten „Frankenstein-Lidzba-Verfahren" (vergl. LIWI). Wir zeigen, daß es den Anforderungen der Praxis weitgehend entspricht bzw. daß es herkömmliche Verfahren in vielen Fällen bei weitem übertrifft.

Unser Beitrag zerfällt in drei Hauptteile. Im ersten Teil werden allgemein die Möglichkeiten und der Nutzen von Datenkomprimierungen in verschiedenen Anwendungsbereichen der elektronischen Datenverarbeitung dargestellt sowie „Pros" und „Contras" diskutiert.

Der zweite Teil enthält in kurzer Zusammenfassung einen Überblick über gebräuchliche Komprimierungsmethoden.

Der dritte Teil schließlich betrifft speziell das von Lidzba und Wiebach entwickelte „Frankenstein-Lidzba-Verfahren". Die Beurteilung dieses Verfahrens gründet sich auf Tests, welche mit einer in Software implementierten Version durchgeführt wurden.

2 Anwendungen

Zahlreiche kommerzielle und technisch-wissenschaftliche Rechneranwendungen sind durch große Datenvolumina charakterisiert, welche gespeichert, übertragen und im Zugriff gehalten werden müssen. Die hier zur Debatte stehenden Anwendungen beziehen sich unter anderem auf

- große (verteilte) Daten- bzw. Dokumenten-Banken (z.B. in der Materialwirtschaft, in der öffentlichen Verwaltung, im Rechnungs- und Personalwesen etc.),
- Verarbeitung statistischer Daten,
- Verarbeitung von Meßdaten,
- Datenverarbeitung in großen, kundenintensiven Unternehmungen (wie Banken, Versicherungen, Handel, etc.),
- rechnergestütztes Entwerfen bzw. Konstruieren (CAD).

Obgleich in den letzten Jahren sowohl in der Speicher- als auch in der Übertragungstechnologie beträchtliche Fortschritte erreicht wurden, bleiben Speicherung und Übertragung von Daten in den genannten Anwendungsbereichen hemmende Engpässe. So bemerkt z.B. *M. A. Bassiouni* in (BASS): „... that the explosive proliferation of data processing applications continues to outgrow any advances in technology." Trotz der Entwicklung immer leistungsfähigerer (externer) Speicher hat sich das Verhältnis von Rechenleistung zu Ein-/Ausgabe-Zeiten kaum grundlegend verbessert. Eine Trendwende ist nicht in Sicht.

Andererseits wurden in der Praxis – von der Bilddatenverarbeitung einmal abgesehen – bisher kaum Anstrengungen unternommen, diese Engpässe durch den systematischen Einsatz von Datenkomprimie-

rungs- oder speziellen Kodierverfahren abzumildern. Derartige Verfahren beruhen auf der Tatsache, daß das den verschiedenen Anwendungen zugrundeliegende Datenmaterial oft erhebliche Redundanzen aufzuweisen pflegt. Diese können sowohl durch den Aufbau von Dateien und Datensätzen bedingt sein („strukturelle Redundanz") als auch durch inhaltliche Zusammenhänge („kontextuelle Redundanz").

Im folgenden wird der potentielle Nutzen von Komprimierungs-/Kodierungs-Techniken für die Speicherung und Übertragung von Daten behandelt. Dabei sind auch positive Nebeneffekte etwa hinsichtlich „Datensicherheit" und „Datenschutz" zu berücksichtigen. Nicht geleugnet werden kann, daß es – insbesondere in traditionellen EDV-Umgebungen – auch Argumente gegen den extensiven Einsatz von Komprimierungsverfahren gibt. Solche Gegenargumente sind zu prüfen.

2.1 Datenspeicherung

Die Speicherung von Daten in komprimierter Form auf Hintergrundspeichern bedeutet eine effizientere Nutzung des Speichermediums (allerdings auf Kosten von Verarbeitungszeit für Komprimierung und Dekomprimierung, worauf noch einzugehen sein wird). Sie ist gleichbedeutend mit einer entsprechenden Erhöhung der Kapazität des Mediums ohne den Einsatz von Komprimierungstechniken. (In bestimmten Umgebungen bzw. bei bestimmten Anwendungen kann komprimierte Datenspeicherung sogar zur Einsparung von Büroraum führen. Dies ist ein Aspekt, den man angesichts der hohen Miet- und Unterhaltungskosten für gewerblich genutzte, ggf. klimatisierte und vor unbefugtem Zugang gesicherte Räume nicht unterschätzen sollte.)

Ein wichtiger Nebeneffekt komprimierter Speicherung ist die Verbesserung der Performance durch Reduktion der Anzahl der erforderlichen (mechanischen) Zugriffe sowie der Transfers „Hintergrundspeicher ⇄ Hauptspeicher", wenn hinreichend schnelle Komprimierungs- bzw. Dekomprimierungs-Verfahren verwendet werden. Damit können Komprimierungs- (und zugehörige Dekomprimierungs-)Verfahren zur Basis effizienterer „Daten-Zugriffs-Methoden" werden: So ist es vorstellbar, daß eine existierende „Zugriffs-Methode" (etwa SAM, ISAM, etc.) mit einem Datenkomprimierungsverfahren derart integriert wird, daß bei gleichbleibender „Zugriffs-Schnittstelle" tatsächlich komprimierend geschrieben und dekomprimierend gelesen wird. (Es sind auch Mischformen denkbar, bei denen komprimierte und unkomprimierte Daten durch Satz- oder Datei-Splitting in einer integrierten Organisation gehalten werden: z.B. wenn die Wahrscheinlichkeiten des Zugriffs auf verschiedene Teilmengen von Daten erheblich differieren.)

Ein einfaches Beispiel für diese Integration bietet das IMS-Datenbanksystem („Information Management System") der IBM. Hier besteht die Option, Datensegmente vor der Abspeicherung einem modifizierten Huffman-Kodierungsverfahren zu unterziehen (vgl. CORM). Auch im Datenbanksystem ADABAS werden partiell Komprimierungstechniken verwendet.

Ein weiterer wichtiger Bereich, in dem Komprimierungsverfahren eine entscheidende Rolle spielen können, ist die Archivierung von Daten, die in vielen Unternehmungen eine (durch Gesetz vorgeschriebene oder anwendungsbedingte) Notwendigkeit darstellt. Bei entsprechendem Effekt des eingesetzten Komprimierungsverfahrens kann der direkte wirtschaftliche Nutzen hier sehr hoch sein. Er resultiert zum einen natürlich aus einem wesentlich geringeren Platz- und Materialbedarf (s.o.!), zum anderen aber auch aus der Möglichkeit neuer Organisationsformen der Archivierung. Unter anderem kann zum Beispiel in bestimmten Fällen (etwa bei Banken) eine On-Line-Archivierung von Daten, welche unkomprimiert bisher auf Bändern abgelegt werden, in Betracht gezogen werden.

Im Zusammenhang mit Datenbanken schließlich können Maßnahmen zur Datensicherung wie „Backup" und „Recovery" durch Komprimierungstechniken unterstützt werden. Standardverfahren für „Backup" und „Recovery" bedienen sich der periodischen Speicherung von Datenbankinhalten („Checkpoint-Dump") und der zwischen solchen Speicherungen abgelaufenen Transaktionen („Before- und After-Images", „Transaction-Log", „Audit-Trail"). Die Erstellung komprimierter Backup-Kopien und die Abspeicherung des „Transaction-Log" in komprimierter Form kann zu erheblichen Zeit- und Platz-Ersparnissen bei der Anwendung derartiger Verfahren führen.

En passant sei bemerkt, daß die Datenvolumina, welche bei der Erfassung mittels optischer Verfahren (z.B. Belegleser) entstehen, oft nur durch den Einsatz geeigneter Komprimierungsverfahren zu bewältigen sind.

Einen nicht ganz offensichtlichen Nutzen des Einsatzes von Komprimierungsverfahren bei der Datenspeicherung vermuten wir ferner im Zusammenhang mit der zunehmenden Verbreitung Endbenutzer-orientierter Werkzeuge für den Datenentwurf („Sprachen der 4. Generation"). Während es früher (und auch heute noch überwiegend) meist Spezialisten waren (und sind), welche Datei- und Satz-Strukturen in Hinblick auf eine möglichst effiziente Speicherung zu entwickeln hatten (bzw. haben), wird diese Aufgabe mehr und mehr zum Endanwender hin verlagert, für den der Gesichtspunkt „Speichereffizienz" nicht im Vordergrund steht. Hier nun könnten Datenkomprimierungstechniken einen nivellierenden Einfluß haben, indem nämlich das Komprimat der „Datei des Spezialisten" und das der „Datei des Laien" ungefähr den gleichen Umfang annehmen.

2.2 Datenübertragung

Für diesen Anwendungsbereich sind zwei Aspekte zu unterscheiden: Zum ersten finden innerhalb eines gegebenen Rechnersystems laufend Datenübertragungen über Kanäle statt, zum zweiten – und dies ist der weitaus wichtigere Punkt – besteht ein immer größer werdender Bedarf für Datenübertragungen über Netze,

seien diese nun lokal (LAN) oder überregional (WAN).

Der Nutzen von Datenkomprimierungsverfahren im ersten Fall wurde im wesentlichen bereits im vorangegangenen Abschnitt besprochen: Hier erzielt man einen höheren effektiven Kanaldurchsatz mit allen positiven Folgen. Beispielsweise könnten Antwortzeiten bei Datenbank-Transaktionen drastisch verkürzt werden.

Datenübertragungen in Netzen dienen den verschiedensten Anwendungen und sind in vielerlei Formen üblich. Diese reichen vom Host-zu-Host-Transfer und vom Remote-Terminal-Betrieb über öffentliche Netze (wie z. B. Datex-P, Datex-L und HfD-Leitungen) bis zur Kommunikation mit File-Servern und der Versorgung von Print-Servern in einem lokalen Netz.

Hier gilt – was leicht einsehbar ist –, daß jede durch Komprimierung erzielte Reduktion der effektiv über Leitungen geschleusten Datenmenge nicht nur zu einem Zeitgewinn, sondern auch zu einer enstprechenden Verminderung der Übertragungskosten führt. So lassen sich, etwa im Falle von Mietleitungen, bei Verzicht auf eine größere Nutzdaten-Übertragungsrate, Kosten durch die Auswahl langsamerer und daher billigerer Leitungen einsparen. Diese Einsparungen können sich im Zeitraum eines Jahres zum Beispiel für eine international operierende Bank auf Beträge summieren, deren Größenordnung im Millionen-DM-Bereich liegt. Konkrete Berechnungsbeispiele hierzu finden sich u. a. in (HELD).

Gerade für die Kommunikation über öffentliche Netze (oder gemietete Leitungen) gewinnt man hieraus das wohl stärkste Argument für den Einsatz von Komprimierungstechniken: Die in solchen Netzen dem einzelnen Anwender zur Verfügung gestellte Bandbreite wird im Inland bis auf weiteres auf Übertragungsraten von maximal 64 Kbit/sec bzw. 144 Kbit/sec (bei Ausnutzung aller ISDN-Kanäle) beschränkt bleiben. (Zur Zeit sind wesentlich geringere Raten üblich. Das krasseste Beispiel ist das BTX-Netz, welches im Regelfall seinen Benutzern einen Rückkanal mit der extrem geringen Kapazität von nur 75 Bit/sec zur Verfügung stellt.) Für den Auslandsverkehr werden ähnliche Übertragungsraten (sofern man nicht über ein privates Satelliten-Netz verfügt) noch länger auf sich warten lassen. Hier fallen im übrigen erheblich höhere Kosten an als im Inlandsverkehr.

Die Bewegung komprimierter Daten in Netzen entlastet diverse Netzkomponenten, insbesondere Knoten-, Vor- und Endstellen-Rechner, und führt zu einer höheren Verfügbarkeit der I/O-Ports (d. h., daß sich de facto die Anschlußkapazität erhöht). Store-and-Forward-Techniken sind für komprimierte Daten leichter einsetzbar, weil die erforderliche Container-Kapazität u. U. nennenswert reduziert werden kann. Selbst die Fern-Wartung bzw. -Diagnose von Datenverarbeitungsanlagen kann von der Datenkomprimierung profitieren, zumal hierfür meist langsame Wählleitungen verwendet werden müssen. Es ist schließlich zu überlegen, ob diverse Postdienste wie Teletex und Telefax sich der Datenkomprimierung bedienen könnten (und sollten).

Eine notwendige Voraussetzung für den Einsatz von Komprimierungsverfahren zum Zwecke effizienter Datenkommunikation ist das Vorhandensein intelligenter oder mit speziellen Komponenten ausgerüsteter Endgeräte (Terminals, Remote-Drucker, Geldausgabe-Automat etc.), auf denen die jeweiligen Komprimierungs- und/oder Dekomprimierungs-Vorgänge ablaufen können. Diese Voraussetzung ist bereits beim heutigen Stand der Technologie ohne weiteres erfüllbar.

2.3 Datenschutz

Sowohl die Speicherung von Daten als auch ihre Übertragung in komprimierter Form bieten einen nicht zu vernachlässigenden Schutz vor eventuellem Mißbrauch, falls das verwendete Komprimierungs-Verfahren ausreichend komplex ist. In kritischen Fällen können darüber hinaus Verfahren benutzt werden, welche ihr Komprimat in Abhängigkeit von einem individuell zu wählenden Schlüssel erzeugen. Eine Dekomprimierung kann dann nur mit Kenntnis dieses Schlüssels erfolgen. Zwar wird dabei nicht der Sicherheitsgrad erreicht, den moderne kryptographische Verfahren (etwa Public-Key-Encryption) bieten, andererseits ist dieser hohe Sicherheitsgrad in der Praxis aber nur selten erforderlich.

Interessant erscheint die Möglichkeit, spezielle Verschlüsselungsverfahren für komprimierte Daten zu entwickeln bzw. vorhandene Verfahren auf komprimierte Daten anzusetzen. Da dann weniger Daten zu verschlüsseln sind, können mit u. U. erheblich geringerem Aufwand gleichwertige (bezogen auf die Verschlüsselung unkomprimierter Daten) Ergebnisse erzielt werden.

2.4 Gegenargumente

Im folgenden werden einige mehr oder weniger naheliegende Argumente gegen den Einsatz von Komprimierungsverfahren aufgeführt. In jedem Fall werden wir versuchen, eine entkräftende Antwort zu geben. Wir beziehen uns in diesem Abschnitt teilweise auf (BASS).

Das nächstliegende Gegenargument dürfte durch den Overhead an Zeit und Speicherplatz begründet sein, welcher bei der Anwendung von Komprimierungs- bzw. Dekomprimierungs-Verfahren entsteht. In der Tat ist – zum Beispiel bei komprimierter Speicherung im Hintergrund – jeweils sehr genau zu prüfen, ob etwa der durch den Kompressionseffekt hervorgerufene Zeitgewinn beim (mechanischen) Datenzugriff incl. Datentransfer groß genug ist, um die zusätzliche Verarbeitungszeit zu rechtfertigen. Dagegen werden im Falle der Datenübertragung über relativ langsame und teure (öffentliche) Leitungen bzw. Netze die Kosten des genannten Overheads mit Sicherheit um Größenordnungen geringer sein als die durch die Reduktion des zu übertragenden Datenvolumens eingesparten Kosten. Dieser Overhead – im wesentlichen bedingt durch die Tatsache, daß Komprimierungsverfahren bisher fast ausschließlich in Software implementiert wur-

den – begründet allerdings durchaus stichhaltige Einwände gegen den Einsatz dieser Verfahren im Zusammenhang mit der Datenspeicherung. Neuere Entwicklungen – soweit uns bekannt durchweg im Laborstadium (vgl. z.B. HAWT) – zielen daher darauf ab, Komprimierungsverfahren in Hardware zu implementieren. Ermöglicht wird dies durch die bekannten Entwicklungen auf dem Gebiet der Mikroelektronik. Die Hardware-Implementierung (etwa durch asynchron arbeitende Spezial-Prozessoren) von Komprimierungsverfahren ist mittelfristig vermutlich einfacher, als wesentlich leistungsfähigere (nicht-mechanische) Datenträger bzw. -Speicher zu entwickeln (d.h. Mechanik durch Elektronik zu ersetzen), da bei der rasanten Leistungsverbesserung der Prozessortechnik die Ein-/Ausgabe der Engpaß bleibt.

Ein weiteres Gegenargument kann dann zum Tragen kommen, wenn Daten auch in komprimierter Form bestimmten anwendungsbezogenen oder durch die jeweilige Speichertechnik bedingten Verarbeitungen (wie Sortieren, Suchen) unterworfen werden sollen. (Ersteres wäre etwa dann notwendig, wenn die mit jedem Zugriff einhergehenden Dekompressionen mit unvertretbarem Aufwand verbunden sind. Insofern besteht hier ein Zusammenhang mit dem oben erwähnten Overhead.) In der Regel zerstören aber Komprimierungsalgorithmen Eigenschaften von Daten, welche für derartige Verarbeitungen Voraussetzungen sind (z.B. lexikalische Ordnung). Indem Hardware-Implementierungen Kompressionsverfahren erheblich beschleunigen könnten, ergeben sie auch hier eine Entkräftung des Arguments. Außerdem gilt, daß bei vielen Anwendungen Daten ohne echte Verarbeitungsabsicht lediglich „hin und her" bewegt werden (z.B. Kopieren). Solche „1:1"-Operationen laufen mit komprimierten Daten dann natürlich entsprechend schneller ab. Selbst Sortiervorgänge können durch Komprimierung der Schlüsselfelder verkürzt werden.

Während die beiden zuerst diskutierten Gegenargumente den Aspekt der Datenspeicherung betrafen, wird ein dritter Einwand hauptsächlich im Hinblick auf die Datenübertragung erhoben: Der Einwand nämlich, die Übertragung komprimierter Daten könne die Zuverlässigkeit etwa beim File-Transfer, aber auch bei DB/DC-Anwendungen herabsetzen. Tatsächlich führen Fehler bei der Übertragung komprimierter Daten im allgemeinen dazu, daß die Daten beim Empfänger nicht wiederherstellbar sind. Mehrere Erwiderungen auf diesen Einwand sind denkbar: Zum einen führen datensichernde Leitungsprotokolle (wie HDLC, SDLC) heute zu einer hohen Zuverlässigkeit der Übertragung selbst, zweitens wird die absolute Fehlerhäufigkeit beim Transfer komprimierter Dateien entsprechend geringer und drittens können auch auf der Ebene des „Dekomprimierers" (in ISO-OSI-Terminologie also der Darstellungsebene oder sogar der Anwendungsebene) zusätzliche Sicherungsmaßnahmen (vom „Komprimierer" erzeugte Checksummen etwa) eingeführt werden, deren Overhead (bezogen auf Übertragungskosten, s.o.) verschwindend gering ist. Zielt man auf die Hardware-Implementierung eines Komprimierungsverfahrens, so kann es schließlich sinnvoll sein, spezielle Fehlererkennungs- und Korrekturtechniken zu entwikkeln.

3 Gebräuchliche Komprimierungsverfahren

Zahlreiche Komprimierungsverfahren sind in der Vergangenheit entwickelt worden. Dabei hatte man mitunter sehr spezielle Anwendungen (etwa im Datenbankbereich, z.B. Index-Kodierung) im Auge. Es würde an dieser Stelle daher zu weit führen, alle diese Verfahren im Detail zu beschreiben. Verwiesen sei auf die Monographie (HELD) sowie auf den Aufsatz (BASS). Einige typische Verfahren, welche heute in der Praxis gerne benutzt werden, wollen wir jedoch kurz darstellen und Kriterien für ihre Bewertung angeben.

Diese Verfahren lassen sich im wesentlichen in zwei Klassen einteilen: Lauf-Längen-Kodierung und statistische Kodierung (letztere geht zurück auf eine Arbeit von Huffman, zitiert in (BASS), (CORM) und (HELD)).

Lauf-Längen-Kodierung („run-length encoding") nutzt die Tatsache aus, daß sich in (für kommerzielle Anwendungen typischen) Datensätzen häufig Strings größerer Länge befinden, in welchen jeweils ein Zeichen (BLANK, NULL, Sonderzeichen etc.) wiederholt wird. Solche Strings können dann reduziert werden auf eine Folge bestehend aus einem geeigneten „Fluchtzeichen", einem Zähler und dem Zeichen, welches im Original an dieser Stelle so oft wiederholt wird, wie der Zähler angibt. Lauf-Längen-Kodierung, welche sich nur auf ein einziges (häufig wiederholtes) Zeichen bezieht, wird auch als „Null-Unterdrückung" („Null-Suppression") bezeichnet. In dieser Variante wird das Verfahren zum Beispiel bereits im IBM 3780 BISYNC-Protokoll angewendet. Weitere Varianten ergeben sich, wenn man etwa die Bedingung, daß nur die Wiederholung jeweils gleicher Zeichen berücksichtigt wird, lokkert und stattdessen auch zuläßt, daß der zu komprimierende String aus Zeichen bestehen darf, deren Codes (in der gewählten Kodierung, ASCII, EBCDIC etc.) strukturverwandt sind. Das wohl bekannteste derartige Verfahren ist das „Byte Packing".

Statistische Kodierungs- (bzw. Komprimierungs-) Verfahren beruhen auf der Feststellung der Häufigkeiten einzelner Zeichen oder Zeichengruppen in einer gegebenen Datenmenge. Diesen Häufigkeiten entsprechend werden Bitfolgen variabler Länge als optimale Codes für Zeichen bzw. Zeichengruppen konstruiert. Optimiert wird dabei jeweils die mittlere Codelänge. Die Information über diese Codes muß dann natürlich als Teil des Komprimats betrachtet werden.

Kriterien zur Beurteilung von Komprimierungsverfahren liegen auf der Hand. Wichtigstes Kriterium ist selbstverständlich der Kompressionseffekt, der ausgedrückt werden kann als Verhältnis zwischen der Größe des Komprimats (inklusive der zur Dekomprimierung eventuell notwendigen Zusatzinformation) und der der originalen Datenmenge: je kleiner also dieses Verhältnis, um so besser wirkt ein Komprimierungsverfahren.

Ein zweites wichtiges Kriterium ist die algorithmische Komplexität des Verfahrens, d.h. sein Bedarf an Rechenzeit und Speicherplatz in Abhängigkeit vom Umfang der zu behandelnden Datenmenge. Zwischen der algorithmischen Komplexität und dem Effekt eines Verfahrens besteht ein „Trade-off"-Zusammenhang: Ein besserer Kompressionseffekt verlangt einen höheren Berechnungsaufwand. Exakte theoretische Resultate liegen hierzu unseres Wissens bisher nicht vor. Einige für praktische Anwendungen hinreichende Heuristiken enthält z.B. (HELD).

Die Hardware-Implementierbarkeit eines Verfahrens ist dann ein wichtiges Beurteilungskriterium, wenn solche Anwendungen zu unterstützen sind, bei denen es auf hohe Verarbeitungsgeschwindigkeiten ankommt.

4 „Frankenstein-Lidzba-Verfahren"

Das Verfahren, soweit es Gegenstand der Patentanmeldung ist, wird in (LIWI) ausführlich beschrieben. Seine Grundidee ist so einfach wie überraschend: Ausgangspunkt ist die Beobachtung, daß die Sätze von für kommerzielle Anwendungen typischen Dateien in der Regel strukturverwandt sind (z.B. befinden sich Felder gleichen Typs jeweils an der gleichen Position). Betrachtet werden nun N ($8 <= N <= 256$) Sätze, welche man sich als Zeilen eines „Blocks" vorzustellen hat. Geeignete („virtuelle", also nicht physische) Umordnung dieser Sätze ergibt dann einen Block, in dessen „Spalten" längere, aus jeweils nur einem Zeichen bestehende Folgen auftauchen. Die Spalten werden nun (im Prinzip) einer Lauf-Längen-Kodierung unterzogen. Diese Grundidee wird um zahlreiche weitere Techniken und „Tricks" (u.a. „Byte-Packing" und auch statistische Analysen) ergänzt. Das charakterisierende Stichwort lautet also: „vertikale" statt „horizontale" Komprimierung, bzw. Kompression einer „Satzmenge" als Ganzes, nicht „Satz für Satz".

Eine Besonderheit des Verfahrens besteht darin, daß der Vorbehandlung der Daten große Aufmerksamkeit geschenkt wird. Diese „Homogenisierung" führt zur Entstehung längerer Strings gleicher Struktur bzw. zur Häufung des Auftretens einzelner Codes. Obwohl nicht auf den ersten Blick erkennbar, liegt hierin eine Verwandtschaft zur mehrdimensionalen Bilddatenkomprimierung. Somit ist das Verfahren auch zur Gewinnung komprimierter physikalischer Nettodaten in TP-Anwendungen geeignet.

Quantitative Aussagen über den Kompressionseffekt sowie den Rechenzeit- und Speicherplatz-Bedarf des Verfahrens konnten mit Hilfe einer in Software (Assembler-Routinen unter BS2000) implementierten Version gewonnen werden. Bemerkenswert ist die Tatsache, daß das Verfahren, angewandt auf typische kommerzielle Dateien (s.o.), einen erheblich besseren Effekt erzielt als herkömmliche Methoden. In (CORM), einer der neuesten Arbeiten über Komprimierung im Datenbankbereich, wird für eine Datei der genannten Art (eine „Student-Records Database") ein Effekt von ca. 58% (Restmenge) berichtet. Verwendet wurde dabei eine statistische Kodierung in Anlehnung an das Huffman-Verfahren. Mit dem „Frankenstein-Lidzba-Verfahren" konnten wir für ähnliche Dateien die wesentlich günstigeren Effekte von <20% (Restmenge) erzielen. Aber selbst für Daten, welche nicht dieser Kategorie zuzuordnen sind (z.B. den für die Treiber von Ausgabegeräten bestimmten Output des Satz- und Formatiersystems TeX) waren Effekte von ca. 40% (Restmenge) feststellbar.

Die gemessenen Werte von Rechenzeiten (im Bereich von 5–10 sec CPU-Zeit je MByte für die Komprimierung, ca. 3–6 sec je MByte für die Dekomprimierung, SIEMENS 7.561, 2.4 MIPS) sind naturgemäß schwerer zu bewerten. Wir verweisen in diesem Zusammenhang jedoch auf (positive) Untersuchungen (vgl. (WEVE) und (LIWI)) zur Hardware-Implementierbarkeit (unter Ausnutzung der Parallelisierbarkeit von Berechnungen) des „Frankenstein-Lidzba-Verfahrens".

Eine Weiterentwicklung der bei unseren Tests verwendeten Software-Version wird inzwischen unter der Bezeichnung „FLAM" (als Lizenzprodukt der Firma limes datentechnik gmbh, Friedrichsdorf/Taunus) für den Einsatz unter BS2000 und OS/MVS (hier nur auf SAM-Dateien anwendbar) angeboten.

Einige typische Testergebnisse werden im folgenden tabellarisch aufgeführt:

1. Anlage SIEMENS 7.561, BS2000, 2.4 MIPS

Dateiinhalt	Typ	Umfang (Byte)	Restmenge %
TeX-DVI	SAM	1.082.896	42,7
Personaldaten	ISAM	487.895	10,9
COBOL-Source	ISAM	65.570	46,0
ASSEMBL.-Listen	SAM	4.873.581	22,2
ADA-Zwi.-Code	SAM	6.887.520	20,7
LISP-Source	SAM	591.044	53,0

2. Anlage SIEMENS 7.570-G, BS2000, 4.4 MIPS

Dateiinhalt	Umfang (Byte)	Restmenge (%)
Komprimierungs-Protok.	779.871	14,6
BS2000-Macro-Bib.	2.079.968	51,0
BS2000-Bedienpl.-Prot.	4.231.557	27,9
Banken-Datentr.-Austausch	346.002	32,1
BTX-Anwendung	6.371.046	15,4
BS2000-REP-Datei	603.204	35,8
BS2000-Software-Monitor	796.566	36,6

Es ist zu bemerken, daß das von uns verwendete Programm das eigentliche Verfahren – aus Performance-Gründen – nur begrenzt abbildete. Andererseits ist z.B. bei Verwendung der mit der Patentanmeldung (vgl. (LIWI)) vorgelegten Hardware-Lösung ein noch um einiges besserer Kompressionseffekt zu erwarten.

Die obigen Beispiele zeigen dennoch recht eindrucksvoll, daß gerade Dateien, deren Sätze in hohem Maße strukturverwandt sind, durch das „Frankenstein-Lidzba-Verfahren" besonders stark schrumpfen (Personaldaten, BTX-Anwendung). Andererseits macht es wenig Sinn, Daten, welche bereits einer anwendungsspezifischen Kompression unterzogen wurden, nachträglich mit dem Verfahren zu behandeln. Verzichtet man aber auf anwendungsspezifische Komprimierung und setzt statt dessen von vornherein das „Frankenstein-Lidzba-Verfahren" ein, so ist der Kompressionseffekt in der Regel besser.

5 Fazit

Datenkomprimierung hat ein breites und sinnvolles Anwendungsfeld. Ihr ökonomischer Nutzen ist nicht nur für die reinen Anwender beträchtlich. Auch Hersteller von Datenverarbeitungsanlagen, die dies zuweilen aufgrund ihrer Interessenslage mit Skepsis zur Kenntnis nehmen, können erheblich profitieren, weil Datenkomprimierung mit geeigneten Effekten prozess- und produktinnovativ sein kann und sich aus ihr neue Anwendungsmöglichkeiten für die Datenverarbeitung schlechthin ableiten lassen. (Für den normierten Austausch von Daten – wie z.B. Datenträgeraustausch zwischen Banken oder Datenaustausch gemäß DÜVO mit Versicherungsanstalten – sollte beispielsweise eine „Komprimierungs-Option" zugelassen werden.)

Das „Frankenstein-Lidzba-Verfahren" ergibt nach unserer Erkenntnis für typische kommerzielle Daten einen erheblich besseren Effekt als traditionell übliche Verfahren. Als Produkt wird es seit Jahresbeginn erfolgreich bei Banken und im File Transfer eingesetzt.

Literatur

(BASS) *Bassiouni, M. A.:* Data compression in scientific and statistical databases; IEEE Trans. on Software Engineering, Vol. SE-11, No. 10, pp. 1047–1058, October 1985

(CORM) *Cormack, G. V.:* Data compression on a database system; Communications of the ACM, Vol. 28, No. 2, pp. 1336–1342, December 1985

(HAWT) *Hawthorn, P.:* Microprocessor assisted tuple access decompression and assembly for statistical database systems; in Proc. Very Large Data Bases (VLDB) 1982, pp. 223–233

(HELD) *Held, G.:* Data Compression – Techniques and Applications, Hardware and Software Considerations; J. Wiley & Sons, Chichester 1983

(LIWI) *Lidzba, R., H.-U. Wiebach:* Verfahren zum Komprimieren und Dekomprimieren mehrerer strukturverwandter Datenfolgen sowie Einrichtungen zur Durchführung des Verfahrens; Patentanmeldung No. P 35 25 898.5 beim Deutschen Patentamt, München 1985

(WEVE) *Wevelsiep:* Gutachten zur Hardware-Implementierbarkeit des „Frankenstein-Lidzba-Verfahrens", 1985 (unveröffentlicht)

(1006)